

CS3202 Software Engineering Project

Revision of the Prototype



Team Number: 4 Consultation Day/Hour: Tuesday, 1pm

Team Name: Team 4

Team Members Information:

Group PKB

Saloni Kaur A0084053L [a0084053@nus.edu.sg](mailto:a0084053@nus.edu.sg)

M I Azima A0085594N [a0085594@nus.edu.sg](mailto:a0085594@nus.edu.sg)

Group PQL

Saima Mahmood A0084176Y [a0084176@nus.edu.sg](mailto:a0084176@nus.edu.sg)

Nguyen Trong Son A0088441 [A0088441@nus.edu.sg](mailto:A0088441@nus.edu.sg)

Vu Phuc Tho A0090585X [A0090585@nus.edu.sg](mailto:A0090585@nus.edu.sg)

August 25, 2014

# Introduction

In the submission of our project for CS3201, we had encountered a couple of issues. Amongst these issues, the main problem was that the query evaluation time was exceptionally long. In our revision for the prototype we have focused on looking further into this and revamped how we implemented this specific aspect of this project. In the following section we will be discussing the design decisions that we have decided to change and those that we have decided to keep.

# Design Decisions

Keeping in mind the design decisions that we had decided on in CS3201, we have decided to focus our changes to the Query Evaluator (QE). The things that we have decided not to change, include the implementation of the PKB and the decision to not use the AST for “such that” query evaluation.

Furthermore, we have decided to keep the Query Representator (QR), to store information from the query preprocessing, for the QE to use. As we go into the project specifications for CS3202, the queries will become more complex. In light of this, having a QR to store all of this information, much like the PKB, will make the query evaluation more efficient.

For the purposes of this revision, our main problem was in the QE and as such we decided to focus on this component.

# Changes

We found a logical error in the QE that we have fixed. The problem was that we had missed to capture a ‘if’ case for the pattern clause of the query. This was a simple fix that was implemented immediately once it was found out.

The main issue that we narrowed down to, was found out to be in the QE component of the prototype. In the design of the prototype for the previous semester, the QE was implemented to evaluate queries after the query file has been preprocessed by the Query Preprocessor (QPP) and relevant information stored into the QR. The QE would use information from the QR and the PKB to come up with each and every possible combination based on the information provided by the query. For example given a query:

Stmt s1;

Select s1 such that Follows (s1, 4)

First we would look at the PKB to find all of the values for s1. Then for each value, we would check if it satisfies the given relationship. In this manner we implemented a brute force solution for query evaluation for every type of a “such that” query, for various relationships.

In the revised implementation for the QE, for each relationship, we have considered every type of possible combination in the query. For instance, for the “Follows” relationship, based on the specification of the prototype for CS3201, there are 4 possible ways in which the query could be asked; Follows (1, 2), Follows (10, s), Follows (s, 10) and Follows (s1, s2). The first query is pretty straight forward to answer by calling out an isFollows() method. The 2nd and 3rd queries have now been implemented by calling out the corresponding method, such as getFollowingStmt() and getFollowedStmt() for the possible combinations of the query based on the Follows table. For the 4th type of query, there is no other efficient solution other than using the brute force method for the evaluation.

As such we have reduced the time taken for the evaluation of each relationship query, by ensuring that we do not use the brute force method when we have an alternative way. The problem with the brute force way of answering the query was that, since we generated all possible combinations of the result, we took more time to evaluate all of the combinations and also the pattern part of the query. As such this unnecessary complexity has been reduced in this revision.