CS3203: Software Engineering Project

Testing

By: Dr. Bimlesh Wadhwa

NUS National University of Singapore | School of Computing

Testing Objectives

Testing is the process of executing a program with the intention of finding errors.

– Myers –

Testing can show the presence of bugs but never their absence.

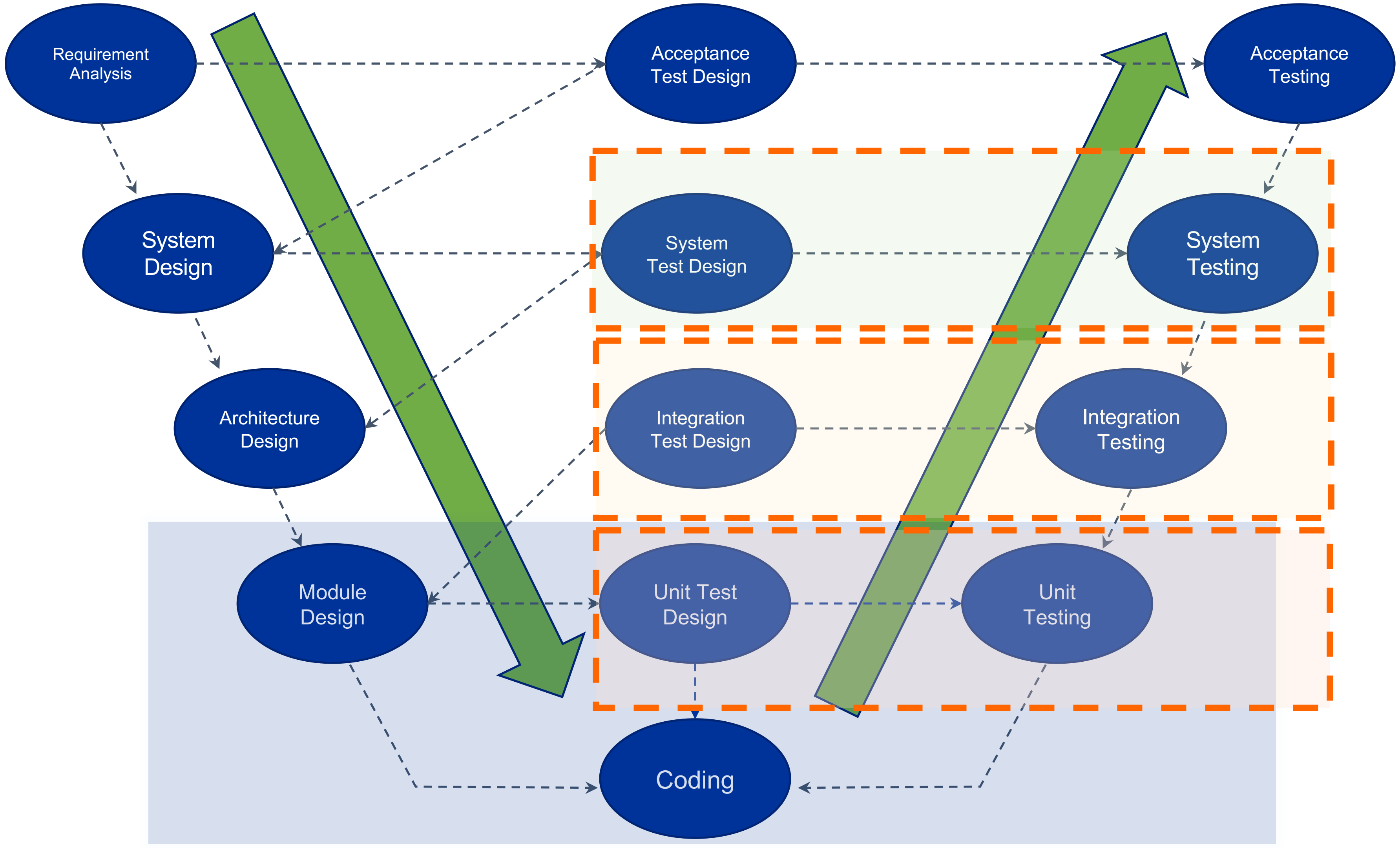
