Incremental Unit Testing

STAP Group

Shanghai Jiao Tong University

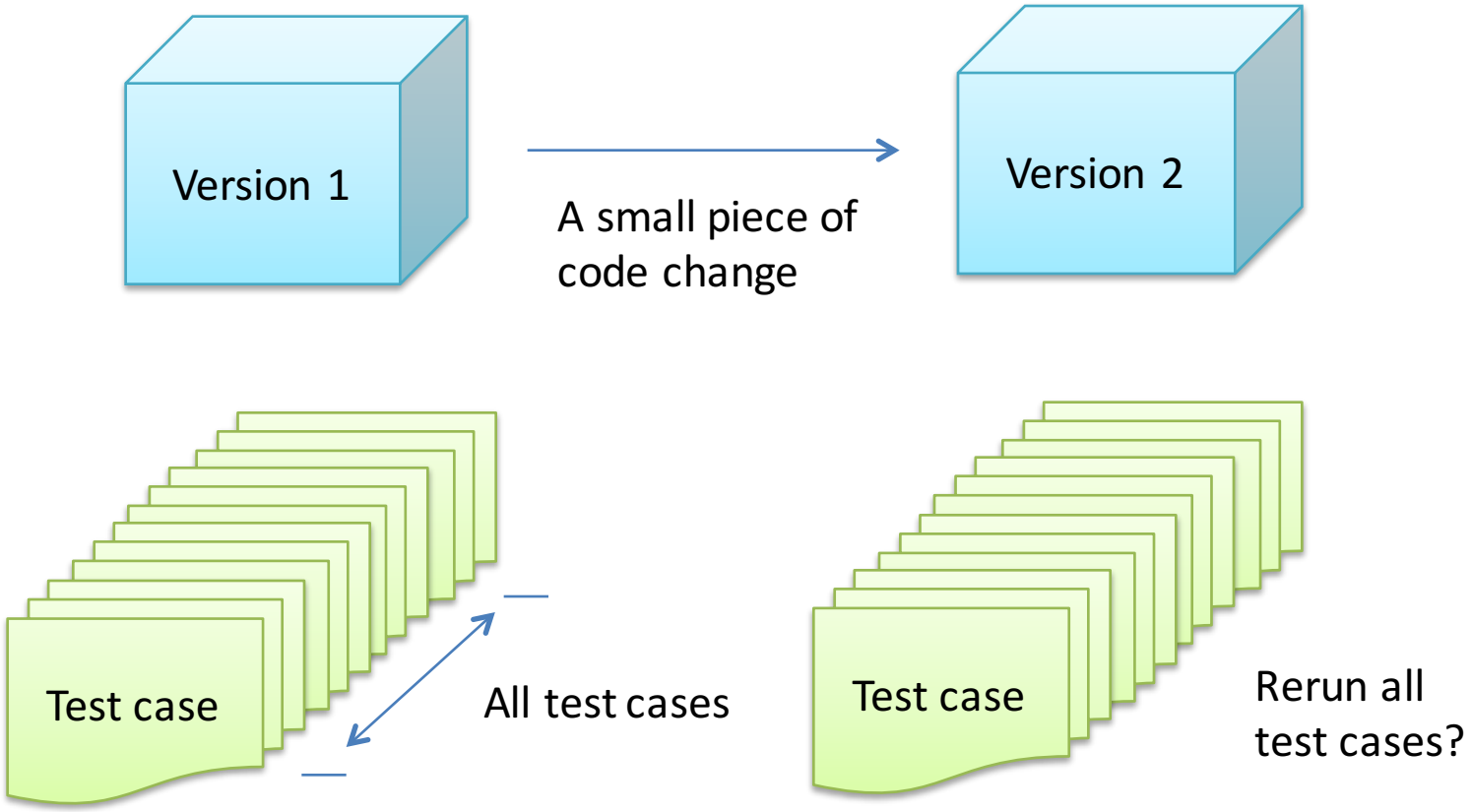
Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

