11/28/2016

ReviewIT!

A Project Management Tool…

Aakash Sahu, Kartikeya Gupta, Manpreet Singh Gulati, Nishit Garg, Shubham Bansal

ME Software Systems

Department of Computer Science & Information Systems

BITS PILANI

Table of Contents

1. Introduction

Software industry has grown by leaps and bounds in the past decade or so and it is seeing a large number of projects being initiated. Such large number of projects has led to huge amount of code being generated which needs to be validated, verified and reviewed by experienced programmers and managed by project management staff to have a smoother development process and minimum mismatch between requirements and end product.

However, it has been observed that there are defects in the development process. Earlier the defects are detected, less the rework incurs. According to the findings from literature, most of the defects occurred during the design and coding phases. Thus there arises a need for proper verification and validation along with a review process by experts. Our tool, "ReviewIT!" aims at providing these capabilities.

2. Problem Statement

"ReviewIT!" will have a feature to examine Java source files against the object-oriented design described in UML class diagrams. Prior to the review process, the class diagrams would have to be converted into XML format so that the information of classes and relations could be extracted and used to generate the review checklists. A Designer will be able to upload the Design XML to the system. The XML code will then be analyzed by Anti Pattern Detector to generate a list of anti-patterns present in the design. An analysis of design will then be stored in the system. A Developer will be able to upload JAVA Code to the system, from where the code will be fed to a comparator which will compare both the code and Design XML and generate a checklist of comparison results. Both the Comparator and Anti Pattern Detector can be triggered by the Reviewer only. The code reviewer will then follow the checklist items to verify whether all defined classes exist in the code, the class structures with encapsulated methods and parameters are correctly implemented, all relations of associated classes are valid. Along with this a review functionality will be provided to get the code reviewed by the subject matter experts/experienced programmers that would be assigned by the project management.

Based on the feedback the developer may discuss with the expert on future course of action to be taken and the management will be effectively able to track all the changes done to code along with defects and comments that may have been logged.

3. System Features

- Examine Java source files against the object-oriented design described in UML class diagrams.
- A Designer can upload the XML of Class Diagram to the system.
- Analysis of XML to generate a list of anti-patterns present in design.
- A Developer can upload JAVA code to the system.
- Comparison of code and Design XML to generate a checklist of comparison results.

- Provision for a Reviewer to provide feedback on the list of anti-patterns and checklist of comparison results.
- Tracking of results and progress facility to Manager.

4. System Functions

System functions describe what a system is supposed to do. A system acts through system functions, which should be identified and listed in logical cohesive groupings. The category field in tables defines the degree of visibility to the user. Hidden means that the function is not visible by any actors, and evident indicates that the user or external interface can see the function actively or passively.

Ref#	Functions	Category
R1.1	Uploading of design to the system	Evident
R1.2	Feeding of design to the comparator	Hidden
R1.3	Uploading of JAVA code to the system	Evident
R1.4	Generation of anti-patterns from XML	Evident
R1.5	Comparison of JAVA code and design XML	Evident

5. Use Case

5.1 List of Actors

- Designer
- Developer
- Reviewer
- Manager
- Admin

5.2 Actor Goal List

ACTOR	GOALS
Designer	1. Submit XML of class diagram 2. View anti pattern checklist 3. View feedback
Developer	1. View XML 2. Submit Java code 3. View comparison checklist 4. View feedback
Reviewer	1. View Java code 2. View XML of class diagram 3. Detect anti-pattern 4. Compare Java code with XML 5. Submit feedback
Manager	1. Add team member 2. View feedback 3. Create project 4. Add project details 5. Submit feedback

5.3 Use Case: Upload XML

Id: UC-1

Description: The Designer uploads the XML description of the design (class diagram) created by him/her.

Primary Actor: Designer

Supporting Actors: None

Pre-Conditions: None

Post-Conditions:

Success end condition: XML description of the design gets uploaded in the system.

Failure end condition: The design is not available in the system

Trigger: The designer clicks on the ‘Submit XML’ button.

Main Success Scenario: The Designer reaches out to the System and enters his/her credentials. He/she then clicks on the 'Submit XML' button. On successful submission, the design gets updated in the system.

5.4 Use Case: View Results

Id: UC-2

Description: Any stakeholder of the project can view the results of the analysis, both anti patterns in design and comparison of code with design.

Primary Actor: Manager, Designer, Developer

Supporting Actors: None

Pre-Conditions: The design or the code and design both should be available in the system and the analysis should have been performed at least once.

Post-Conditions: None.

