Automatic Debug Down to the Line

Daniel Hansson, Patrik Granath

verifyter

Introduction

Binary Search

a) Can find a single regression bug

Step	Commits (forming one Timeline) 1 2 3 4 5 6 7 8	Result
1	■	Pass
2		Fail
3		Pass
4		Fail
5		Fail
Result		

b) Cannot handle two regression bugs

Step	Commits (forming one Timeline) 1 2 3 4 5 6 7 8	Result
1	■	Pass
2		Fail
3		Pass
4		Fail
5		Fail
Result		(Pass)

Delta Debugging

a) Can debug independent code chunks

Step	Code Chunks 1 2 3 4 5 6 7 8	Result
1	■ ■ ■ ■	Pass
2	■ ■ ■ ■	Fail
3	■ ■	Pass
4	■ ■	Fail
5	■	Fail
Result		

b) Cannot debug along a timeline

Step	Commits (forming one Timeline) 1 2 3 4 5 6 7 8	Result
1	■ ■ ■ ■	Pass
2	■ ■ ■ ■	Fail
3	■ ■	Fail
4	■	Fail
Result		(Fail)

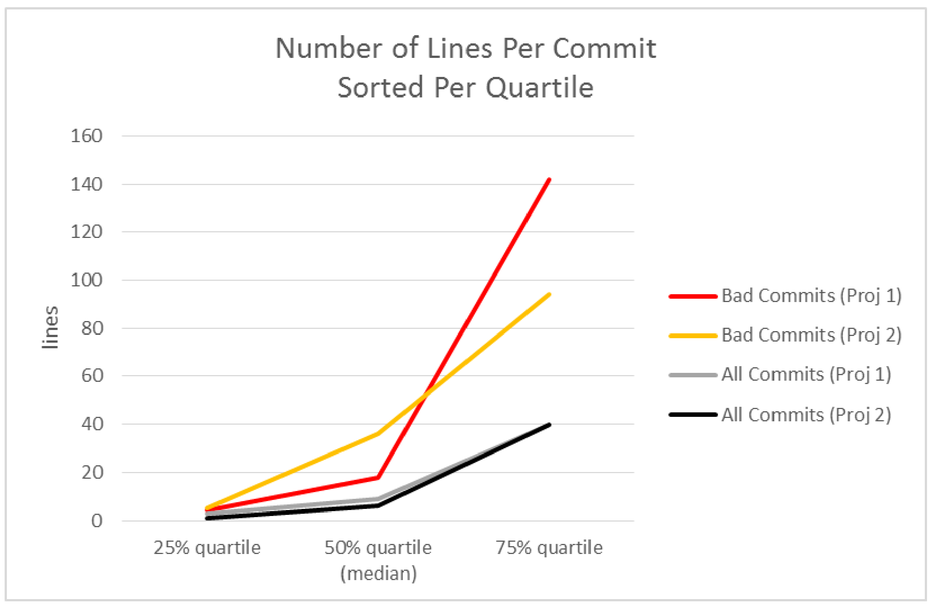
Summary

No of Bugs	Timeline Exists	No Timeline
Single Bug	Binary Search	Both
Multiple Bugs	None	Delta Debug (only independent bugs)

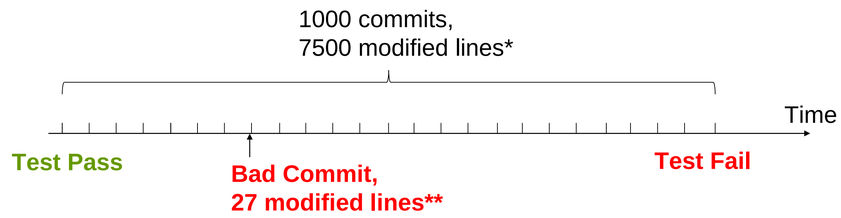
Results

Commit Sizes in real ASIC projects

Project	No of Commits Between Bad Commits	Days Between Bad Commits	Time Period	Total Commits
1	40.5	2.8	Jan 1 – Aug 11 2014	4690
2	45	1.3	Aug 1 2013 – Mar 1 2014	8585



Example using the Median Commit Size



* median commit size is 7.5 lines
** median bad commit size is 27 lines

Using Binary Search to find the bad commit and Delta Debugging to find the bad chunks within the commit

Iteration	Find Bad Commit, Then Delta Debugging (commits)	(line chunks)
1	Every 100	
2	Every 10	
3	Every 1	**Within bad commit:**
4		14, 7, 3 chunks
5		4,3,2,1 line chunks
6		1,2 line chunks
7		1 line chunks

Conclusion

Combining the two algorithms provides a good debug time for line granularity

Scenario	Debug Time (No of Times to re-run Failing Test)
1000 commits	5-7
100 commits	4-6
10 commits	3-5