– Dijkstra –

General Guidelines

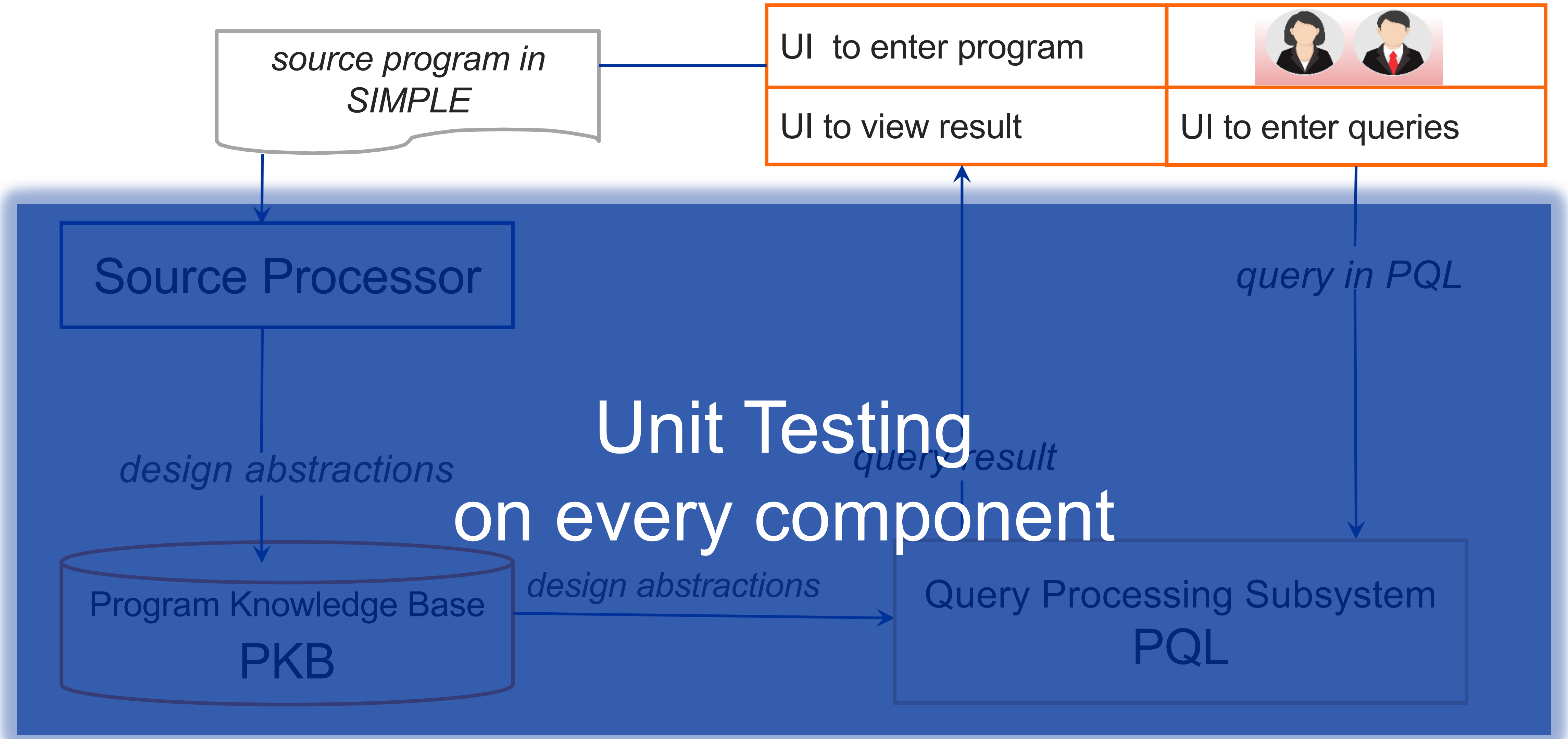
- Determine if the software meets all of the requirements
- Avoid non-reproducible or on-the-fly testing
- Inspect the results of each test
- Probability of undetected defects increases with the number of detected defects (bugs)

Development and testing can be done by different members!

