

Optimized and refined bug/file correlation

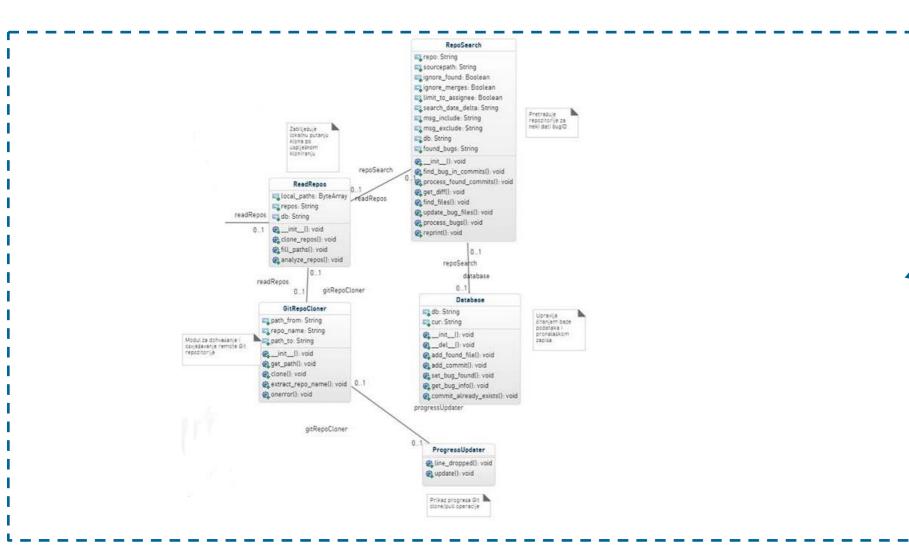
Armando Vega, Marko Kujundžić, Dea Marin

Summary

- •Project goals included several search refinement options:
 - •Limiting the search to a number of months before/after closure of a given bug
 - •Limiting the search to messages (not) containing certain specified keywords
 - •Limiting the search to commits made by the bug assignee
 - •Ignoring certain types of commits, e.g. merge commits
 - •Ignoring searching for bugs that have already been found
- Libraries/tools required: Git, MySQL, GitPython, PyGTK

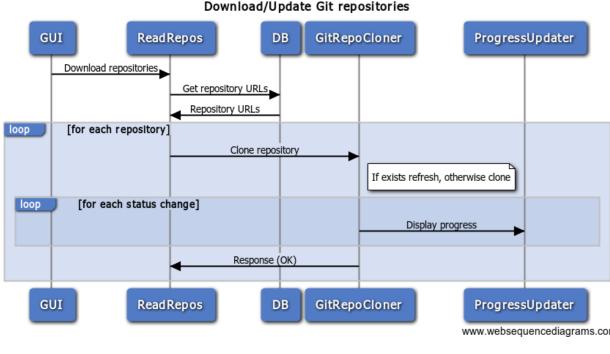
Developed Component

Class diagram



- The ReadRepos uses the GitRepoCloner for Git repository maintenance
- It also uses RepoSearch for searching the repository history for BugIDs and link them to corresponding files
- GitRepoCloner uses ProgressUpdater to display Git repository clone/pull operation progress
- RepoSearch uses Database for fetching bug information and storing data about found matching commits and corresponding files

