

# Role of Functional Clones in Software Development

Kavitha Esther Rajakumari<sup>1</sup>, Dr.S.Srinivasan<sup>2</sup>

<sup>1</sup>Sathyabama University

<sup>2</sup>Anna University

**Abstract:** *The software organizations have a great demand on producing quality software. The methodology used to produce quality software varies depending on the organization's time, cost and resource constraints. In most cases, as the need for quality arises cost and time also increases. An efficient technique helps to cope up with these constraints and deliver a quality software system. In this paper, a cost-effective method is proposed to produce quality software. This proposed method uses the concept of code clones to effectively develop quality software. Most of the surveys regarding code clones project its negative aspects, but here the positive side of it is focused. The results show that this method uses limited resources and also user-friendly.*

**Keywords:** *Quality software, software development, code clones, functional clones.*

## 1. Introduction

Software engineering has three major tasks embedded in it. They are software development, software reuse and software maintenance. Developing quality software helps in the steady growth of an organization. The techniques and methods used by an organization in software development determine its economical growth. Some techniques help in developing quality software but very expensive. Some methods are complex which are too difficult to implement. Some are simple but time consuming. Some techniques make use of more resources which indirectly leads to lack of resources for other tasks. So choosing an appropriate method is very important in developing software. Most of the organizations are equipped with customized tools which help in software development. Some organizations especially small-scale industries are not able to face the challenge of producing quality products. Industries which have very limited resources are often considered as small-scale industries. The proposed method paves way for these types of organizations to carry out the development process successfully.

In this paper, the proposed method makes use of code clones. Implementing this method is less expensive, affordable and meets the requirements of small-scale industries. Code clones are identical codes present in a software system. They are also known as duplicate codes. There are four types of code clones. Type-1 clones are the exact replica of the original code fragments. Type-2 clones are also identical to original fragments but names of variables, identifiers etc are changed. Type-3 clones are ones where the original fragment is modified i.e. additional changes or deletions are done. Type-4 clones are functionally similar. In general, code clones are considered harmful which deteriorate the software's quality. But by careful analysis beneficial clones can be extracted which aid in improving the quality.

## 2. Literature Survey

The survey based on software engineering and code clones are discussed in this section. Francesca and his team mates [1] have discussed about the various tools assist in detecting code clones. They consider clones as bad smells. James R Cordy and Chanchal K. Roy [2] have proposed a tool named DebCheck to detect code clones in open source software systems. Dongxiang and Miryung [3] have found that clones that stay in the program over a long period of time cannot be refactored as soon as it is detected. They have termed those code clones as long-lived clones. The authors have mentioned that clones are not necessarily harmful.

The need for a new software development approach was discussed by Rupinder and Dr.Jyotsna [4]. They have listed some of the major issues faced in software development such as over budget, premature termination of projects etc. The concept of sharing knowledge among the development team members is stated as the key process in software development by Sharon and Rory [5]. Jesper and his peer group [6] have indicated that component reuse and iterative processes increase adaptability to changing customer needs.

The above discussions show that clones are identified as one of the bad smells present in the software. Therefore detecting and removing clones is one of the processes while developing software systems. Moreover the software development process should be fine tuned and must be able to overcome the drawbacks of the organization. Although the demerits of an organization have an impact on the products being developed, proper steps should be taken to maintain quality at an acceptable level.

### 3. Proposed Method

The proposed method makes use of the data mining approach to extract functional clones. Among the available data mining algorithms, FP-Growth algorithm [7, 8] is used. Though it is an existing algorithm, it proves to be efficient when used in source codes. The input to the system can be any software application. The output gives the set of functional clones used in the software system. They are maintained in the database for future use.

The proposed method consists of five steps, each of which is briefly given below.