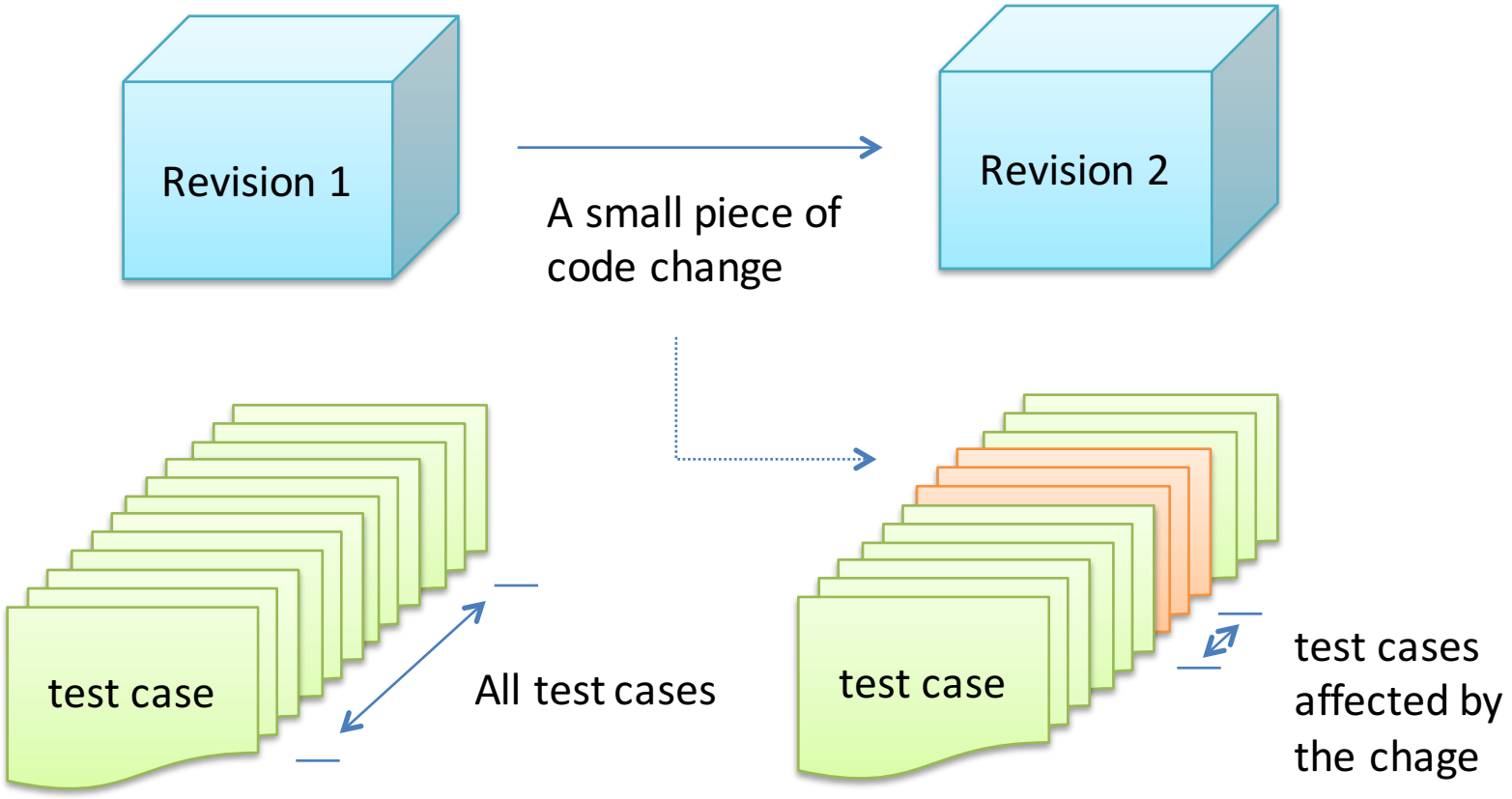
Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