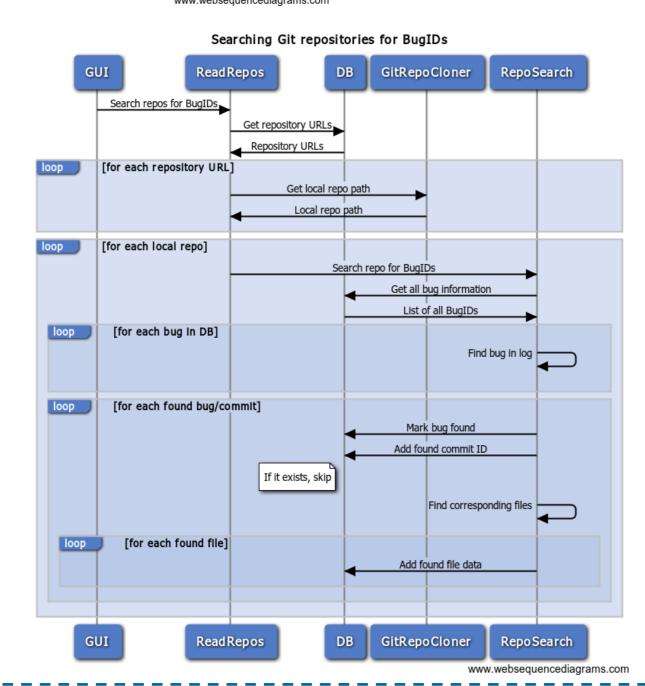
Sequence diagram



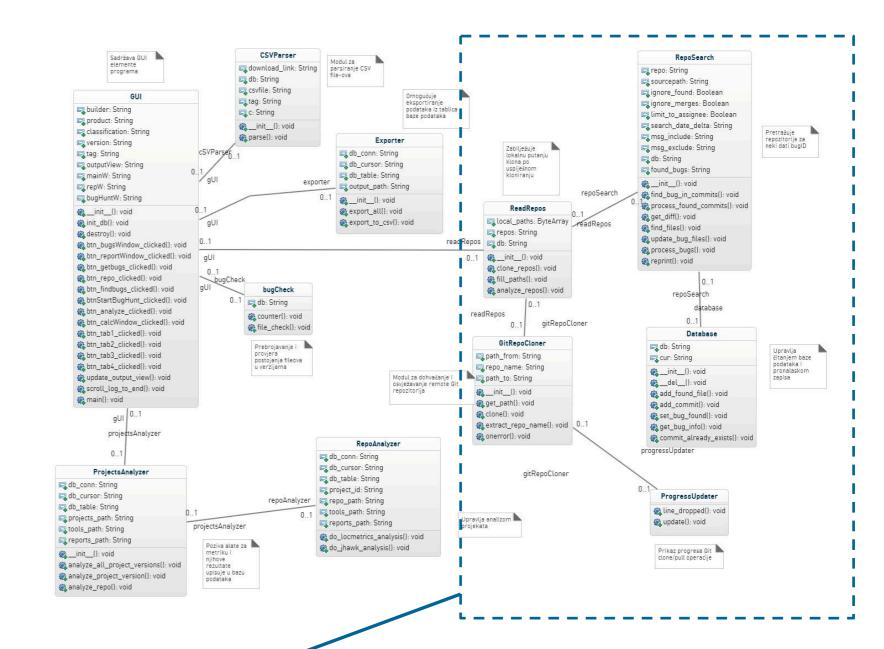
- User can initiate full
 Git repository
 download or update
- It displays progress as it is working

• User initiates the analysis of local Git repositories

- The module retrieves the list of bugs from the database
- It searches the history of every repository for given BugIDs
- It writes data about corresponding files into the database



Overall Architecture



• Architecture components: CSV parser, Project/repository analyzer (using external tools), Repository maintainer, Repository history analyzer, Database abstraction utilities

GUI

Bug finding refinement dialog

Dug jinang rejinement atatog						
	Bug finding refinement				Σ3	
	Pretraži unutar	etraži unutar		mjeseci prije i poslije Bug Changed		
	Message uključuje	resolve, bug				
	Message NE uključuje					
	Ograniči povezivanja na Bug Assignee					
	☑ Ignoriraj "merge" commitove					
	☑ Ignoriraj već pronađene bugove					
			Sta	rt		

- The GUI presents the user with multiple choices of bug finding refinement
- After checking desired options and specifying required input, the user can start the optimized search process by clicking the provided "Start" button
- This part of the GUI provides the facade to the optimized *BugID->Commit->File* finding and correlating process

Conclusion

- Experienced problems
 - Regular expressions The default distribution of the Git binary is missing PCRE (negative lookaround) support which limited the efficiency of the Git repository history searching process
- Learned concepts
 - Data layer abstraction
- Future work improvements
 - Further optimizing the search process for accuracy
 - Implementing threads to provide for parallelization of the search process
 - Displaying the progress of all longer lasting processes through a standard GUI element, such as a progress bar
 - Implementing and using an ORM solution for easier data object handling and future extendability