- **Input Selection:** In this phase, the input to the system is selected. Mostly the real-time applications are taken so that harmful codes are less in number. Sometimes not-in-use applications are also taken as inputs. Not-in-use means old version application software which was once efficiently working but due to new version release has stopped functioning.
- **Module Separation:** Here the selected software application is separated into modules for further investigation. Each module becomes an input to the proposed system. The application as a whole is also possible to be given as an input but problem of segregating the detected functional clones evolves. As a result modules are taken as input so that classifying functional clones according to their usage will be a simple task.
- **Functional Clones Extraction:** After module separation, functional clones from each module are extracted. Apart from library functions, functions that have occurred more than ones are considered as clones. Beneficial functions with more than five occurrences can be made as library functions for future use.
- **Functional Clones Categorization:** In this phase the extracted functional clones are examined manually to check for harmful codes. If noxious codes are present within a function, those functions are removed. Based on the inspection of codes present inside a function, decision will be made either to reject or retain the function. Beneficial functions are retained and unproductive functions are discarded.
- **Storage of Beneficial Clones:** In this step, the clones that were termed beneficial in the previous phase are stored separately in a database. The database must be highly secured in order to avoid inconsistency among the stored data. The developers alone have access to the database apart from the system administrator and project manager. This database helps in providing useful functional clones in software development, software reuse and software maintenance.

Thus the proposed method assists in the software engineering process efficiently. It helps the developers to complete their project work within the stipulated period. This prospective achieves quality software development in an inexpensive means. So each time when a software is under development the programmer or the developer must keep in mind that while repeating certain codes or functions it should be reasonable and valuable to pave way for future usage . The concept of the proposed method is depicted below.

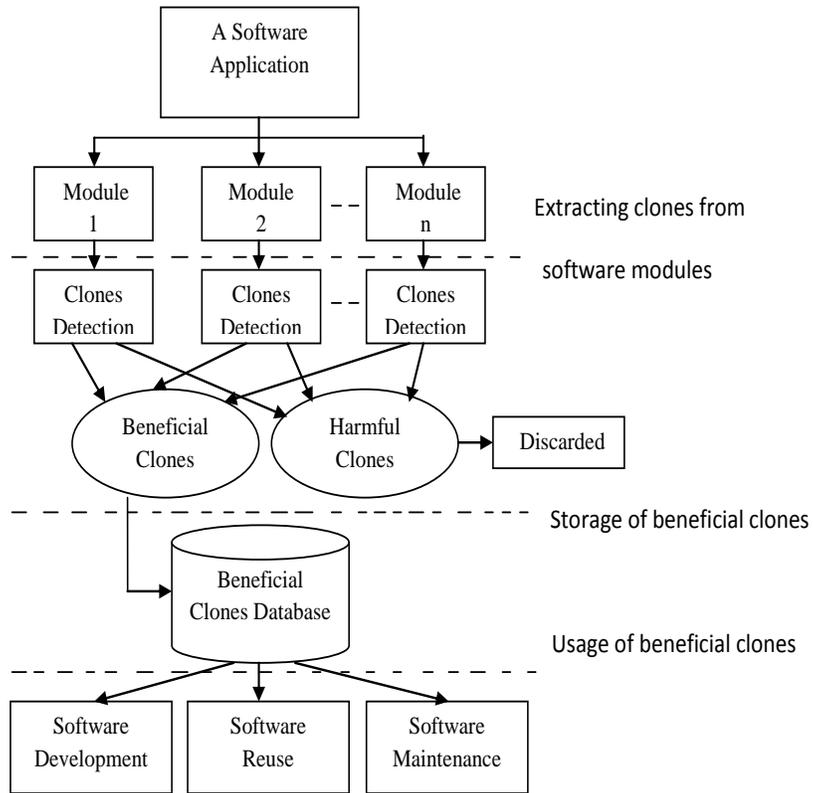


Fig. 1: Block Diagram

#### 4. Implementation

An application software is taken as the input for the proposed system. It asks for the user name and password for authentication purpose so that unauthorized person cannot access the system. This is shown in Fig.2. In Fig.3 browsing the input software is displayed. After entering the system, can browse any application software stored in the database. Here the Project Management System is taken as input.