Introduction



Introduction



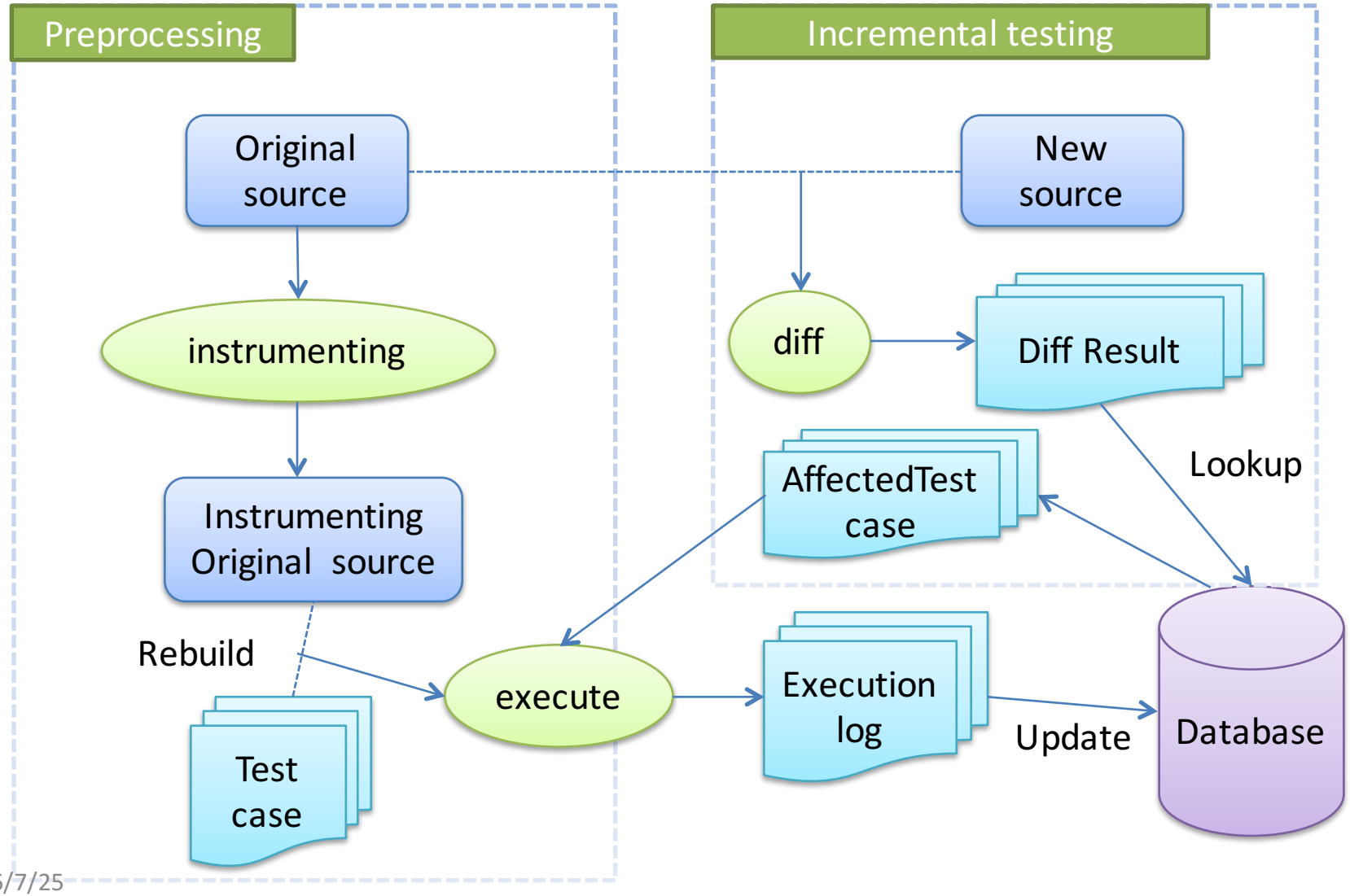
Agenda

- Introduction
- Design
- C++ Implementation
- Java Refinement
- Demo
- Future Work

Design

- Instrument original codes to log the function coverage of every test case and write them to database.
- Find differences of two versions of codes in level of function.
- Select the test cases that cover the changed codes and rerun them.