Trigger: The stakeholder clicks on 'View Analysis Results' available on the UI.

Main Success Scenario: Once the analysis has been performed, any stakeholder can click on 'View Results' to view the results obtained after running analysis. Depending on the type of analysis that has been performed, the user can view the anti-patterns present in the design or whether the design has been implemented properly.

5.5 Use Case: View Design XML

Id: UC-3

Description: The XML can be viewed by any stakeholder. It can be used by the stakeholders to see whether the XML has been correctly uploaded by the Designer. It also allows the Developer to obtain the XML of design as a reference for implementation.

Primary Actor: Manager, Developer, Designer

Supporting Actors: None

Pre-Conditions: The XML description of design should be available in the system.

Post-Conditions: None.

Trigger: The stakeholder of project clicks on the 'View Design XML' button.

Main Success Scenario: After the XML of design has been uploaded by the Designer or Developer, any stakeholder can view the design of the project.

5.6 Use Case: Submit Java code and XML

Id: UC-4

Description: The developer submits the Java code and XML of class diagram provided to him.

Primary Actor: Developer

Supporting Actors: None

Pre-Conditions: The developer must be logged in the account

Post-Conditions: Success end condition: The developer will have Java code and XML of class diagram uploaded in the system.

Failure end condition: The Java code and XML will not be uploaded and system state will be unaffected.

Trigger: The developer provides paths to Java code and XML of class diagram and clicks on the upload button

Main Success Scenario: The developer provides paths to Java code and XML of class diagram and clicks on the upload button. On successful completion of upload process, the documents will be available for comparison to the developer for detecting any mismatch between given class diagram and the Java code developed.

5.7 Use Case: Run Comparator

Id: UC-5

Description: The developer runs the comparator to detect mismatch between XML of class diagram and the Java code.

Primary Actor: Developer, Reviewer

Supporting Actors: None

Pre-Conditions: The developer must be logged in the account

Post Conditions: Success end condition: The system will display a checklist for various attributes and their status(Pass/Fail) after comparing the XML of class diagram and the Java code developed both of which are already uploaded in the system.

Failure end condition: The Java code and XML will not be compared and system state will be unaffected. A failure message will be displayed to the developer.

Trigger: The user clicks on the "compare code" button.

Main Success Scenario: The user clicks on the "compare code" after uploading both the XML of class diagram and the developed Java code. On successful completion of compare process, a checklist

will be displayed for various attributes and their status (Pass/Fail) after comparing the XML of class diagram and the Java code. The attributes present in the checklist will be-

- Class Name
- Class Attributes: attribute name, attribute type, attribute visibility
- Class Operations: operation name, operation return type, operation visibility, operation parameter details: parameter name, parameter type
- Class relations to other classes

5.8 Use Case: View code and XML

Id: UC-6

Description: The developer can view the code and XML uploaded by him/her in the system.

Primary Actor: Developer, Reviewer

Supporting Actors: None

Pre-Conditions: The developer must be logged in the account

Post Conditions: Success end condition: The user will be displayed the Java code file or XML file depending on the option selected by him/her.

Failure end condition: No file will be displayed and a failure message will be displayed to the user.

Trigger: The user clicks on the view code or view XML button.

Main Success Scenario: The user can see the uploaded file by clicking the view code or view XML button. The user will be able to view the Java code file or XML file depending on the option selected by him/her. In case the path of code specified by the developer was that of the Java project folder, a single file containing all the class (.Java) codes of the project will be created and displayed to the user.

5.9 Use Case: Detect Anti-patterns

Id: UC-7

Description: The developer can view the code and XML uploaded by him/her in the system.