Testing Levels

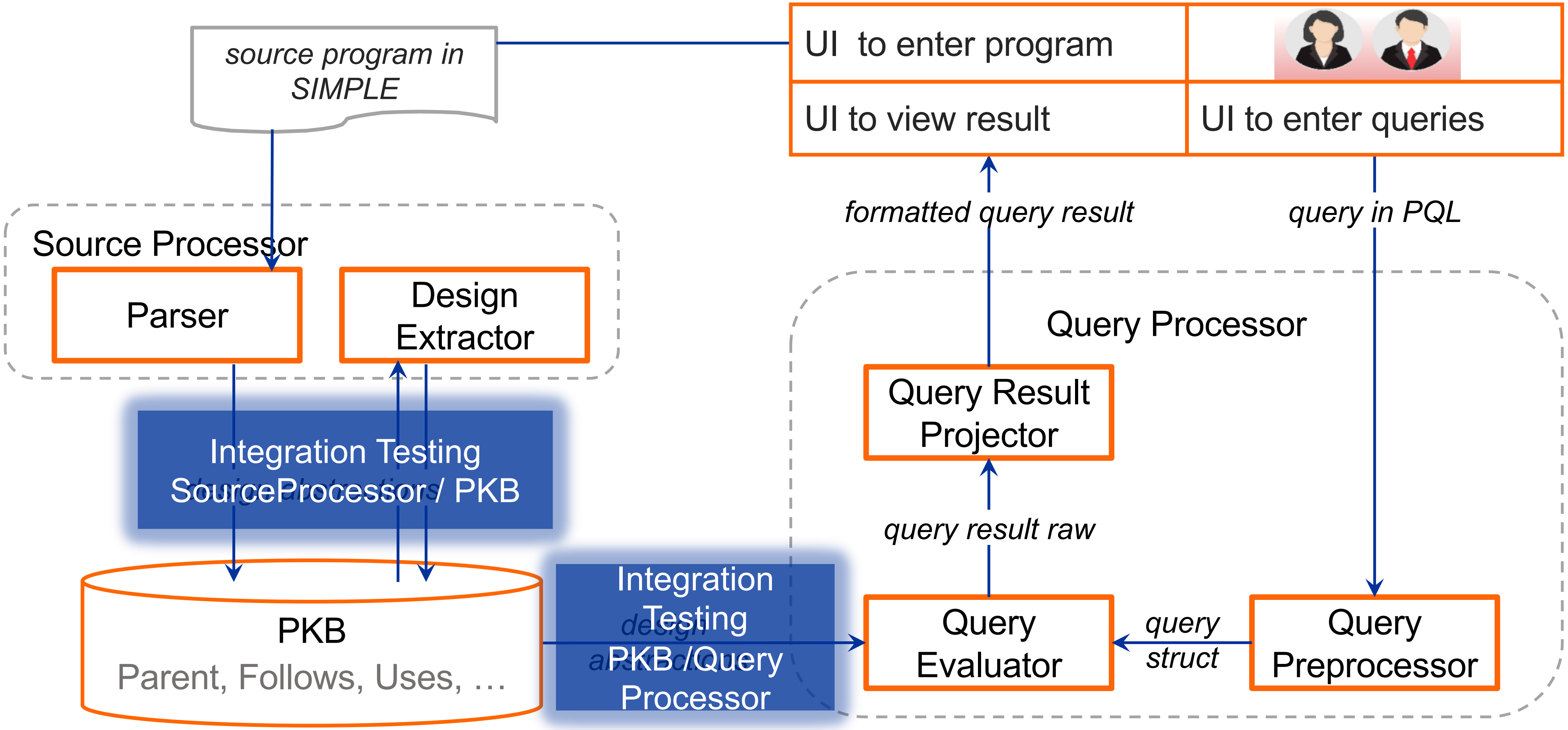


SPA Architecture – Unit Testing

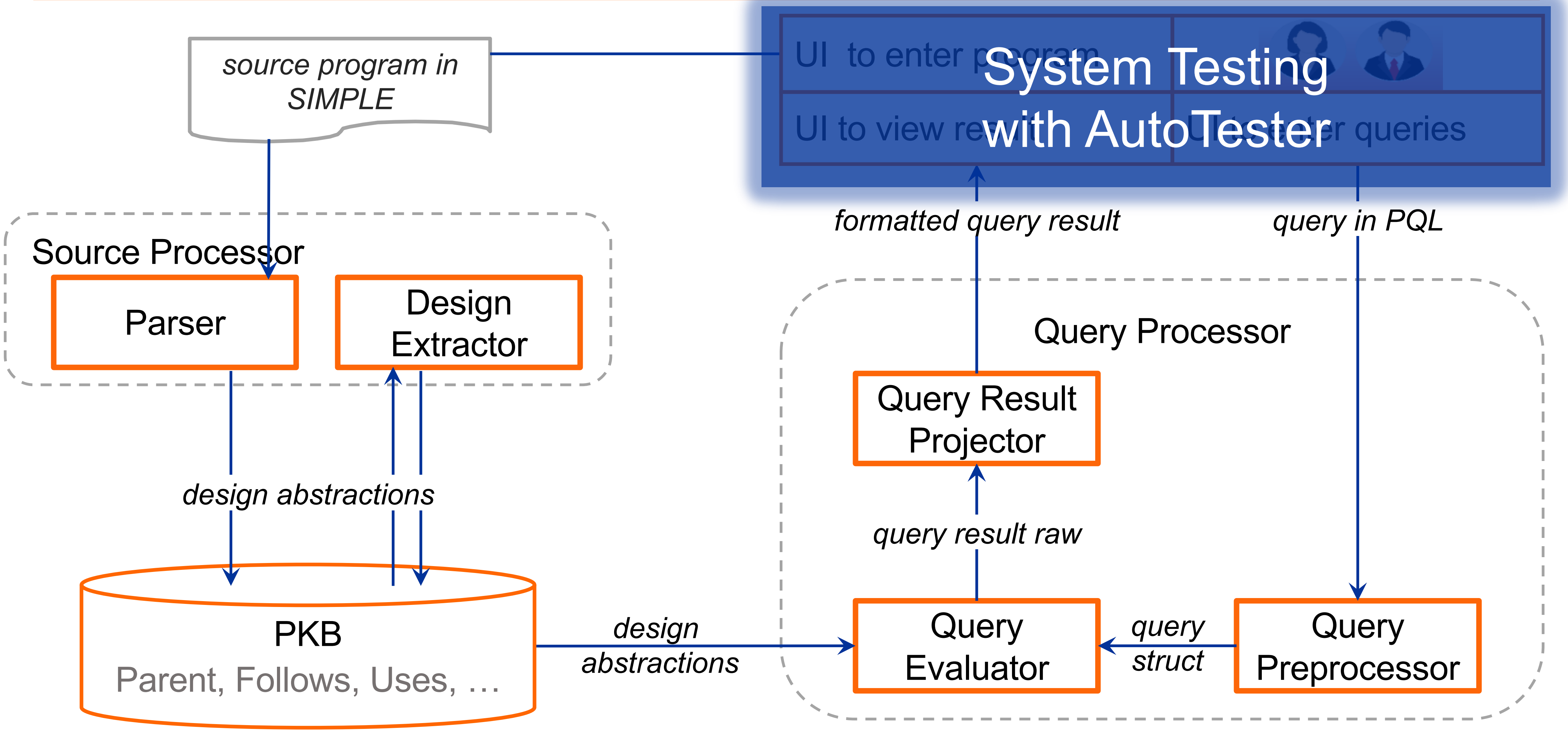


Unit Testing
on every component

SPA Architecture – Integration Testing



SPA Architecture – System Testing



Phases in System Testing

Plan	analysis and design of test cases
Procedure	documents describing the existing test cases, scripts, method of executing
Report	documents containing the test results
Tracking	follow up on bugs and use regression testing

Plan and Prepare

- Analysis
- Estimation
- Design and informal validation
- Validation readiness review and formal validation

Test Analysis

GENERAL

- Review - test basis, testability
- Identify test requirements and test data
- Identify test infrastructure and tools

PROJECT SPECIFIC

- SPA testing needs SIMPLE source and query files
  - e.g. One SIMPLE source for each set of test cases
- Features to test: Modifies, Uses, etc
- Use AutoTester