Workflow of the Project



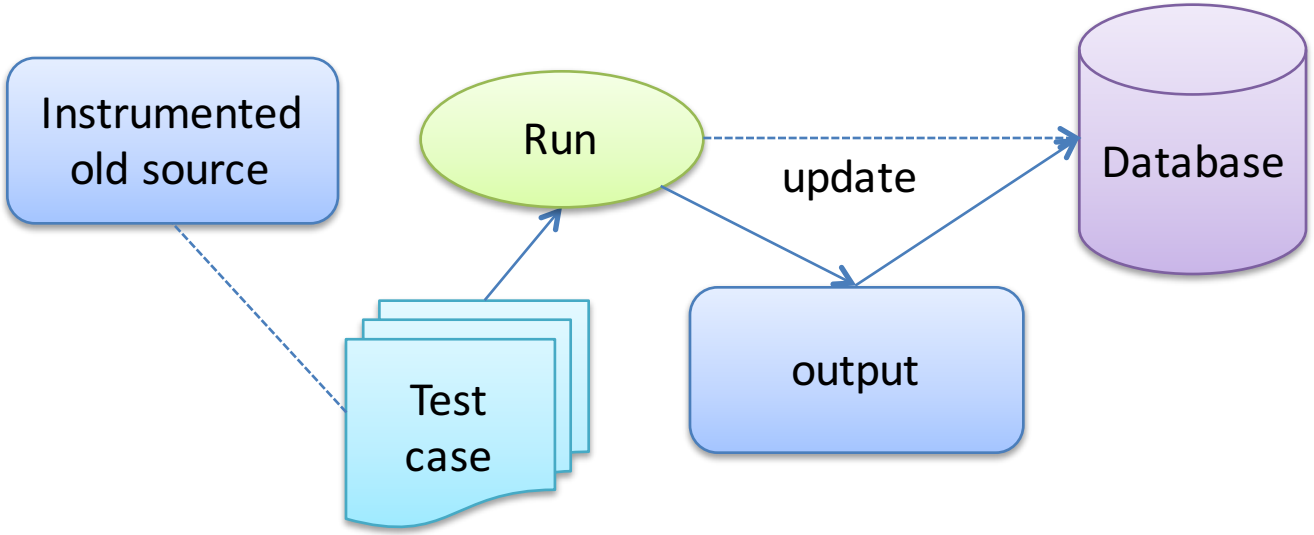
Instrumentation

- Find all functions using AST building tool.
- Insert a log statement at the beginning of every function to print out the signature of the function.

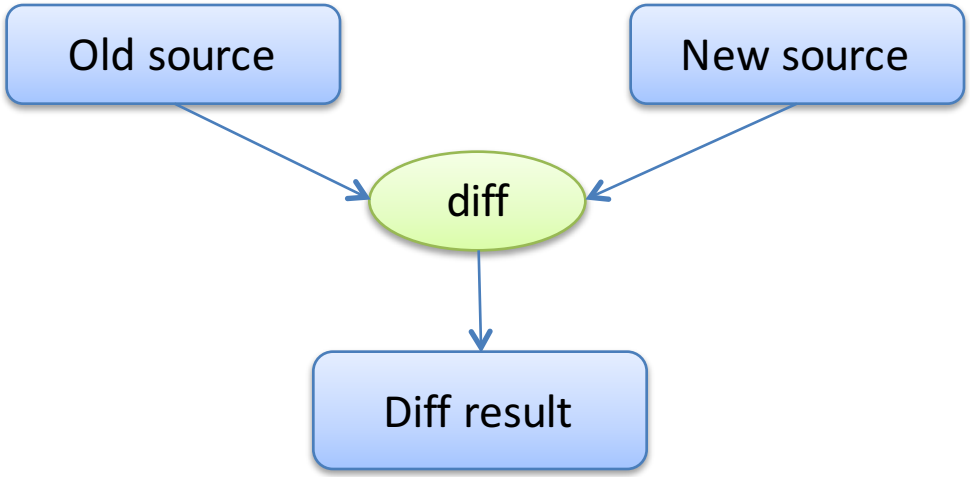
Coverage

- Run the instrumented codes.
- Parse the output and get the functions covered by every test case.
- Write the coverage to database.

Coverage


Instrumented
old source
Run
update
Database
Test
case
output

Diff



- Transform source into AST form
- Compare AST structure and content
- Diff result:
  - File: added/deleted/modified
  - Function: added/deleted/modified
  - Other element

