

Use LLM to improve
software quality

Objective

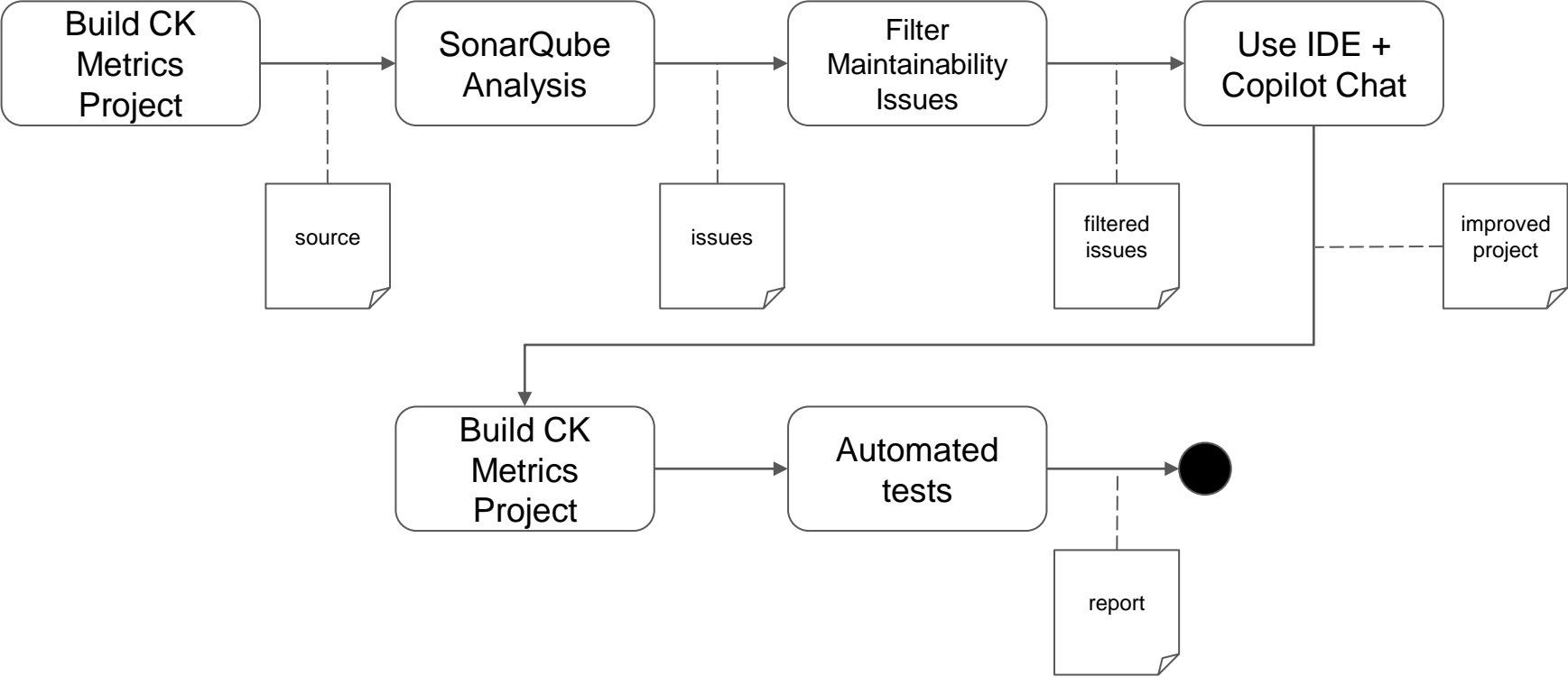
Assess the capability of Large Language Model (LLM), integrated in an IDE, to improve the readability of methods from Github with design issues.