- Design scripts to run and identify bugs

Test Estimation

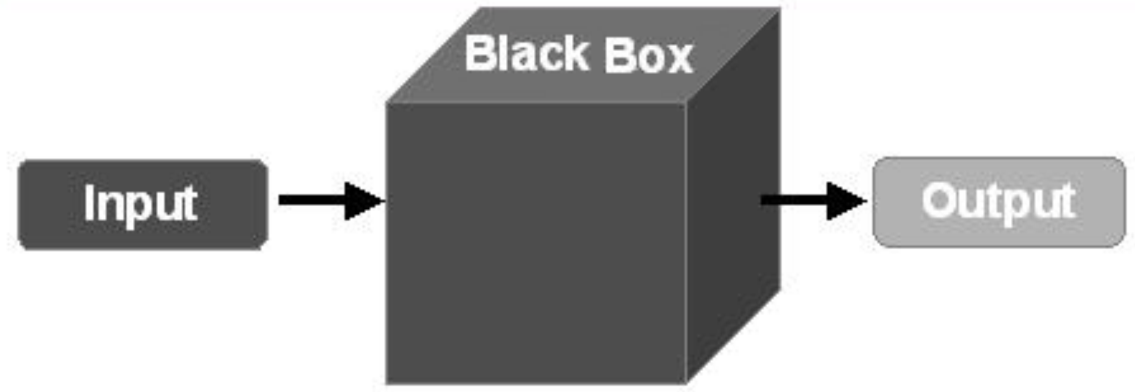
- Complexity
  - Many small tests or a few large ones?
- Different platforms
  - How easy is it to setup and run on a different machine?
- Automated or manual tests
  - Use a script to run the tests or run each test manually?

Tips for Test Planning

- Estimate test development time
  - Number of tests: 300
  - Average test development time: 5 mins/test
  - Estimated time: 25 hours
- Plan for easy execution
  - AutoTester integration
  - Write scripts to automatically run tests
  - Estimated time: 10 hours
- Test **early** and **regularly**!

Test Design Technique

- Specification based (Black Box Testing)
  - Examines the functionality of an application without peering into its internal structures or workings.
  - Requires understanding of the specifications and requirements.
  - aka functional testing or use-case based testing