Primary Actor: Developer, Reviewer

Supporting Actors: None

Pre-Conditions: The developer must be logged in the account

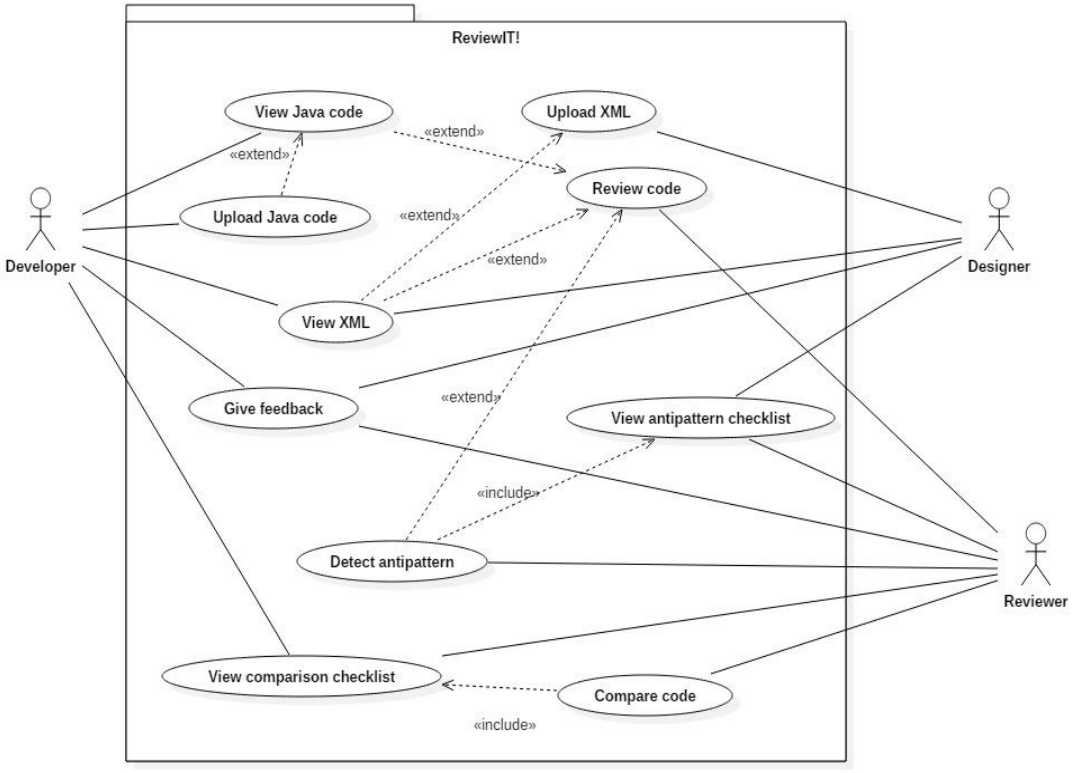
Post Conditions: Success end condition: The user will be displayed the Java code file or XML file depending on the option selected by him/her.

Failure end condition: No file will be displayed and a failure message will be displayed to the user.

Trigger: The user clicks on the view code or view XML button.

Main Success Scenario: The user can see the uploaded file by clicking the view code or view XML button. The user will be able to view the Java code file or XML file depending on the option selected by him/her. In case the path of code specified by the developer was that of the Java project folder, a single file containing all the class (.Java) codes of the project will be created and displayed to the user.

5.10 Use Case Diagram



6. Conceptual Diagram

The conceptual model is a model of significant concepts, attributes, and associations in the problem domain. The model also provides information about relationships and multiplicity in the interactions between concepts. Creating the conceptual model helps to gain a better perspective through an object-oriented analysis of the problem domain.

6.1 Noun Phrase Analysis of Problem Statement

Nouns:

Software, industry, decades, project, code, project management staff, development process, requirements, end product, defect, rework, literature, design, coding phase, experts, tool, ReviewIT!, Java Source Files, Object Oriented Design, UML Class diagrams, XML format, information, classes, relations, checklists, Designer, Design XML, system, Anti Pattern Detector, anti-patterns, design analysis results, Developer, JAVA Code, comparator, comparison results, Reviewer, defined classes, class structures, methods, relations, associated classes, review functionality, subject matter experts/experienced programmers, project management, feedback, action, changes, comments.

Good Classes:

Developer, JAVA Code, Designer, Design XML, Comparator, Checklist, Reviewer, Anti Pattern Detector, Design Analysis Result

Bad Classes:

Software, industry, decades, project, code, development process, requirements, end product, defect, rework, literature, design, coding phase, experts, tool, ReviewIT!, Object Oriented Design, UML Class diagrams, information, classes, relations, system, anti-patterns, comparator, defined classes, class structures, methods, relations, associated classes, review functionality, feedback, action, changes, comments.

