Progress Report - Week 3

[Graphical Tool for Refactoring Class Diagrams]

(Akanksha, Ashay, Gaurav, Prateek, Sree)

This week we focused on the following components:

1. Adding more refactorings.
    a. [COMPLETE] Change Method Signature.
    b. [COMPLETE]Move.
2. [COMPLETE] Generating prolog file using XML and position collection file.
3. [COMPLETE] Writing constraints for the diagram verification.
4. [In Progress – 80% Done] An attempt to link front end and backend. i.e. Merge refactoring code in JAVA with constraint verification code in Prolog.

Details:

1. Change Method Signature: - We are able to modify parameter types, return types, method name. In addition, we were successful in propagating the updates of the parent class to the child class.
2. Move: - Challenging part here is to list down all the classes on the diagram. We were able to figure out that part and now we can successfully move attributes and methods of one class to another.
3. Generating Prolog files (T2M): - We have written a parser, which generates prolog file from XML.
4. Constraints file: - A prolog constraints file which validates the XML. This takes care of doing M2M conversion and then checking constraints.
5. Link Front End and Back End: - We are able to:
    a. On each refactoring, we can now export our diagram from the java code itself.
    b. Invoke Swipl to run constraints.

    Both of these actions are encapsulated inside one class. Basically any refactoring that we create, would use that class to validate.

Pending Items:

1. Use results of swipl verification code and take appropriate actions with the JAVA's refactoring code. This action would vary for each refactoring.

    In our current approach, we update the memory model of Argo before check constraints. i.e. We actually do the refactoring and then check constraints (However, this change is not visible to the user). Challenge now stands is, how to revert back, if there is an error detected. We are exploring two approaches, use Argo's Undo infrastructure or create a copy of memory model before checking constraints.

2. Another challenge we face is to create new diagrams programmatically. i.e. If some refactoring requires creating another class, we should be able to do this from the Argo code itself.

Plan for next week:

Once we are able to figure out solutions to the challenges listed above, we would have 3 end-to-end working refactorings. Then we would target to add 2-3 more refactorings, which would again be done in parallel by groups members. Tentative ones are:

1. Extraction of a method from multiple classes into a new class.
2. Merging multiple classes into one.