Pilot: Project

Simple Java project (academic tool):

Project	Language	Stars	Contributors	Classes	Methods
CK Metrics 0.7.0	Java	344	23	688	9.185

Pilot: Design

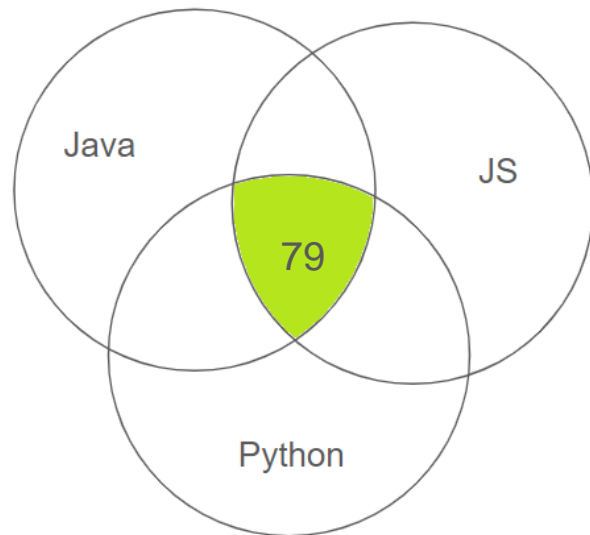


Pilot: Issues Selection

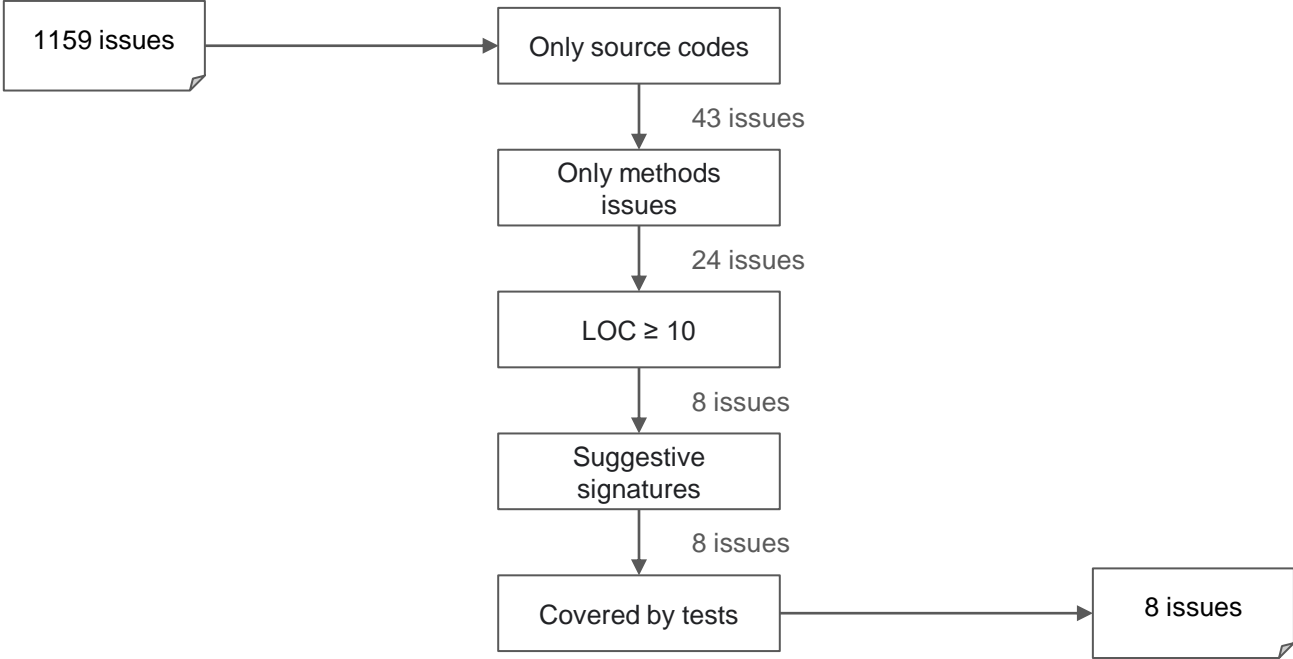
Extract all issues from SonarQube database.

Chosen only common issues in Java, JS and Python (79 rules).

Sonar Analysis: 1159 issues (CK metrics)



Pilot: Methods Filter



Pilot: Prompts

id	path	issue_line	method_signature	prompt
1	src/main/java/com/github/mauricioaniche/ck/util/MetricsFinder.java	68	private void loadMethodLevelClasses(boolean variablesAndFields)	In this class, the method 'private void loadMethodLevelClasses(boolean variablesAndFields)' has the following issue 'Generic exceptions should never be thrown'. Can you identify, fix it and change old code for new code? Import new libraries, if it is necessary.
2	src/main/java/com/github/mauricioaniche/ck/util/FileUtils.java	31	public static String[] getAllDirs(String path)	In this class, the method 'public static String[] getAllDirs(String path)' has the following issue 'Generic exceptions should never be thrown'. Can you identify, fix it and change old code for new code? Import new libraries, if it is necessary.
...
8	src/main/java/com/github/mauricioaniche/ck/CKVisitor.java	162	public boolean visit(AnonymousClassDeclaration node)	In this class, the method 'public boolean visit(AnonymousClassDeclaration node)' has the following issue 'Unused assignments should be removed'. Can you identify, fix it and change old code for new code? Import new libraries, if it is necessary.