Not Known:

Subject matter experts/experienced programmers, project management

6.2 Domain Model

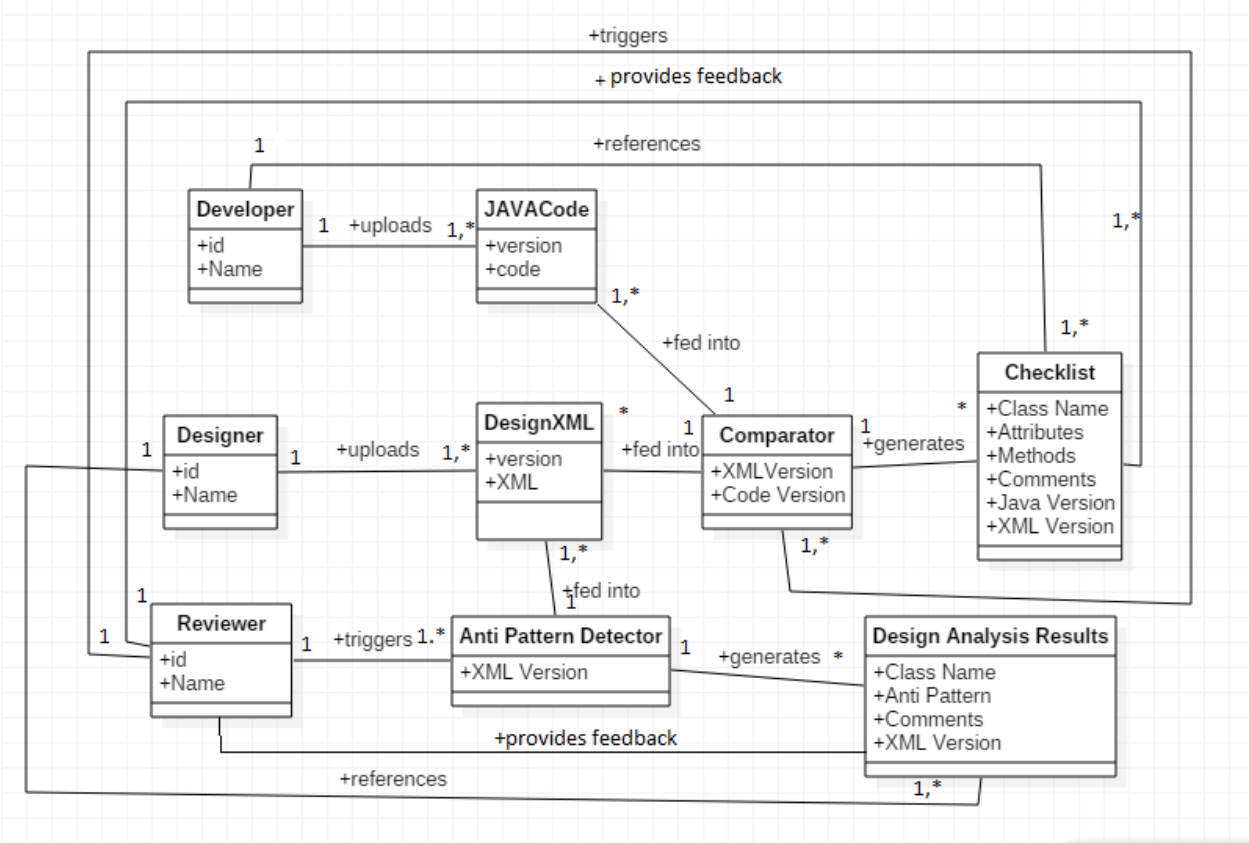


Fig: Domain Model of ReviewIT!

7. Activity Diagram

Activity diagram is an important diagram n UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

7.1 Detection Of Anti Patterns

The Designer uploads the XML of class diagram into the system. The XML is fed to the anti-pattern detector. Reviewer clicks on 'Detect Anti Patterns', views the results and provides a feedback. The feedback reaches the Designer, who can then modify the design in case of non-acceptance.