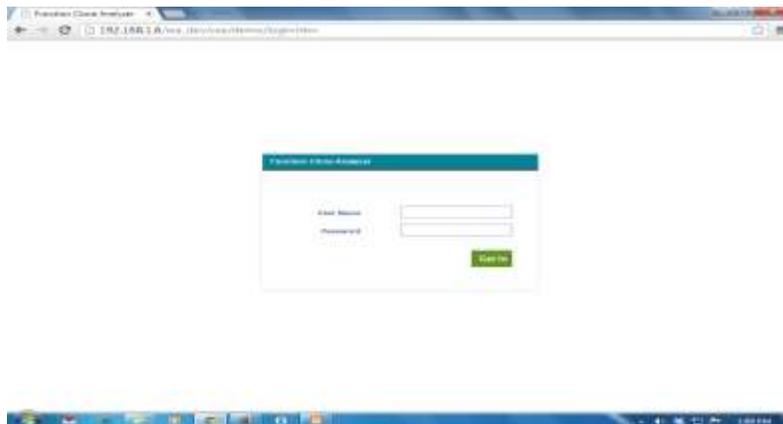


Fig. 2: Login Page

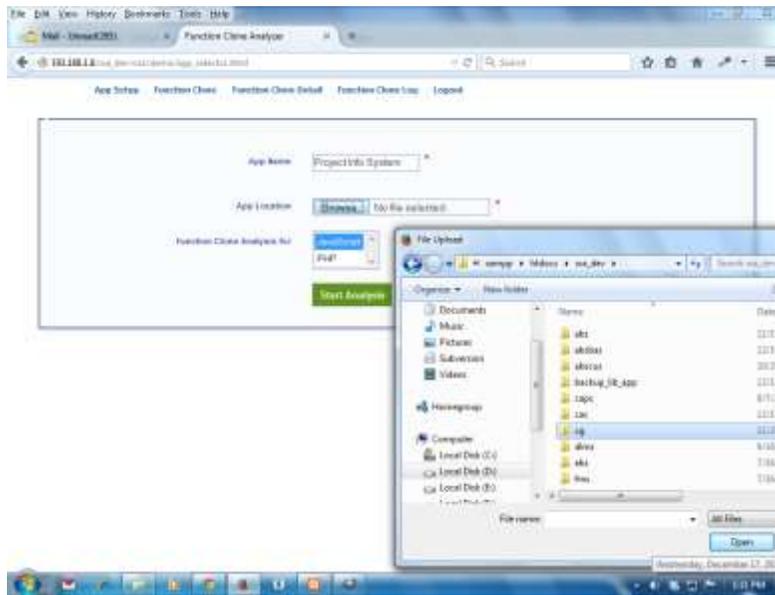


Fig. 3: Browsing the application s/w

After the input software is given to the system, the application software is separated into modules. The Fig.4 below shows some of the modules and the relevant clones detected in each module. Each functional clones extracted from the modules are checked manually for their correctness. When each functional clone is clicked its relevant code is opened. Based on the codes, the usage of the clones are determined and categorized. This process is depicted in Fig.5.

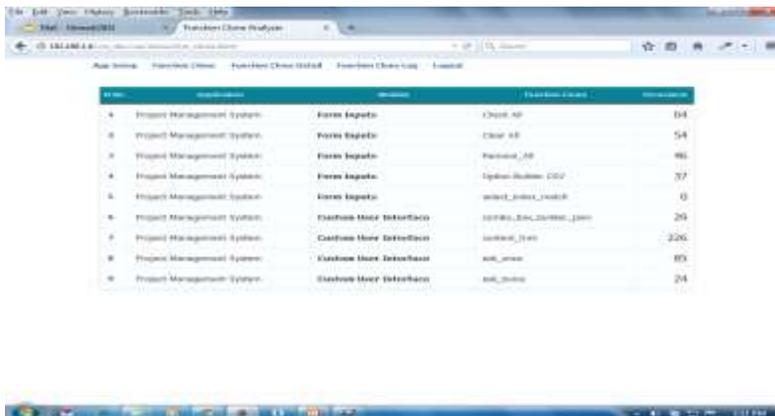


Fig. 4: Module separation and clone detection



Fig. 5: Examination of codes of a functional clone

After categorization, the details of detected clones are displayed. Number of occurrences, line number, page number and the module in which it occurs etc, are given. These details provide the exact location of the extracted clones. These details are a great help when need to track the origin of clones in the future. This conceptualization is shown in Fig.6.

