****

**CS3203 Software Engineering Project**

**AY20/21 Semester 2**

**Iteration \_\_ Project Report**

**Team \_\_**

|  |  |  |
| --- | --- | --- |
| **Team Members** | **Matriculation No.** | **Email** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Consultation Hours:** <Insert day and time here>

**Tutor(s):** <Insert name here>

Table of Contents

[1 Scope 3](#_Toc61188266)

[2 Development Plan 4](#_Toc61188267)

[2.1 Project Plan 4](#_Toc61188268)

[2.2 Iteration \_\_ Plan 5](#_Toc61188269)

[3 SPA Design 6](#_Toc61188270)

[3.1 Source Processor 7](#_Toc61188271)

[3.2 PKB 7](#_Toc61188272)

[3.3 Query Processor 7](#_Toc61188273)

[4 Testing 8](#_Toc61188274)

[4.1 Unit Testing 8](#_Toc61188275)

[4.2 Integration Testing 8](#_Toc61188276)

[4.3 System Testing 9](#_Toc61188277)

[4.4 Any other testing 9](#_Toc61188278)

[5 Extensions to SPA 10](#_Toc61188279)

[6 Coding & Documentation Standards 11](#_Toc61188280)

[6.1 Coding Standards 11](#_Toc61188281)

[6.2 Documentation Standards 11](#_Toc61188282)

[7 Discussion 12](#_Toc61188283)

[8 Appendix 13](#_Toc61188284)

[8.1 PKB Abstract API 13](#_Toc61188285)

[8.2 API Discovery Process 14](#_Toc61188286)

[8.3 Sample Test Cases 16](#_Toc61188287)

[8.4 Any other appendices 16](#_Toc61188288)

# 1 Scope

Describe and list the features you have implemented for SPA.

Describe any bonus features you have implemented for SPA (e.g. Implementing full BasicSPA requirements instead of Iteration 1 SPA requirements).

Suggested number of pages for Section 1: 1 to 2 pages.

# 2 Development Plan

**[Iteration 1]** Describe your plan for the whole project, spanning across Iteration 1.

**[Iteration 2/3]** Describe your plan for the whole project, spanning across Iteration 1 to 3.

Consider using landscape pages instead of portrait pages.

You can include Gantt charts etc. to substitute the tables, if it expresses your plan better.

Suggested number of pages for Section 2: 5 to 10 pages. Keep your tables concise.

## 2.1 Project Plan

**[Iteration 1]** Plan your task assignment during the project, which should be covered for Iteration 1. Consider merging with subsection 2.2.

**[Iteration 2/3]** Plan your task assignment during the project, which should be covered for all iterations.

Tasks correspond to considerably large work units, such as “Design Program Parser for subset of SIMPLE language”, “Design PKB API”.

Indicate when each task is carried out by whom, and when it is carried out.

Explain the rationale behind your plan**.**

## 2.2 Iteration \_\_ Plan

**[Iteration 1/2]** Plan your activity assignment during Iteration 1 & 2. Separate them into two subsections.

**[Iteration 3]** Plan your activity assignment during Iteration 3.

Activity corresponds to smaller work units, such as “Testing interface to AST”. Tasks consists of activities.

Indicate when each task is carried out by whom, and when it is carried out.

Explain the rationale behind your plan**.**

# 3 SPA Design

In this main section, give an overview of your main SPA components and the way they interact.

Provide an architecture diagram and explain briefly how the component work with each other.

Do not repeat information from the course materials. Use UML diagrams when needed.

Design patterns provide standardized solutions to design problems. By applying a design pattern, you usually win more flexibility, but an overall program solution may be more complex to work with. Evaluate the trade-offs involved in terms of expected benefits and cost of applying a design pattern. If you have applied design patterns in implementing SPA, document in the main section by:

1. Explaining the design problem and pattern you applied
2. Document expected benefits and cost of applying a design pattern
3. Document the actual benefits and cost of a design pattern that you experienced in the project after applying it

After which, you should have subsections detailing the design of every component / subcomponent, and how they interact with each other. Feel free to add / edit the subsections. Do not go more than 3 nesting levels in your report. Do not show code snippets or pseudocode for explanation.

