Cover page

Content page

Abstract

1. Development Plan

*Plan the tasks and activities for each team member. Activity is a smaller work unit than task. For example, you may have an activity such as testing the interface to AST. Tasks consist of activities. Define activities of a size that you feel comfortable planning with.*

1. Scope
2. The basic requirements of SPA in iteration 1 include parsing a single procedure, parsing queries, and responding to these queries. Our SPA is able to parse a program line-by-line, and populate the tables implemented to respond to the PQL with one run. The implemented query evaluator is able to use the API provided by the PKB to respond to the PQL's queries. Our SPA can answer Follows, Parent, Modifies and Uses queries, and recognise patterns in the input program.
3. SPA Design
   1. Overview

Our SPA components include the program parser, PKB, query parser, and the query evaluator. The parser dissects the program lines based on its type (while, if, etc.), and calls PKB setter functions accordingly. The PKB is implemented using tables, and stores instead of the program in AST form, answers to possible queries from the PQL. A substantial amount of query processing is done in the PKB, and the query evaluator calls for pre-computed results, and outputs values according to its input query; the query evaluator breaks the query into multiple parts and queries the PKB, putting together sets of responses and finally output the correct set of result. The PQL aids the query evaluator with this process, parsing the query as clauses to be received by the evaluator, and later identified to make appropriate calls to the PKB and output the final result.Design of the SPA Components

* + 1. SPA Front-end Design

*Describe how you parse and validate SIMPLE source code. Explain*

*when and how your PKB is populated by Parser / Design Extractor. b. PKB design ‐ Explain the structure of your PKB, as well as when and how*

* + 1. PKB Design

The PKB is designed after a black box, where insertion of values comes only from the parsed in code, and results are obtained by the query evaluator with only knowledge of the API.

The PKB is first accessed by the parser. The parser calls the PKB’s set functions and stores all variations of Follows, Parent, Modifies and Uses in its various tables. The PKB contains a varIndexTable and a procIndex Table (which is private to its class), which gets the index of any variable in the program, or if it does not have one, creates and returns it. It also has tables that identify statement types – if statement 8 is a while statement, and is able to return all variables, constants and statement numbers in the program.

Our PKB was designed to reduce setter calls from parser, for example: in the setModifies method, for setModifies(3, “z”), in addition to adding statement 3 modifies variable z in our ModifiesTable, we also call the function getParent(3), and setModifies((iterate through all parents), “z”);. For functions such as modifies, uses, etc., the modifiedBy, usedBy functions and implicitly called within the functions itself, freeing the parser from these calls.

The PKB provides conversion from variable or procedure names, to its index values, runs its methods, and returns its result in the form required by SPA. This removes components interacting with the PKB to know of the indexing table; the components either parse in the statements or obtain the results as required.

* + 1. Query Processing

Explain your query validation. A sample query validation rule is: "Check if all relationships have the correct number and types of arguments". DO NOT provide procedural description (pseudocode) of how Query Pre‐processor checks the rules. Very briefly, describe how you encode validation rules, and how you perform query validation. Describe your design and implementation of Query Evaluator ‐ describe the data representation for program queries (query tree) and how your Query Evaluator works.

* 1. Component Interactions

*Include any UML diagram that you have found useful. For each diagram, explain how you have used it (e.g., in project planning, communication, test planning or in other situations), and comment on the value it has added to your project.*

1. Documentation and Coding Standards

As version control is simple and easy to use for a large team, documentation was done using the same version control system we used for our project – Git. API was constructed by the members of our team that worked on the PKB, and commenting was done in the PKB header file to aid the components interacting with it, namely the parser and query evaluator. We’ve adopted the conventional naming standard, camelCase, and for parameters such as (int statement), we’ve chosen to name the statement s if there is only one statement in the function, and s1, s2, … for more than one statements, etc.

1. Testing

their concrete API counterparts (C++ classes).

5. Testing

Describe your approach to testing. Comment on the testing experience gained from this project. Be sure that you understand the role of the given AutoTester, integrate it with your SPA, and use it to automate regression testing throughout the project.

* 1. Test Plan

*Your test plan should explain the activities you have conducted during all phases of testing and describe when (or how often) these activities were conducted throughout this iteration. Focus more on System testing and explain how you have designed your testcases to cover the SPA requirements. Also, explain about the process used to run these testcases (regression testing, any scripts, bug tracking, etc).*

* 1. Examples of Test Cases
     1. Unit Testing
     2. System (Validation) Testing

1. Discussion

With a five-man team, the largest difficulty we faced was being held responsible for a certain component in the SPA. With a program parser, PKB, program evaluator and PQL parser, each one of us had to handle one component alone, with the exception of the PKB, and understand the components interacting with it to make the appropriate API and function calls.

When we first designed our SPA implementation, we decided to store the entire program as an AST (linked list) and using traversal, obtain results for the query evaluator. We soon realised that storing the program in well-defined tables is much more advantageous in terms of efficiency with regards to results-calling - we can have pre-computed values as possible queries to the PKB can be easily created by us. This was a design decision, and one we stand by with the implementation of SPA.

The scale of this project slightly underestimated by us, and only as we started to implement the program did we realised the complexity and scale of the program. I do believe with more in-depth discussions on the implementation and with larger dissection of tasks (with shorter deadlines), we could have found the implementation of SPA to be less overwhelming. Better management with regards to watching over and enforcing the completion of this smaller tasks would have greatly helped in this iteration.

1. *Documentation of Abstract APIs*
   1. VarTable
   2. ProcTable
   3. Follows, Follows\*
   4. Parent, Parent\*
   5. Modifies, …

*Comments:*

*􀁸 APIs for Follows, Follows\*, Parent, Parent\* can be defined separately from AST, or as part of AST API.*

*􀁸 Please follow the examples from lecture notes or from Handbook (Section 9.6) to document your API.*

*􀁸 Appendix might contain the answers for Section 3 of this document (Discover Abstract APIs for design abstractions).*