Pilot: Results

7 changes passed in compilation and automated tests

1 change have compilation error

Pilot: Use of Copilot Chat in VS Code

```
50
51     public CK() {
52         this(false, 0, true);
53     }

```

In this class, the method 'public void calculate(String path, CKNotifier notifier)' has the following issue 'String literals should not be duplicated'. Can you identify, fix it and change old code for new code? Import new libraries, if it is necessary. Use and import only libraries that the project already have.

Accept Discard ↕ ↻ Changed 3 lines


```
54
55     private static final String JAVA_FILE_EXTENSION = ".java";
56
57     public void calculate(String path, CKNotifier notifier) {
58         String[] javaFiles = FileUtils.getAllJavaFiles(path);
59         log.info("Found " + javaFiles.length + " java files");
60
61         Path basePath = Paths.get(path);
62         calculate(basePath, notifier,
63             Stream.of(javaFiles)
64                 .map(basePath::resolve)
65                 .filter(file -> file.toString().endsWith(JAVA_FILE_EXTENSION))
66                 .toArray(Path[]::new)
67         );
68     }
69
70     /**
71     * Convenience method to call ck with a path rather than a string

```

Pilot: Example 1 - Generic Exception

```
private void loadMethodLevelClasses(boolean variablesAndFields) {
    try {
        Reflections reflections = new Reflections("com.github.mauricioaniche.ck.metric");

        methodLevelClasses = sorter.sort(reflections.getSubTypesOf(MethodLevelMetric.class).stream()
            .filter(x -> variablesAndFields || !Arrays.asList(x.getInterfaces()).contains(VariableOrFieldMetric.class))
            .collect(Collectors.toList()));
    } catch (Exception e) {
        throw new RuntimeException("Could not find method level metrics. Something is really wrong", e);
    }
}
```



```
private void loadMethodLevelClasses(boolean variablesAndFields) {
    try {
        Reflections reflections = new Reflections("com.github.mauricioaniche.ck.metric");

        methodLevelClasses = sorter.sort(reflections.getSubTypesOf(MethodLevelMetric.class).stream()
            .filter(x -> variablesAndFields || !Arrays.asList(x.getInterfaces()).contains(VariableOrFieldMetric.class))
            .collect(Collectors.toList()));
    } catch (ReflectionsException | NoSuchMethodException | IllegalAccessException | InstantiationException | InvocationTargetException e) {
        throw new RuntimeException("Could not find method level metrics. Something is really wrong", e);
    }
}
```


Pilot: Example 2 - Unused Variable

```
public boolean visit(MethodDeclaration node) {  
    IMethodBinding binding = node.resolveBinding();  
    boolean isConstructor = node.isConstructor();  
  
    CKMethodResult currentMethod = new CKMethodResult(JDTUtils.getMethodFullName(node), JDTUtils.getQualifiedMethodFullName(node),  
    currentMethod.setLoc(locate(node.toString()));  
    currentMethod.setStartLine(JDTUtils.getStartLine(cu, node));  
  
    // let's instantiate method level visitors for this current method  
    List<MethodLevelMetric> methodLevelMetrics = instantiateMethodLevelMetricVisitors(JDTUtils.getQualifiedMethodFullName(node));  
}
```

```
public boolean visit(MethodDeclaration node) {  
  
    CKMethodResult currentMethod = new CKMethodResult(JDTUtils.getMethodFullName(node), JDTUtils.getQualifiedMethodFullName(node),  
    currentMethod.setLoc(locate(node.toString()));  
    currentMethod.setStartLine(JDTUtils.getStartLine(cu, node));  
  
    // let's instantiate method level visitors for this current method  
    List<MethodLevelMetric> methodLevelMetrics = instantiateMethodLevelMetricVisitors(JDTUtils.getQualifiedMethodFullName(node));  
}
```