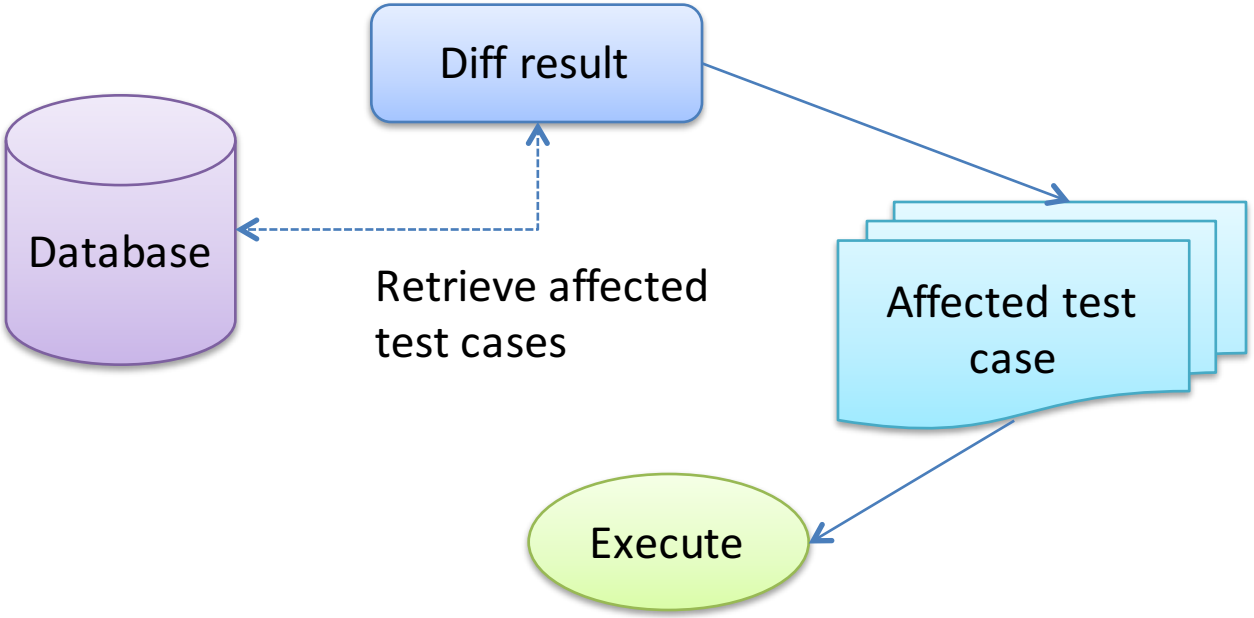
Differ Result

- File
  - Added: rerun all test cases
  - Modified:
    - Function
    - Other elements
- Function
  - Added: rerun all test cases
  - Modified: rerun affected test cases
- Other element
  - any change: rerun all test cases

Select

- select(coverage, difference) -> test cases.
- Rerun them.

Diff & Select



- File
- Function
- Changes of program elements besides functions

Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

C++ IUT Implementation

- A command-line tool ‘iutc’
- Main sub-commands included in ‘iutc’
  - i
  - c
  - s
- Using sqlite3 as database to save record

Sub-Commands

- iutc -i
  - Initialize a config template in the current directory. Fill it.
- iutc –c
  - Generate a .db file that records the coverage and project version.
- iutc -s
  - Select affected test cases and rerun.

Implementation Status

- Basic incremental unit testing tool for C++ Project
  - Instrumentation source code for logging
  - Find Function level difference between two source folder
- Maintain a database to keep track the relationship between test case and function
- A simple bash script to support the workflow ,and each sub-command in this workflow is implemented

Limitations

- Only available for projects that use Google Test framework
- After rerunning the affected test cases or rerunning all test cases, the iut program doesn’t update the running result to the database.