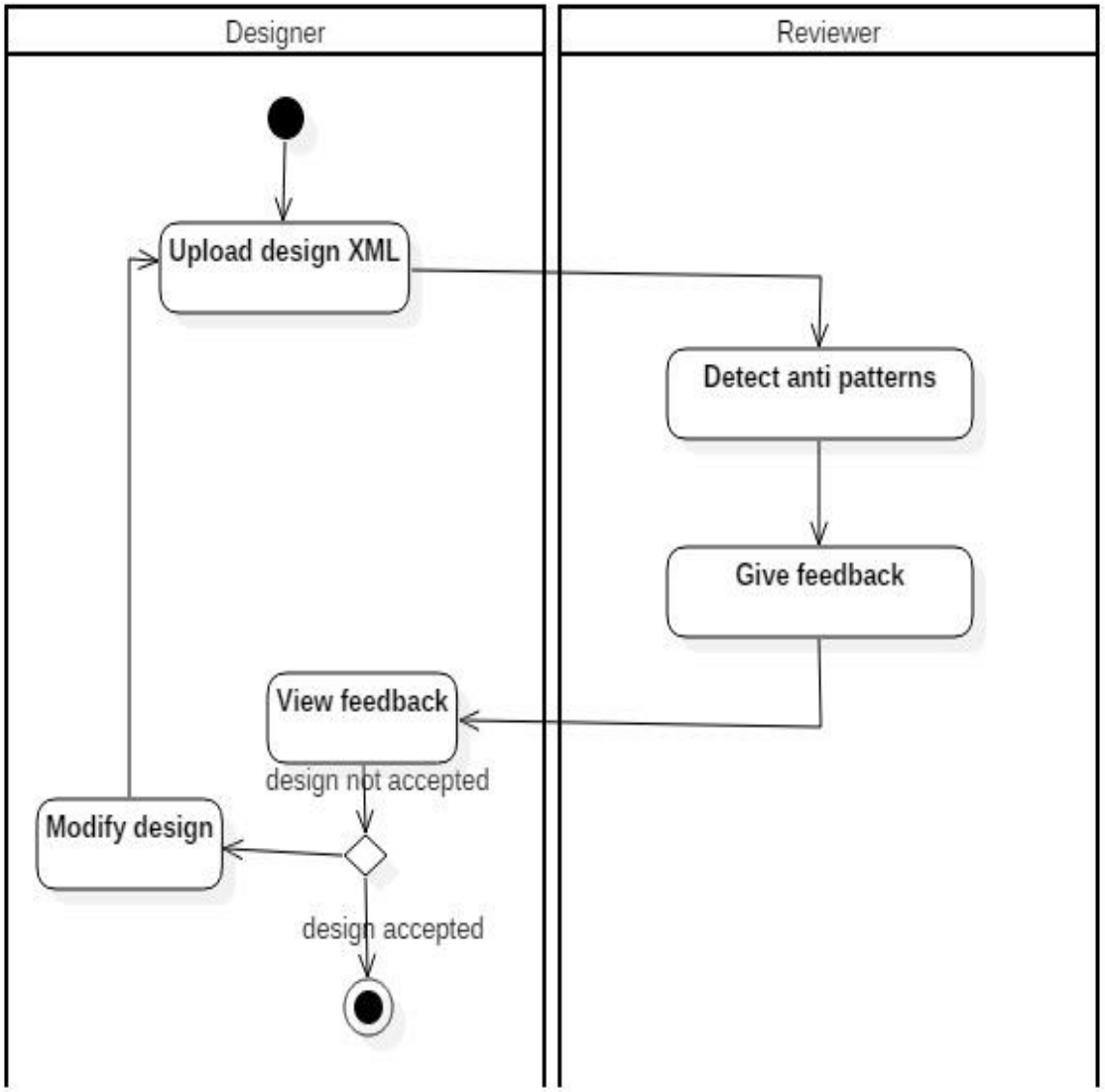


Fig: Activity diagram for detecting anti pattern in XML of class diagram

7.2 Comparison of XML & JAVA Code

The Developer uploads his/her JAVA code, from where the code is fed to a system which already contains the XML of Class Diagram. The Reviewer triggers the comparison, views the checklist and sends a feedback to the Developer, who can then modify his/her code in case of non-acceptance.

