CS3215 (2011) Assignment 4: Development iteration 2

Due date: Monday, March 21, by 12 noon, to your supervisor

Deliverables: Integrate descriptions produced by team members and submit one coherent document per team, organized in the format given in the Required Format for Assignments and Final Project Report. You will also present your solution during the consultation hour immediately after the assignment due.

1. Describe project plans

Describe updated plan for the whole project and for this development iteration. Follow formats from Assignment 3.

2. SPA implementation

Feel free to extend the suggested scope of implementation (without compromising the quality!).
If you want to change the scope of implementation (narrow in certain areas and extend in other areas), you can do so but please discuss your plan with supervisor.

The following is suggested scope of implementation for this iteration:

Implement complete SIMPLE according to the specifications.

Query sub-system to be implemented in this iteration:

a) queries should involve the following relationships:
   Follows, Follows*, Parent, Parent*
   Modifies, Uses
   Calls, Calls*
   Next, Next*

Remark: Your solution must be scalable, therefore pre-computing transitive closure of relationships (Next*, Parent*, Follows*, Calls*) and storing the result is not acceptable. Design efficient algorithms to compute Affects and Affects* on demand, during query evaluation.

b) Query Pre-processor should validate queries and build a query tree.
c) Implement **Basic Query Evaluator** (see below), and start planning your query optimization strategy.
d) Be sure that you allow all valid types of arguments in relationships (**such that** clause) and under with clause, as specified in PQL. In relationships, arguments can be synonyms, '_' and, depending on the relationship, integer (statement line numbers or program line numbers) or string (variable or procedure names). Under **with** you should be able to compare an attribute value and constant (integer or string, depending on the type of attribute) or two attribute values (provided they are of the same type).
e) Allow all kinds of query result specifications, including tuples.
f) Queries can involve **and** operators and a mixture of different relationships, for example:
   Select s **such that** Next* (n1,n2) **and** Follow* (n1, s) **and** Parent (s, w) **and** Modifies (n2, "v")
g) Queries involving multiple **and** operators in **with** and **such that** clauses, and mixture of **with** and **such that** clauses, such as
   Select s **such that** … **and** … **and** … **with** **… and … such that … and … with … and … and …**

Hints for Query Evaluator:

First design the best possible Basic Query Evaluator (BQE) that can correctly evaluate any query without optimizations. Query involving many relationships must be evaluated incrementally, in steps. In the design of BQE pay attention to computing and representing intermediate query results. BQE should be prepared to employ various optimization strategies such as re-arranging the order in which to evaluate relationships in a query and many others. You will have to experiment with various optimization strategies that's why it is important that different optimization strategies can be plugged-into your BQE. BQE should not depend on any specific optimization strategy, but easily work with any optimization strategy you can think of.

Hints for implementation (excerpt from Handbook Section 9):

Use abstract PKB API to guide your implementation. Maintain full traceability between abstract PKB API and relevant classes in your program.
Follow these two guidelines:

1. Do not retrofit implementation details into abstract PKB API. Abstract PKB API will fail to play its role if you did that.
2. Use the same naming conventions in abstract PKB API and corresponding implementation classes (the same names in abstract operations/arguments and member functions). Use the same symbolic names for data types in abstract PKB API and corresponding implementation classes. 'typedef' can easily assign specific C++ data type to symbolic names.

3. Update (if necessary) documentation of abstract PKB API

Include complete, updated documentation of abstract PKB API in the assignment report.

4. UML diagrams

Draw UML diagrams that you found useful. For each diagram that you draw, explain how you used it (e.g., in project planning, communication, test planning or in other situations), and comment on the value a diagram added to your project.

Hint: Read through relevant parts of Handbook Section 10. Refine UML sequence diagrams given in examples into more detailed sequence diagrams showing communication between SPA functional components and specific data abstractions in PKB.

5. Documentation of important detailed design decisions

Follow guidelines in Handbook Section 10.2 to properly analyze, justify and document detailed design decisions. Pay attention to clarity of the description (check hints in Section 10.2).