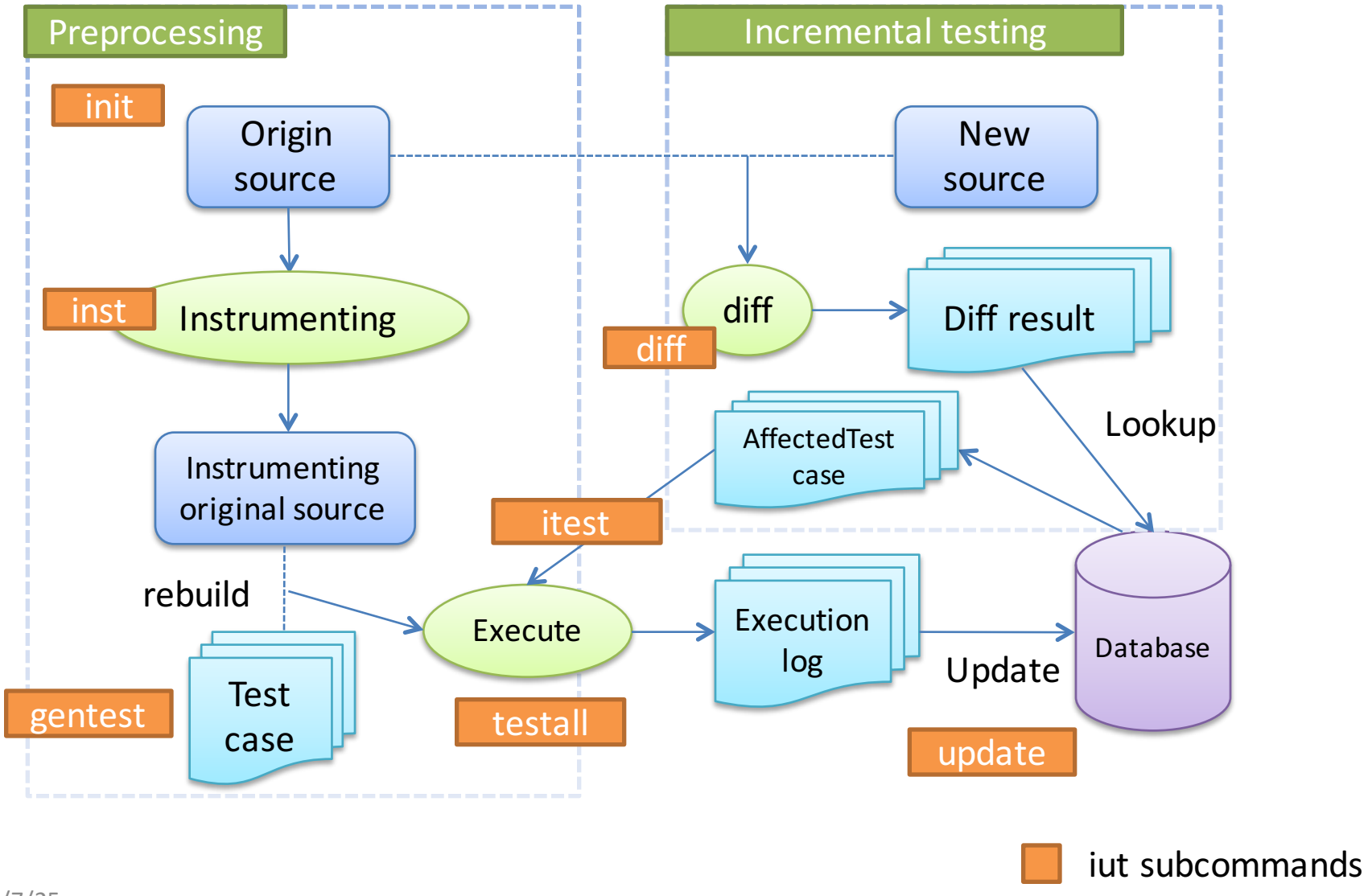
Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