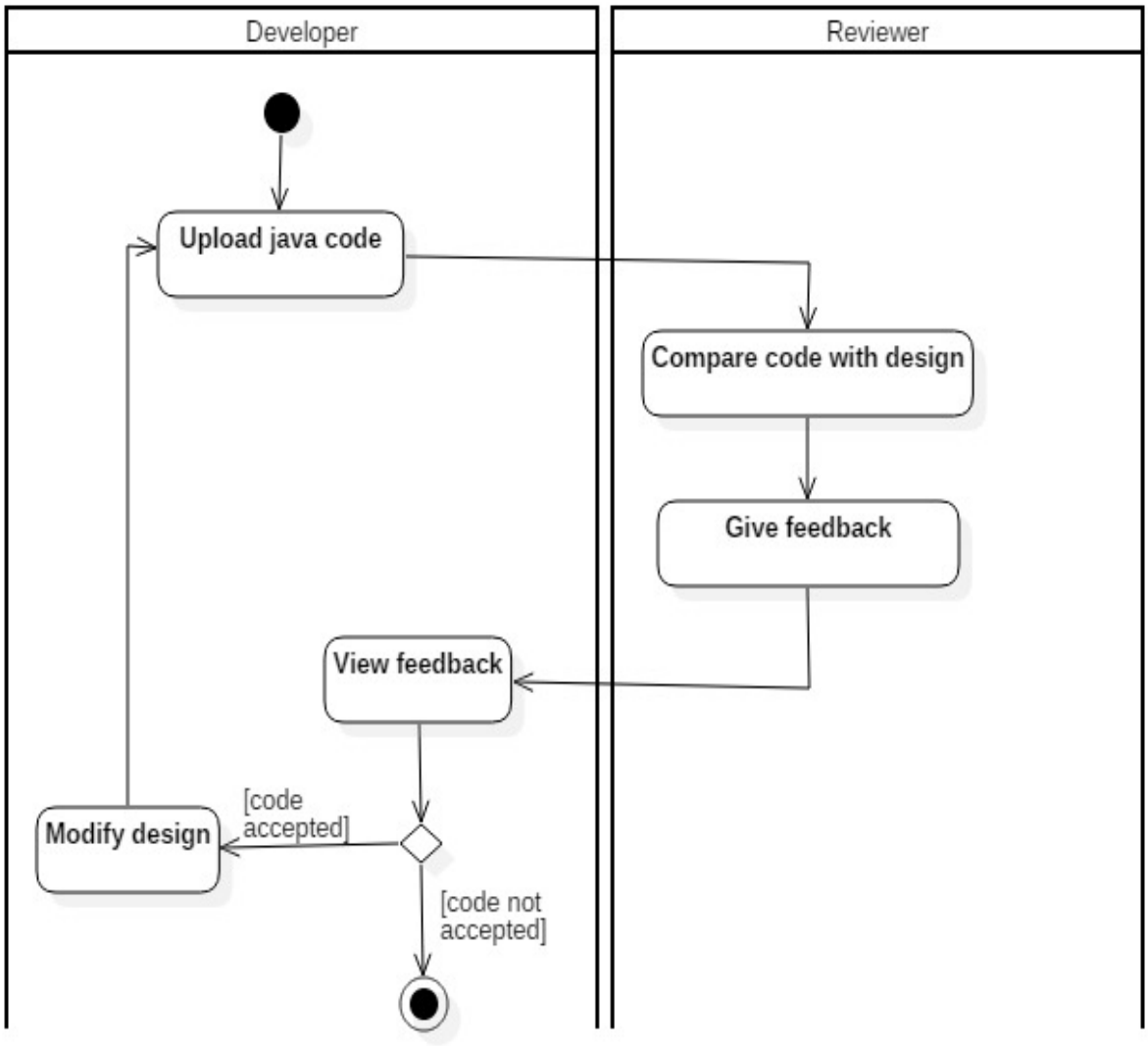


Fig: Activity diagram for comparing XML of class diagram with Java Code

8. System Behavior

Before moving on to logical design, System Behavior needs to be analyzed to define how a system behaves. System Behavior is a description of what a system does, while hiding the actual implementation. For this document, collaboration diagrams are used to demonstrate the important system behaviors of the UDP Host.

