Senior Design

Sociotechnical Solution to Large-Scale Formal Specification Mining

Design Document

Group May1620

Alex Dana, Deeksha Juneja, Cody Hanika
Advisor: Professor Rajan

Table of Contents

Table of Contents	Page 1
List of figures	Page 2
Overview	Page 3
Problem Statement/Background	Page 3
System Design	Page 3-4
Detail Description	Page 5-7
Conclusion	Page 8

Figures

Figure 1: This is a diagram outlining the basic structural design and communication flow for our project.

Overview

Java specifications are precise definitions of methods of Java libraries. They would benefit all programmers. However, formal specifications are hard to come by and often times difficult to produce. Formal specifications involve describing methods in mathematically precise language and verifying preconditions and postconditions. Hence, the purpose of this project is to create a socio-technical solution to this problem. The problem will be solved by using pre-generated specifications (which may or may not be complete or correct) which will be automatically posted to a Q&A site (for example - StackOverflow) asking for further analysis on the specification. Then the program designed by May1620 will be used to analyze the answers and comments on the question to generate accurate and complete specifications.

Problem Statement and Background

Formal specifications provide a generalized, high-level, and easily extendable approach to verification. Through the use of formal specifications, it is easier to write correct code and to avoid the use of an API in a manner that was not intended. Currently, popular Java libraries either lack publicly available formal specifications or it is difficult to obtain these specifications. Previous attempts at creating specifications at a large scale has failed, due to the difficulty of generating correct formal specifications without human intervention.

System Level Design

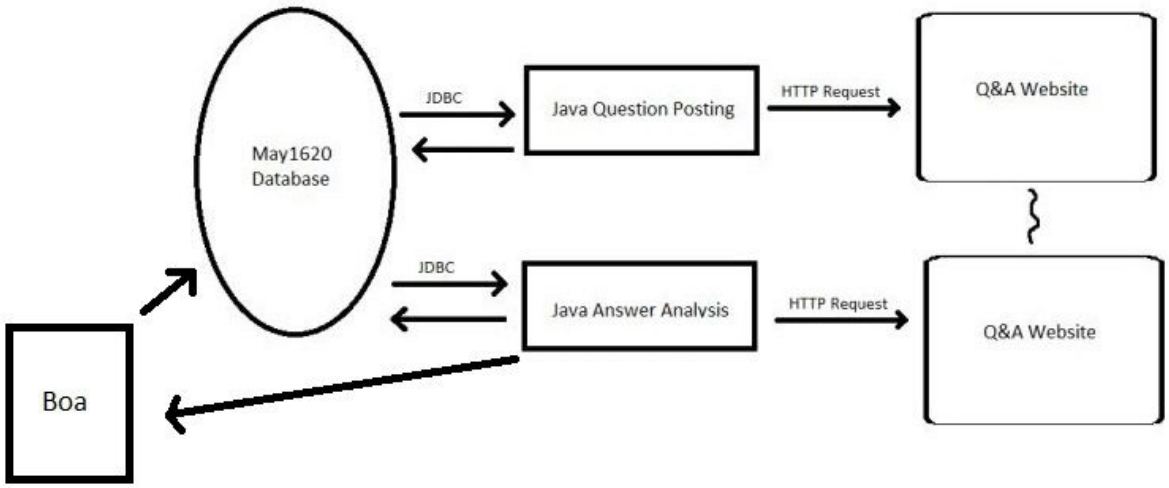
- The application must be able to communicate with the Q&A site via HTTP requests
- Specification questions must be posted to the Q&A site via HTTP requests
- Answers to the questions must also be retrieved with HTTP requests
- Specifications must be stored in a database to be retrieved and inserted
- Java and Stanford NLP will be used to analyze answers grabbed via the HTTP requests
- Specifications that we collect must be persisted into the Boa system

There are 3 main parts of the application.

1. Database communication - This is the part of the project where the communication with the database is setup (as well as communication with Boa).

2. HTTP communication - In this part, the link between the project and the website is established.
3. Java analysis - Finally, using the various functionalities of Stanford NLP, the responses are searched through to find meaningful specifications for a method.