At the end of each subsection, document important design decisions. Follow guidelines in lecture notes and in-class activities to analyse and justify. Pay attention to clarity of the description.

You should discuss design decisions related to:

1. Architectural changes
2. Parsing and validation
3. Data representations for design abstractions in PKB
4. **[Iteration 2/3]** Solutions to speed up information access in PKB
5. Query evaluation
6. **[Iteration 2/3]** Optimisation strategies
7. Any other issue that is important

Suggested number of pages for Section 3: 60 to 80 pages. If you go over 80 pages, you are most likely not organising your information in the report well enough.

## 3.1 Source Processor

Your Source Processor should consist of Program Parser and/or Design Extractor. Consider separating the subcomponents into subsections.

Describe how you parse and validate SIMPLE source code.

Also describe when and how you extract information and store the information into the PKB.

You should also show the interactions between the Source Processor and PKB using examples, UML sequence diagrams, and/or UML activity diagrams.

## 3.2 PKB

Describe your PKB structure, and when / how it is used.

Consider using examples.

## 3.3 Query Processor

Your Query Processor should consist of Query Preprocessor (Parser + Validator) and Query Evaluator.

Consider separating these subcomponents into subsections.

For Query Preprocessor, describe how you parse and validate PQL queries. You should describe the validation rules used. DO NOT provide procedural description (pseudo-code) on how validation is done.

Also describe the data representation for program queries (Query Object).

For Query Evaluator, describe your evaluation strategies and optimisations. Show examples of how evaluation is done and its intermediate / final results table.

You should also show the interactions between the Query Processor and PKB using examples, UML sequence diagrams, and/or UML activity diagrams.

# 4 Testing

Provide your test plan, which includes the schedule and rationale.

Your test plan should explain the testing activities you have conducted during Iteration 1/2 or Iteration 3. Describe exactly when the testing activities were conducted. Ensure that the test plan corresponds to your development plan.

Also explain your testing strategy, such as testing scripts or bug tracking methodology.

In the following subsections, you will be asked to provide sample test cases.

Document each test case as follows:

1. Test Purpose: Explain what you intend to test in this test case
2. Required Test Input(s): Explain what component (or the whole system) you test, and what input must be fed to this test case
3. Expected Test Result(s): Specify the results to be produced when you run this test case
4. Any other requirements for running this test case (e.g. to run a test case for a component, you may need to implement a simulated environment)

Suggested number of pages: 10 - 25 pages, focusing more on System Testing subsection.

## 4.1 Unit Testing

Explain how you have designed your unit test cases.

Provide two sample unit test cases for PKB and two sample unit test cases for PQL processor.

Consider placing these sample test cases in the appendix if the section gets too long.

## 4.2 Integration Testing

**[Iteration 2/3]**

Explain how you have designed your integration test cases.

Show communication among SPA components using UML sequence diagrams.

Use sequence diagrams to plan integration testing and to indicate which component integrations you have tested.

Provide two sample integration test cases.

Consider placing these sample test cases in the appendix if the section gets too long.

## 4.3 System Testing

Explain how you have designed your system test cases. Pay equal attention to SIMPLE source code and PQL queries used for system test cases.

Provide two sample test cases in AutoTester format.

Consider placing these sample test cases in the appendix if the section gets too long.

## 4.4 Any other testing

Include subsections for regression testing, stress testing etc. Rename the subsections appropriately.

# 5 Extensions to SPA

**[Iteration 2]**

Describe your proposed extension. It should contain:

1. Definition of the extension
2. Overview of the possible implementation challenges
3. How they affect the SPA components
4. Proposed schedule for implementation over a 3-week timeframe
5. Test plan

**[Iteration 3]**

Describe how you implement NextBip/\* and AffectsBip/\* . It should contain:

1. Implementation changes to each SPA component
2. Schedule for implementation in Iteration 2/3
3. Test plan

Basically, this section is a mini-report for extensions.

Use subsection headers if necessary. Do not go over 3 nesting levels.

Suggested number of pages: 5 – 15 pages

# 6 Coding & Documentation Standards

This section documents the coding standards used in the project, and documentation standards for API design.

Suggested number of pages: 3 - 5 pages

## 6.1 Coding Standards

Briefly state or provide a reference to what kind of coding standards are adopted by your team. Comment on any changes made to the coding standards.