8.1 Sequence Diagram

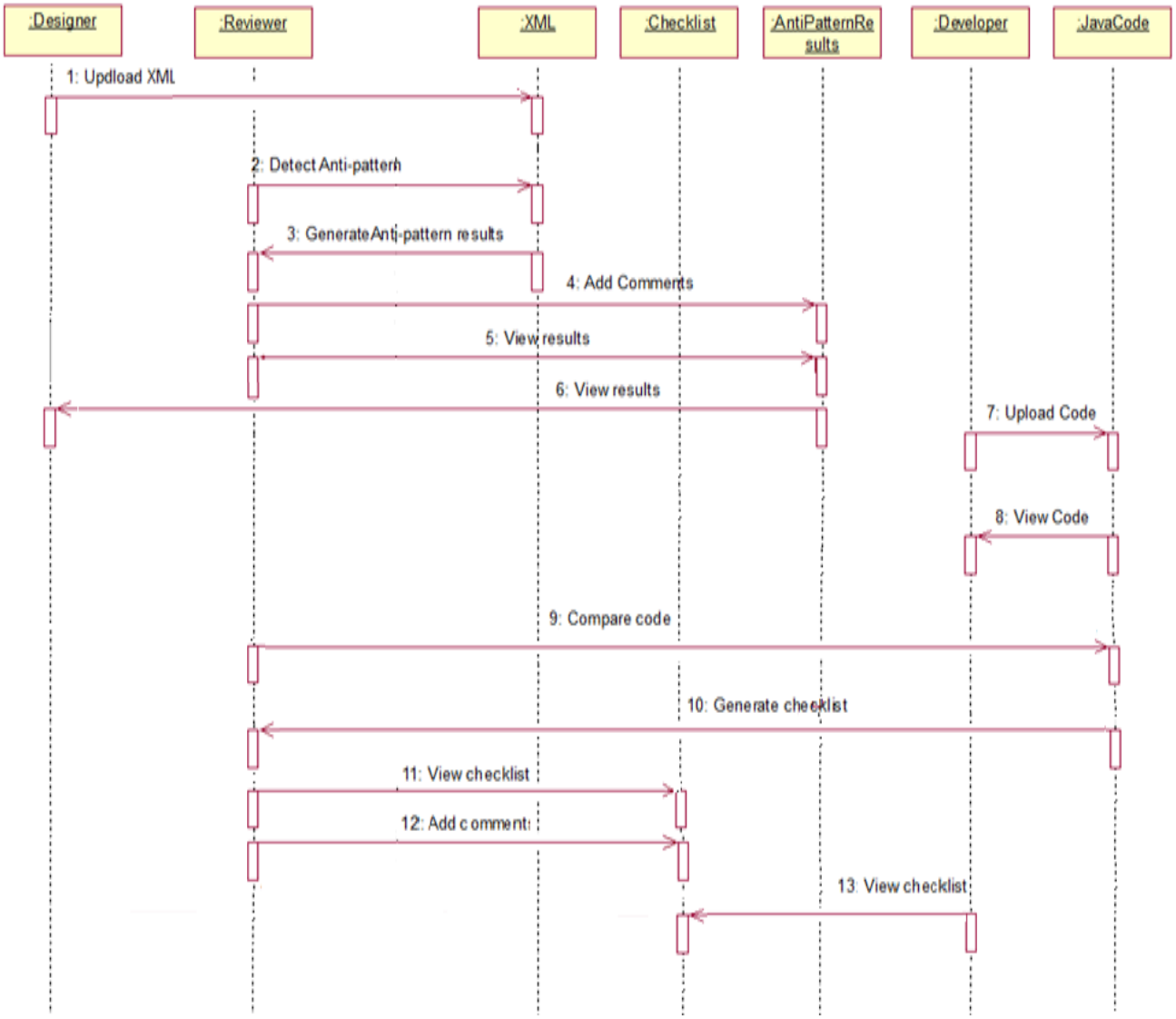


Fig: Sequence Diagram for main success scenario of ReviewIT!

8.2 Collaboration Diagram

Collaboration Diagrams illustrate the interactions between objects in a graph format.

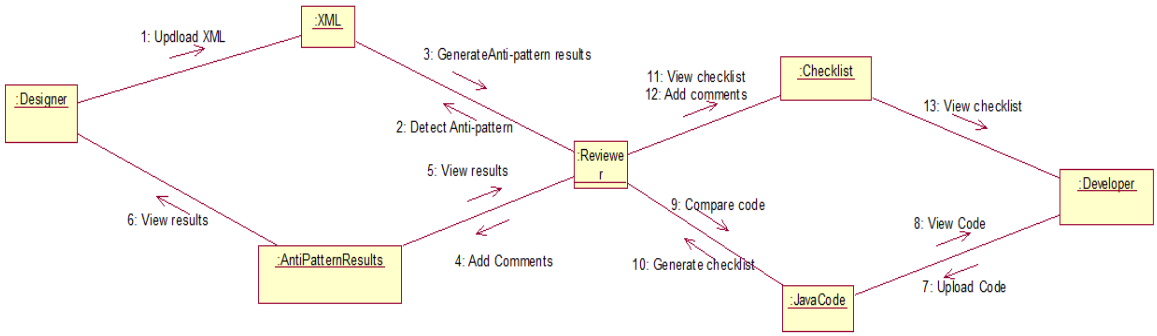


Fig: Collaboration Diagram for main success of ReviewIT!

9. Classes

Classes are the set of objects that share the same attributes, operations, methods, relationships, and semantics. A good software engineering technique requires an engineer to divide up the system responsibilities into smaller sub-responsibilities. This separation process is beneficial because as a result of this process, the system gains many positive qualities such as reusability, modularity, and encapsulation.