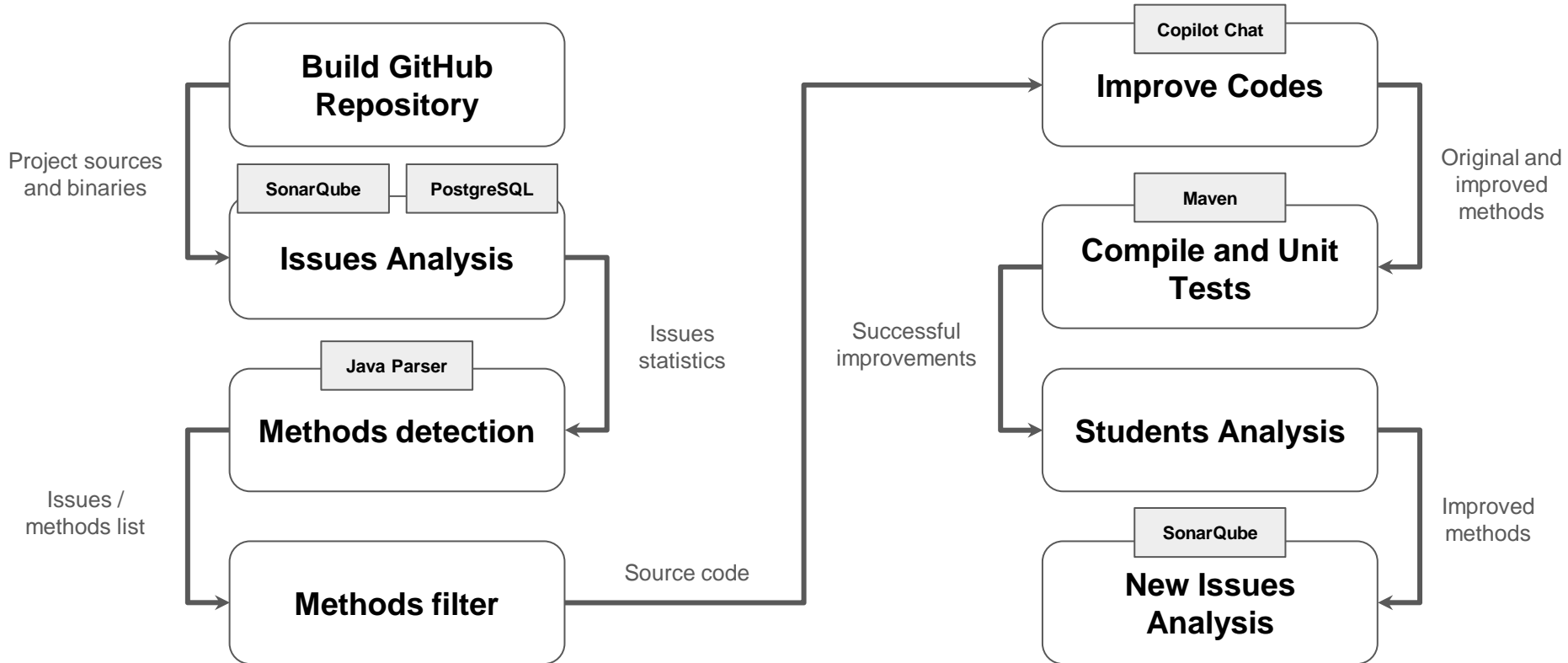
Pilot: Example 3 - Duplicated String Literals

```
public void calculate(String path, CKNotifier notifier) {  
    String[] javaFiles = FileUtils.getAllJavaFiles(path);  
    log.info("Found " + javaFiles.length + " java files");  
  
    calculate(Paths.get(path), notifier,  
             Stream.of(javaFiles)  
                   .map(Paths::get)  
                   .toArray(Path[]::new)  
             );  
}
```



```
public void calculate(String path, CKNotifier notifier) {  
    String[] javaFiles = FileUtils.getAllJavaFiles(path);  
    log.info("Found " + javaFiles.length + " java files");  
  
    Path basePath = Paths.get(path);  
    calculate(basePath, notifier,  
             Stream.of(javaFiles)  
                   .map(basePath::resolve)  
                   .toArray(Path[]::new)  
             );  
}
```

EXPERIMENT EXECUTION



Thank you!