Figure 1: This is a basic diagram showing the communication flow and design for the project



Detailed Description

Major Packages

analyze: The analyze package will contain all the classes related to the Stack Exchange answer analysis portion of our work.

common: The common package will contain classes that are used by many components of the system. For example, Specification is a class that will need to be used by all the components as the whole system will be dealing with a specification.

post: The post package will contain all the classes related to the posting side of the system. The Post class will contain the main method to run our class.

stackexchange: The stackexchange package will contain classes related to retrieving and posting data to Stack Exchange.

Each of these packages will have a corresponding test package that will contain JUnit tests for their functionality.

Database Tables

A MySQL database will be used to store all of the data.

Type

Columns	**Type**
TypeId	INT AI PK
Name	VARCHAR
Package	VARCHAR

Method

Columns	Type
MethodId	INT AI PK
TypeId	INT FK
Name	VARCHAR
Signature (e.g. *void add(E item)*)	VARCHAR
SequenceFile	VARCHAR
SequenceFilePosition	BIGINT

Question

Columns	Type
StackExchangeQuestionId	INT PK
MethodId	INT FK (Optional)
ClassId	INT FK (Optional)
LastModified	TIMESTAMP
Open	BIT

Boa Specification Storage Schema

```
message Specification {
        // Here I am storing the location of the method in the AST sequence file to allow for quick
        // lookup of the code associated with this spec. If that is not needed, we can just store
        // the fully qualified method signature
        required string sequence_file
        required int64 sequence_file_pos
        required bool is_exceptional
        repeated SpecExpression preconditions
        repeated SpecExpression postconditions
        repeated string assignable_fields
        repeated string exception_types
}
```

```
message SpecExpression {
	enum SpecExpressionType {
		QUANTIFIED // body is in subexpressions
		VARACCESS
		ARR_INDEXING // index expression is in subexpression
		// For most of these, subexpressions will contain the left and right side of these
		// operations
		IMPLIES
		IFF
		LOGICAL_AND
		LOGICAL_OR
		PLUS
		MINUS
		DIVIDE
		MODULUS
		MULTIPLY
		BIT_LSHIFT
		BIT_RSHIFT
		BIT_UNSIGNEDRSHIFT
		BIT_AND
		BIT_OR
		BIT_NOT
		BIT_XOR
		EQUAL
		NOT_EQUAL
		LESS_THAN
		GREATER_THAN
		LESS_THAN_EQ
		GREATER_THAN_EQ
		NOT
		OLD // the value of a subexpression before running the method
		RESULT // the return value of the method
		LITERAL
	}
	enum Quantifier {
		FORALL
		EXISTS
		JML_SUM
		JML_PRODUCT
		JML_MAX
		JML_MIN
	}
	required SpecExpressionType type
```

```
        optional Quantifier quantifier
        optional string declared_vars // used for QUANTIFIED
        optional SpecExpression quantified_scope // used for QUANTIFIED
        repeated SpecExpression subexpressions
        optional string value // used for VARACCESS, ARR_INDEXING, and LITERAL
}
```

Major Classes

Post: This class will be responsible for posting a method to Stack Overflow to be specified. The high-level algorithm is this:

1) Query the Specification database table for a specification that is not yet finalized.
2) Create a Stack Overflow post title and body using this specification
3) Make an http request to Stack Exchange to post this question and store the question id returned
4) Store this question id in the OpenQuestion table

Analyze: This class will be responsible for analyzing the answers to open questions on Stack Overflow and inserting finalized specifications in the database.

1) Query the OpenQuestion table to get a list of open question ids.
2) For each open question, analyze the answers using stanford NLP. Retrieve the relevant parts of the answer and analyse the sentiments of the comments.
3) If an acceptable answer is found, accept the answer on Stack Overflow, update the specification in the Specification table as finalized, and remove the question from the OpenQuestion table.

Current Progress

We have achieved seeding our database with Boa information and posting questions on this through the application to StackOverflow. Now we must work on wording our questions in a way that best gets answers from the community. Furthermore, the Stanford NLP is being researched in order to analyze the answers that we receive on StackOverflow. Once this is done we will be able to extract formal specifications from the answers that are posted.

Conclusion

Overall this project will reduce the problem of specifications being unavailable for popular Java libraries. The accessibility of these specifications will hopefully promote proper usage of the libraries and functions within; which will result in better and safer code. Although attempts have been made in the past to generate specifications, we feel that this project will be more successful because of the social side to of it. It allows for a larger level of analysis that cannot be achieved with just a technical solution.