Types of Tests

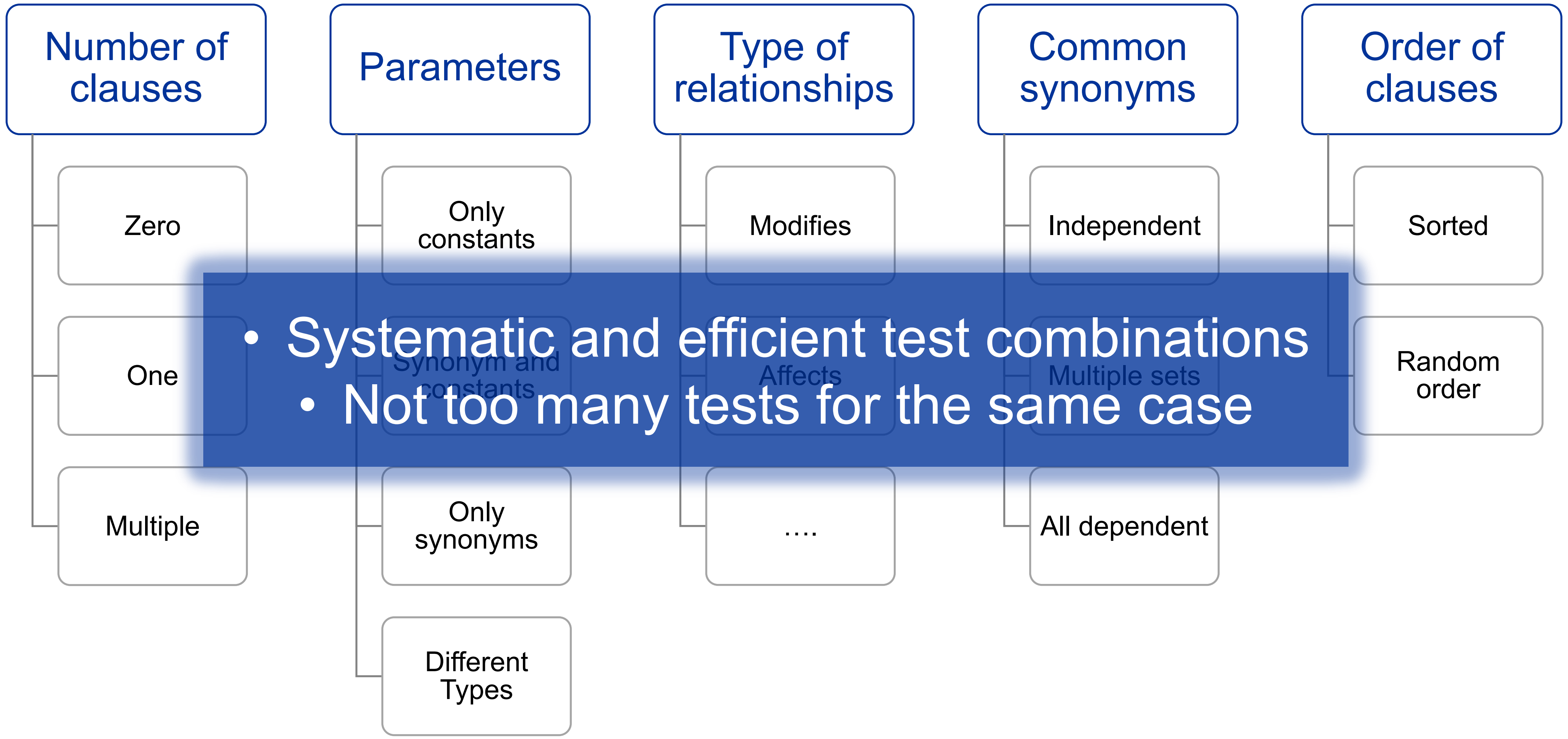
General	Project specific
Functional tests	Correct source and queries
Algorithmic tests	Correct computation of each relationship
Positive/negative tests	Gracefully handle invalid input
Usability tests	Acceptance of correct inputs; error messages
Boundary tests	Queries with first, last, non-existing statements
Load/stress tests	Complex queries: multiple clauses of different types

Design of Test Cases for PQL

- Number of clauses
  - Zero
  - One
  - Multiple
- Parameters
  - Only constants
  - Synonym and constants
  - Only synonyms
  - Different Types
- Type of relationships
  - Modifies
  - Affects
  - ….
- Common synonyms
  - Independent
  - Multiple sets
  - All dependent
- Order of clauses
  - Sorted
  - Random order

*** Multiple clauses, optimization and new relationships eg Affects will be introduced in advanced SPA requirements in iteration 2 and 3.*

Design of Test Cases for PQL



- Systematic and efficient test combinations
  - Not too many tests for the same case

Documenting Test Cases

- The purpose of a test case and description
- Required inputs to a program
- Expected results produced by a program
- Any other requirements for running a test case