9.1 Class Diagram

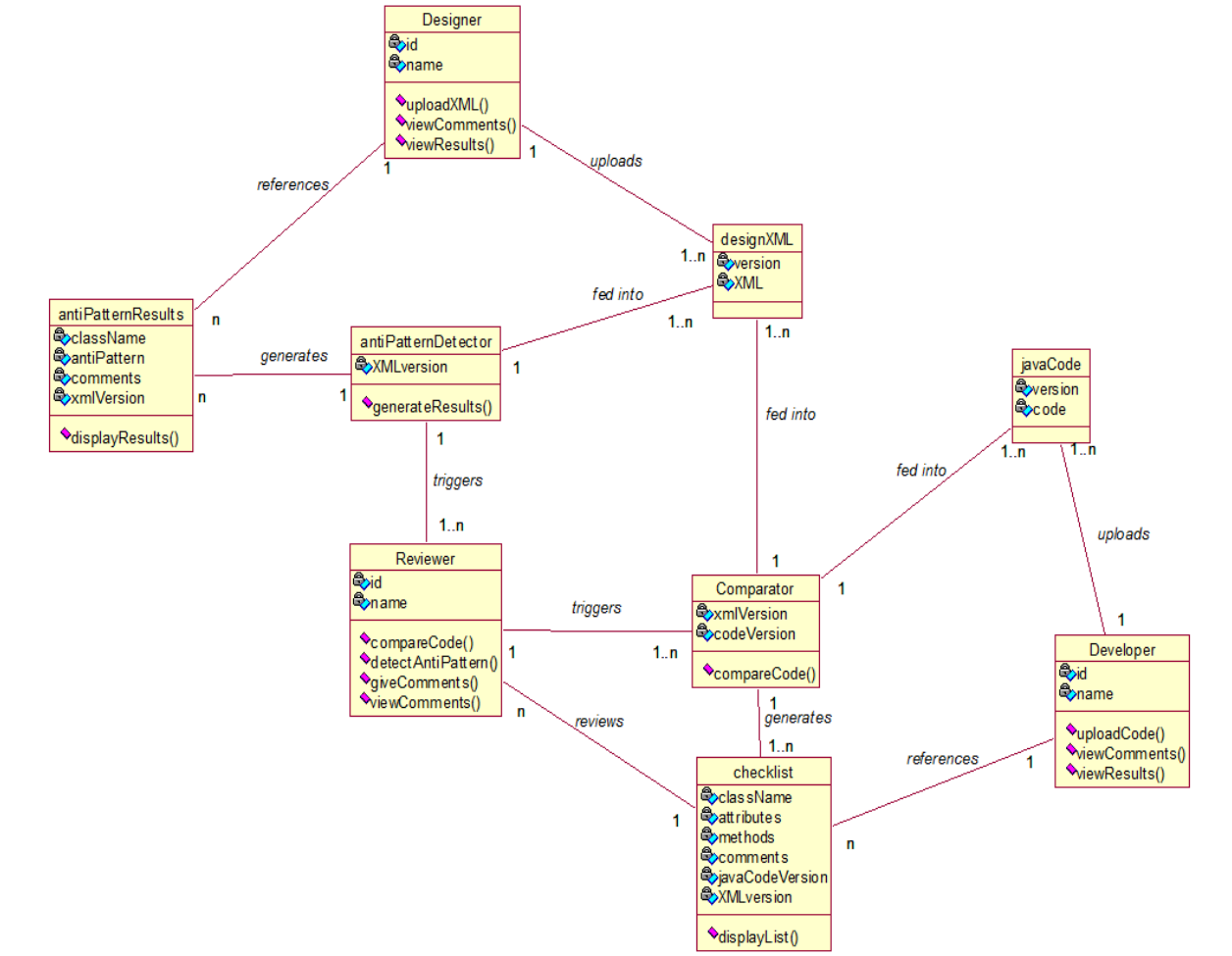


Fig: Class Diagram of ReviewIT!

10. Problems, Conflicts & Resolution

There were various challenges faced by the team during the development of this project. The first and foremost of the challenges was to analyze and finalize the requirements and scope of the project. Various discussions and meetings amongst the team members resulted in defining the essential features of the project. Another challenge faced by the team members was the lack of prior experience of working with tools and technologies required for project development. The team members learned to use tools such as Rational Rose and StarUML for creating various UML diagrams and used Java as programming language for implementation. Overall harmony and coordination amongst the team members led to successful completion of all the modules that were planned to be delivered.

11. Glossary

Design: A document/diagram that describes how the system needs to be developed.

XML: Extensible Markup Language is a markup language that provides a more structured medium than HTML, allowing us to define new document types and style sheets, as needed providing methods to add well-defined markup to electronic documents.

Reviewer: A person who monitors the status of different components of a project, and provides a valuable review.

Designer: A person who prepares the design of system to be developed.

Developer: A person who implements the design provided by the Designer.