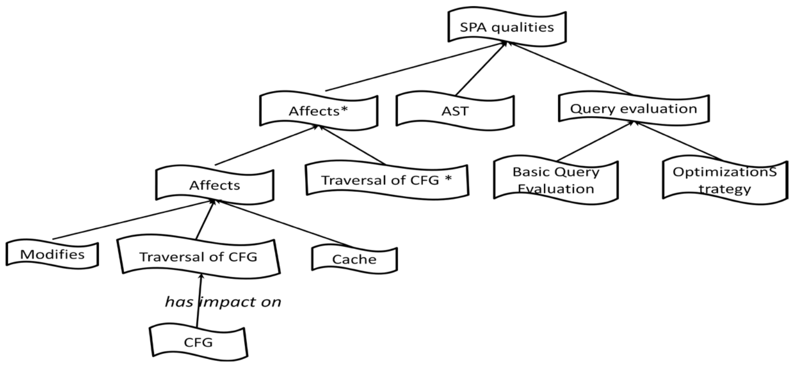
Address detailed design decisions related to data representations for design abstractions in PKB, any other solutions to speed up access to information stored in PKB, algorithms to fetch data from PKB during query evaluation, and any other issues that you consider important, except query evaluation (which is addressed in Section 7.2).

5.1. Documentation of design decisions

While you should document all the design decisions that you consider important, it is obligatory that you document your choice of design solution for relationship Modifies, AST, CFG and computation of Next*.

Documentation of design decisions should consist of:

1) Diagram depicting inter-dependencies among design decisions. Expand example given below:



2) For each design problem:
   a) State design goals relevant to this
   b) Consider alternative design solutions
   c) Evaluate design solutions in view of design goals. Use Big O notation to describe complexity of algorithms
   d) Justify your choice of design solution
   e) Document the above process

5.2. Documentation of design patterns

Design patterns provide standardized solutions to design problems. You studied some typical design problems for which published design patterns exist in CS2103. By applying a design pattern, you usually win more flexibility, but an overall program solution may be more complex to work with. Always evaluate carefully the trade-offs involved in terms of expected benefits and the cost of applying a design pattern. Apply a design pattern only if the benefits outweigh the cost.

If you applied design patterns, document them in this section:

1) Explain the design problem and pattern you applied to solve it
2) Document expected benefits and costs of applying a design pattern
3) Document the actual benefits and costs of a design pattern that you experienced in the project after applying it.

6. Coding standards and experiences

1) State coding standards adopted by your team.
2) Comment on how you used abstract PKB API to guide design of relevant C++ classes and how you keep abstract and concrete PKB API in sync with other.
   a) Do you use same naming conventions in abstract PKB API and your C++ program?
   b) Do you use typdef to map type names used in abstract PKB API to C++ types?
3) Comment on any experiences - problems and benefits - that you observed working from abstract PKB API towards C++ program.

7. Query processing

7.1. Validating program queries

Describe query validation rules, only in case there is some difference as compared to what you described in your previous assignment. An example of query validation rule is: "checking if all relationships have correct number and types of arguments, as defined in PQL definition in Handbook". DO NOT provide procedural description (pseudocode) of how Query Pre-processor checks the rules.
If you use table-driven approach to query validation – show the structure of your tables.

7.2. Design and implementation of query evaluator

1) Describe data representation for program queries
2) Describe your strategy for Basic Query Evaluation (BQE)
3) Describe optimizations
4) Document detailed design decisions regarding BQE and optimizations, in the same style as you documented design decisions in the above point 5.2.

Hint: Follow guidelines in Handbook Section 10.2 to properly analyze, justify and document detailed design decisions regarding BQE and optimizations (if any). Pay attention to clarity of the description (check hints in Section 10.2). Do not repeat what you already discussed in Section 5, but refer to relevant points.

8. Testing: Group-PKB and Group-PQL

8.1. Describe your test plan for this iteration

8.2. Provide examples of test cases of different categories

1. unit testing:
   a) provide FIVE samples of specific unit test cases for PKB and FIVE test cases for PQL
   b) if you used assertions, describe how and show examples
2. integration testing
   a) UML sequence diagrams show communication among SPA components. Use sequence diagrams to plan integration testing and to indicate which component integrations you have tested.
   b) provide FIVE sample integration test cases
3. validation testing
   a) provide FIVE sample test cases

Document each test case in standard way as follows:

Test Purpose: explain what you intend to test in this test case
Required Test Inputs: explain what program module (or the whole system) you test and what input must be fed to this test case
Expected Test Results: specify the results to be produced when you run this test case
Any Other Requirements: describe any other requirements for running this test case; for example, to run a test case for a program module in isolation from the rest of the system, you may need to implement a simulated environment to call the test case.

9. Discussion

Discuss problems encountered that affected project schedule.

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

Discuss any other project experiences and issues.

--- The End ---