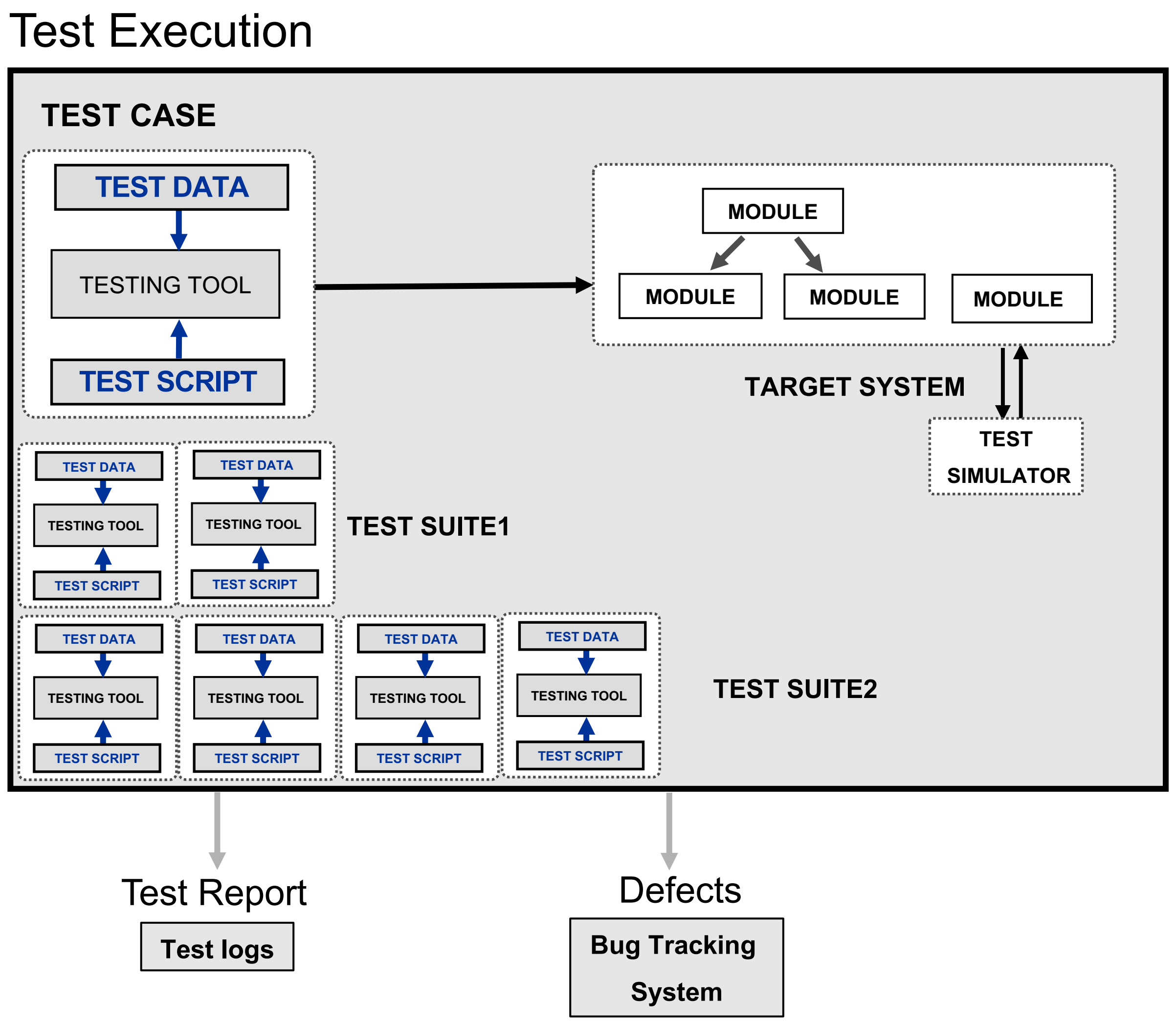
Note: Same test cases are run multiple times throughout the project

Test Procedure

- Documents explaining test execution flow
  - How to run the tests
  - Which tests to run
  - Test scripts usage
- Expected results for test cases
- Execution
  - Define severity and priority
  - Scripts, test suites