## 6.2 Documentation Standards

Comment on the correspondence of the abstract API with the relevant C++ classes.

Consider these questions:

1. Do you use same naming conventions in abstract API and your C++ program?
2. Do you use typedef to map abstract API type names to C++ types?

Comment on any experiences, problems, and benefits, that you observed working from abstract API towards C++ program.

# 7 Discussion

Discuss problems encountered that affect project schedule.

What worked fine for you? What was a problem?

In which areas do you plan to improve in the next iteration or project?

What management lessons have you learned?

Discuss any other project experiences and issues.

Suggested number of pages: 1 – 2 pages.

# 8 Appendix

## 8.1 PKB Abstract API

**[Compulsory Appendix]**

Write APIs for the design abstractions listed below.

1. VarTable
2. ProcTable
3. AST (if any)
4. Follows, Follows\*
5. Parent, Parent\*
6. Modifies for assignment statements
7. Uses for assignment statements
8. **[Iteration 2/3]** Calls, Calls\*
9. **[Iteration 2/3]** Next
10. **[Iteration 2/3]** Next\*, Affects, Affects\* (if any, bear in mind that they are to be computed in query time)

Note that APIs for Follows/\*, Parent/\* can be defined separately from AST or as part of APIs involving AST.

Please follow the examples from course materials to document your API.

Remember to use abstract types and describe normal and abnormal behaviours.

Consider using subsections.

## 8.2 API Discovery Process

**[Good-to-have Appendix]**

To help you discover your APIs for PKB and Query Evaluator, start by answering the numbered questions. You can answer ONE question in this appendix to show the thought process in coming up with the APIs. You do not need to answer more than one question in this appendix, but they can still be used in discovering your APIs.

Using subsections, describe how SPA components work with design abstractions (possibly via APIs), following examples in course materials using English.

Feel free to shorten/abbreviate repeated steps. For each interaction, continue description until you feel you are ready to switch to document a given API. Include as many steps as you think it is enough for you to get useful feedback.

Base your answers on the following SIMPLE source program and queries.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| procedure main {  1. read x;  2. read y;  3. while (y != 0) {  4. x = x / y;  5. read y; }  6. print x; } | assign a; stmt s; variable v;   |  |  | | --- | --- | | 1 | Select a pattern a(“x”, \_”y”\_) | | 2 | Select s such that Follows (1, s) | | 3 | Select s such that Follows (s, 3) | | 4 | Select s such that Follows\* (1, s) | | 5 | Select s such that Follows\* (s, 3) | | 6 | Select s such that Parent (3, s) | | 7 | Select s such that Parent (s, 5) | | 8 | Select s such that Parent\* (3, s) | | 9 | Select s such that Parent\* (s, 5) | | 10 | Select v such that Modifies (1, v) | | 11 | Select a such that Modifies (a, “x”) | |

Questions:

1. Write a sequence of steps describing how Parser works with ProcTable & VarTable.  
   For example,
   1. Insert “main” to ProcTable, return index
   2. Insert “x” to VarTable, return index
2. Write a sequence of steps describing how Parser builds an AST when parsing procedure Main.
3. Describe how Parser works with Follows and Parent  
   For example,
   1. Parser must set Follows (1, 2)
4. Describe how Query Processor works with AST and VarTable when evaluating Query #1  
   *Hint: Query Processor must traverse the AST in depth-first order and find subtrees that match pattern.*  
   For example, you could start off as follows:
   1. Get root of procedure main, return p
   2. Get first child of p, return p
   3. isMatch(p,”assign”) – no
   4. etc.
5. Describe how Query Processor works with Follows/\* when evaluating Query #2, #3, #4, and #5.
6. Describe how Query Processor works with Parent/\* when evaluating Query #6, #7, #8, and #9.
7. Describe how Program Parser works with Modifies when parsing statements 1 and 4.
8. Describe how Query Processor works with Modifies when evaluating Query #10, and #11.

## 8.3 Sample Test Cases

**[Compulsory Appendix]**

See Section 4: Testing.

## 8.4 Any other appendices

For example, you may include Iteration 1/2 development plan and test plan in the appendices for Iteration 3 report.