ID No.	Application	Module	Function Clone	Page	Line No.	Occurrences
1	Project Management System	Form Inputs	Check All	proj_new_task.html	24	1
1	Project Management System	Form Inputs	Check All	proj_new.html	23	2
1	Project Management System	Form Inputs	Check All	proj_task_action.html	85	3

Fig. 6: Functional clones description

The figures 7 & 8 displayed below shows the software development process of inserting functional clones in a software under development wherever necessary depending on their usage.

```

320
321     GRAPH.override_settings = 0;
322
323     }else{
324     }var graph_override_confirm = ask_once('Sorry, the given name is already taken');
325
326     if(graph_override_confirm==false){
327
328         D.getElementById("get_save_graph_name").value = "";
329         D.getElementById("get_save_graph_name").focus();
330
331     }else{
332
333         GRAPH.override_settings=1;
334
335         select_match(ELEMENT("get_save_graph_name").value,ELEMENT("graph_variable_group"));
336
337         create_graph_request();
338     } // else
339
340     } // end
341
342 } //end
343
344
345
346 // set graph preset
347 //Graph area functionality
348 function graph_pre_set(element){
349
350     var l_v = new Array();
351
352     l_v["get_data"] = element.options[element.options.selectedIndex].lang;
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Fig. 7: Insertion of a functional clone during s/w development



## 5. Conclusion and Future Work

Code clones have their own advantages and disadvantages. But by careful analysis of detected clones, the beneficial clones are used for software development. Therefore the proposed method proves to be cost- effective and most likely to be used in small- scale industries. This method is user friendly, easy to implement and easy to configure depending on the requirements of the organization.

In the future this method can be extended by adding various features according to the needs of the organization.

## 6. References

- [1] Francesca Arcelli Fontana, Marco Zanoni, Andrea Ranchetti and Davide Ranchetti, “Software Clone Detection and Refactoring,” *ISRN Software Engineering*, Volume 2013 (2013), Article ID 129437, 8 pages, January 2013.
- [2] C.K.Roy and J.R.Cordy, “DebCheck: Efficient Checking for Open Source Code Clones in Software Systems,” in *Proc.2011 IEEE 19<sup>th</sup> International Conference on Program Comprehension*, Kingston, ON, June 22-24, 2011.
- [3] Dongxiang Cai and Miryung Kim, “An Empirical Study of Long-Lived Code Clones,” in *Fundamental Approaches to Software Engineering*, Springer Berlin Heidelberg, 2011, vol. 6603, pp. 432-446.
- [4] Rupinder Kaur and Dr. Jyotsna Sengupta,” Software Process Models and Analysis on Failure of Software Development Projects,” *International Journal of Scientific & Engineering Research*, vol. 2, issue 2, February 2011.
- [5] Sharon Ryan and Rory V. O’Connor,” Acquiring and Sharing Tacit Knowledge in Software Development Teams: An Empirical Study,” *Information and Software Technology*, vol. 55, No. 9, pp. 1614 -1624, September 2013.
- [6] Jesper Pedersen Notander, Martin Höst and Per Runeson,” Challenges in Flexible Safety-Critical Software Development – An Industrial Qualitative Survey,” in *Product-Focused Software Process Improvement*, vol. 7983, pp. 283-297, 2013.
- [7] Ankita Parmar, Kamal Sutaria and Krutarth Joshi, “An Approach for Finding Frequent Item Set Done By Comparison Based Technique,” *International Journal of Computer Science and Mobile Computing*, vol. 3, pp. 996 – 1001, April 2014.
- [8] Abdullah Saad Almalaise Alghamdi, “Efficient Implementation of FP Growth Algorithm-Data Mining on Medical Data,” *International Journal of Computer Science and Network Security*, vol. 11, No.12, pp. 7-16, December 2011.