Report and Track

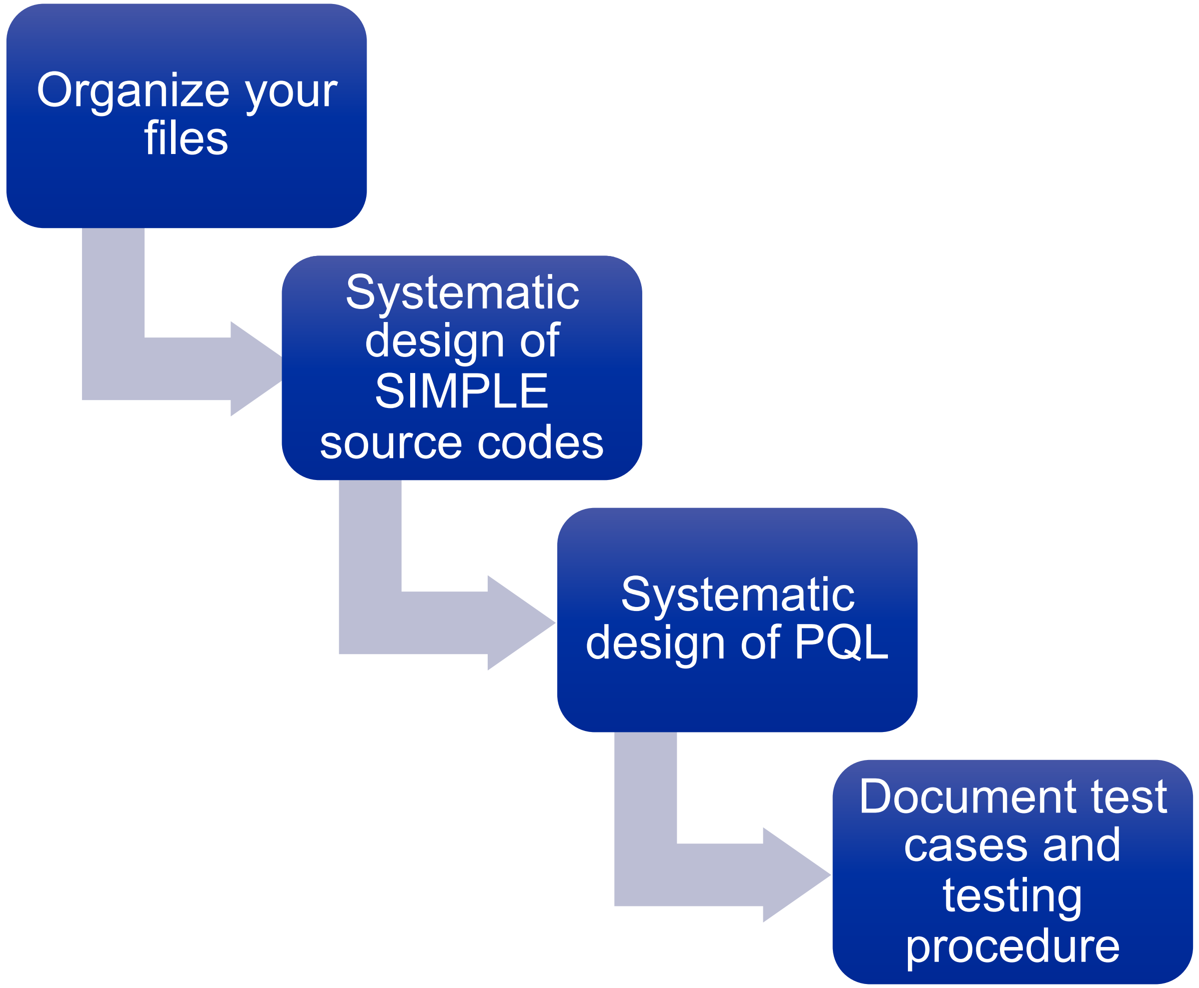
- Report
  - Logging expected and actual result
  - Current status
  - Time & resource usage
- Tracking
  - Bug in the test cases vs. bug in the system
  - Use bug tracking system (issue tracker)
  - Define tracking workflow
    - » Assign the bug to a developer
  - Use regression testing after issues have been fixed



Performance Tuning

- Use tools to profile your code:
  - Visual Studio Enterprise profiling tools
  - Run as Administrator
- Optimize sections where execution spends more time
- Solve the bottleneck and observe effects (before / after)
- In conjunction with regression testing

Some Tips for SPA System Testing


Organize your files
Systematic design of SIMPLE source codes
Systematic design of PQL
Document test cases and testing procedure

Organize your files

- Multiple files with source codes and queries
- Use meaningful names
- All files in the same folder?
  - Easier to run your testing
- Too many files?
  - Use a document to explain the files usage (test procedure)

Examples

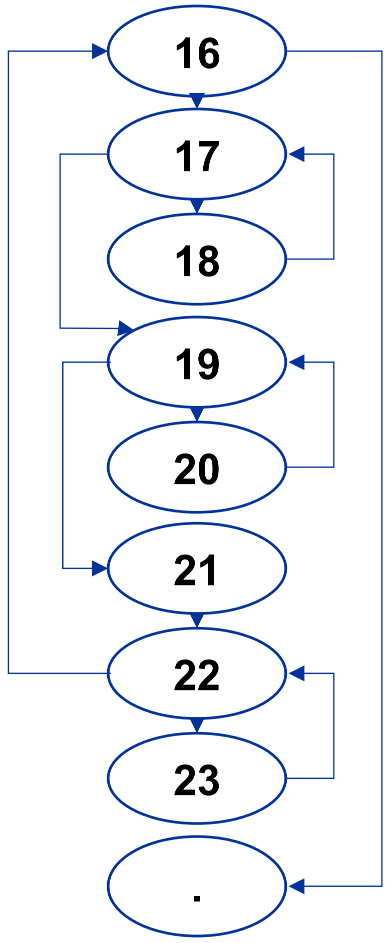
Folder 1

- modifies.txt
- pattern.txt
- pattern+modify.txt
- simple.txt
- uses.txt
- uses+pattern.txt

Folder 2

- query1.txt
- query2.txt
- query3.txt
- query4.txt
- query5.txt
- query6.txt
- query7.txt
- readme.txt
- simple.txt