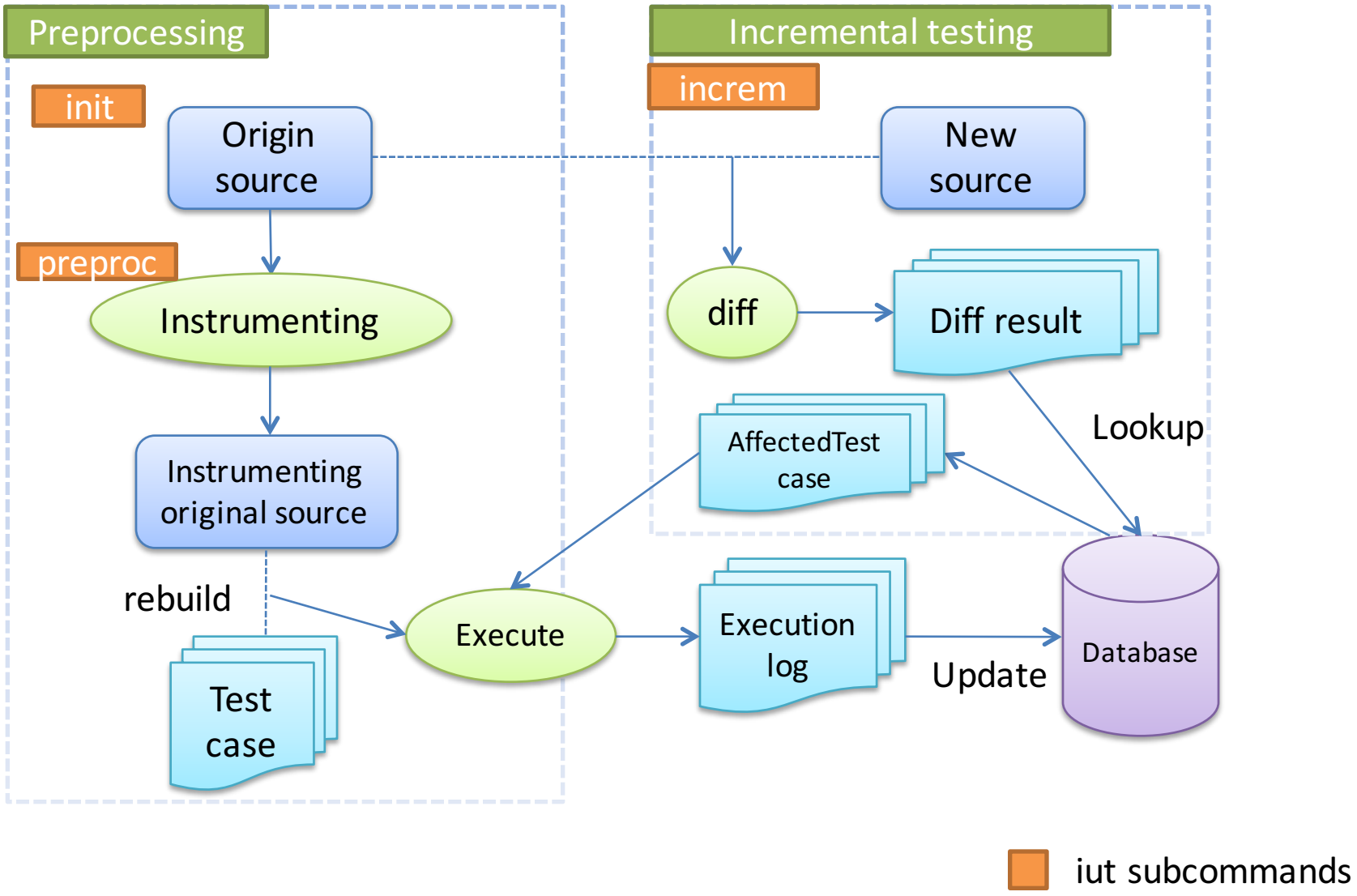
Java IUT Refinement

- The Former 7 subcommands to support the workflow have been reduced to only 3 subcommands.

Former Workflow



Simplified Workflow



Workflow Simplification

Phase	New Command	Old Commands	Description
Prerequisite	N/A	N/A	1. Setting IUT_HOME folder -> $HOME/.iut (which contains files that are required to run iut. 2. Modify the "build.gradle" file of the target project
Phase1: Preprocessing	• *iut init <projectName>*	• *iut init <projectName>*	Initialize db file and several settings.
	• *iut preproc <projectName>*	• *iut inst* • *iut gentest* • *iut testall* • *iut update*	instrumentation -> run testcases -> update database
Phase2: Incremental Testing	• *iut increm <projectName> <oldVersionPath> <newVersionPath>*	• *iut diff* • *iut itest*	diff to find changed methods -> query for affected testcases -> run affected testcases

Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

DEMO

Agenda

- Introduction
- Design
- C++ IUT Implementation
- Java IUT Refinement
- Demo
- Future Work

Future Work

- Support smooth multi-version evolution. Now the demo are based on two version.
- Identify and analyze more special cases when comparing versions of codes.
- Integration with version control system.

Conclusion

- We have implemented a tool to support all basic functionalities of the incremental unit testing for C++ projects.
- We have simplified the workflow of incremental unit testing, and applied it to both the C++ and the former Java IUT project.
- The tool has some limitations and further improvements

Thanks to Morgan Stanley for supporting the project!