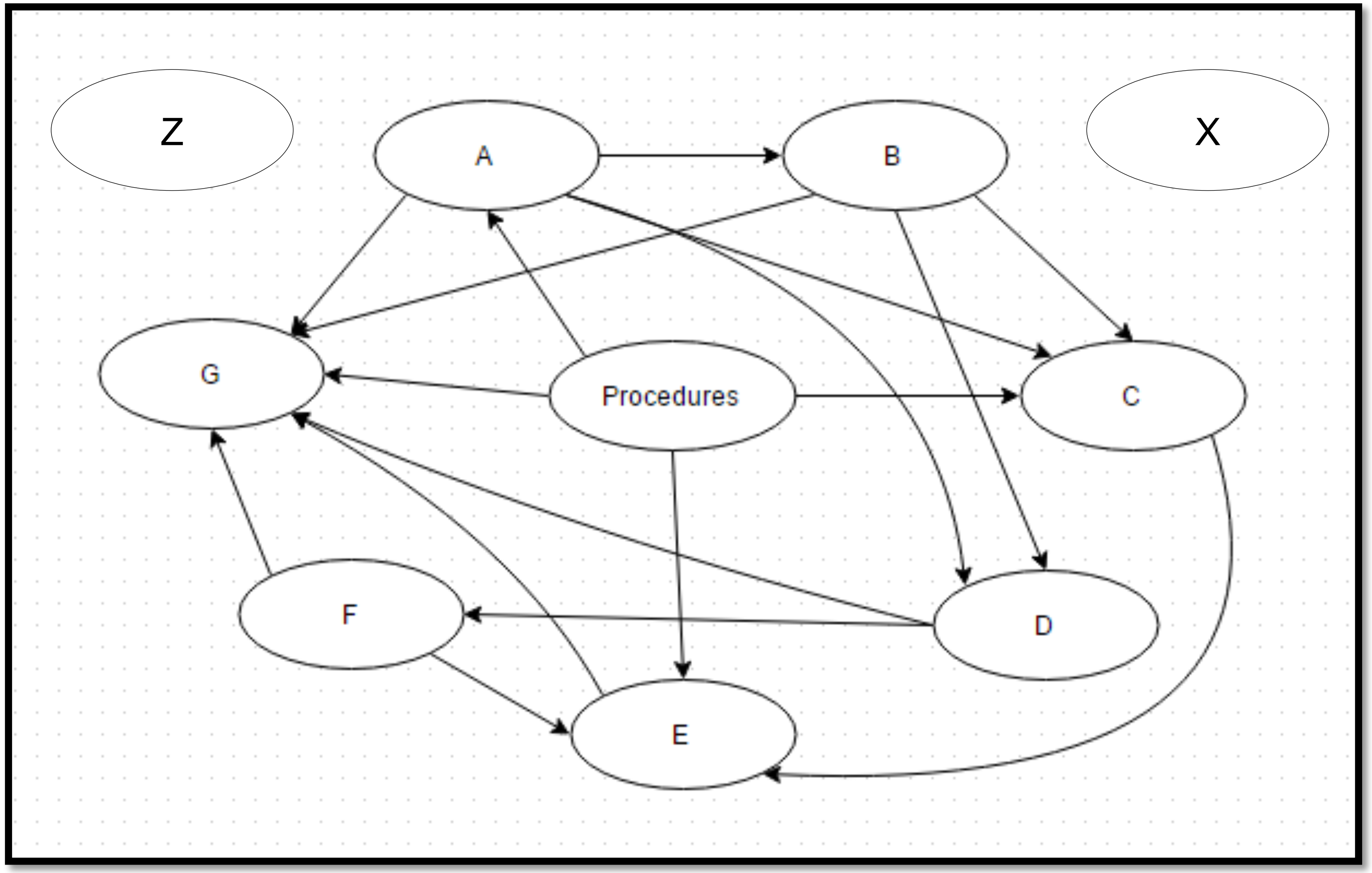
Using Graphs



```
procedure Second{
16.   while (w > 0) {
17.     while (w1 > 0) {
18.       c = b + a; }
19.     while (w2 > 0) {
20.       b = a + c; }
21.     a = a + b + c;
22.     while (w3 > 0) {
23.       c = 9; }}}
```

** A graph concept of CFG will be introduced in AdvSPA

Using Call Graph



** applicable in Iteration-2 and iteration-3

Create Queries by Type

Types of Queries	Uses, Modifies, Parent, Parent*, Follows, Follows*, Next, Next*, Affects, Affects*, pattern, "with" clauses
	such that+and | with+and | such that+with | with+such that
	assign, stmt, while, if, procedure…
	tuples
	invalid queries
	stmtRef: stmt, if ,while, assign, 'DIGIT+', '_' entRef: variable, '_', 'NAME'

Generate Variations of Queries

Testing different variations of query but same/similar results:

- stmt s, s1;
  Select s such that Follows(s, s1) and Parent(s, s1)
- stmt s, s1;
  Select s such that Follows(s, s1) such that Parent(s, s1)
- stmt s, s1;
  Select s such that Parent (s, s1) and Follows (s, s1)
- stmt s, s1;
  Select s such that Parent (s, s1) such that Follows (s, s1)
- stmt s, s1;
  Select <s, s1> such that Follows(s, s1) such that Parent(s, s1)

** applicable in Iteration-2 and iteration-3

Notes on Testing in Project Evaluation

- Final testing (Iteration 3)
  - Main SIMPLE source file will be about 500 lines
  - About 500 (mostly valid) queries

- Test your parser with *prototype_sample_SIMPLE_source.txt*
  - Available in the startup solution / repo under TeamXX/CodeXX/tests