IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 25, Issue 3, Ser. II (May. – June. 2023), PP 50-55
www.iosrjournals.org

A Review on Enhancement of software quality by the use of various software artefacts to remove code smells.

Indu[1] and Nitika[2]
[1]M.Tech. Scholar, CBS Group of Institutions, Jhajjar, Haryana, India
[2]Assistant Professor, CBS Group of Institutions, Jhajjar, Haryana, India

Abstract
Refactoring is a technique to make a computer program more readable and maintainable.A bad smell is an indication of some setback in the code, which requires refactoring to dealwith. Many tools are available for detection and removal of these code smells. These toolsvary greatly in detection methodologies and acquire different competencies. In this work, westudied different code smell detection tools minutely and try to comprehend our analysis byforminga comparativeoftheirfeaturesandworkingscenario.Wealsoextractedsomesuggestions on the bases ofvariationsfoundintheresultsofboth detectiontools.
Keywords: *Refactoring,CodeSmell,Eclipse,JDeodrant,inCode*

I. Introduction

Refactoringhasbecomeawellknowntechniqueforthesoftwareengineeringcommunity.MartinFowlerhasdefineditasa processtoimprovetheinternalstructureofaprogramwithoutalteringitsexternalbehavior[1].Frequentrefactoringofthe codehelpsprogrammertomakethecodemoreunderstandable,findbugsandmakeitsuitablefortheadditionofnewfeature sandtoprogramfaster.Aboveallthat,itimprovesthedesignofthesoftwareandthereforetheoverallqualityofthesoftware [1].Refactoringcanbedonemanuallyaswellasautomatically.Extensiveliteratureisavailableonrefactoringoftheobject oriented-programsandanumberoftoolsareavailablefortheautomaticrefactoringofthecode.

Refactoringhasaspecialrelationshipwiththeconceptsofreverseengineeringandagilesoftwaredevelopment.Oneofagi lesoftwaredevelopmentmodels,eXtremeProgramming(XP),proposedbybeck[3],considersrefactoringasoneofitsess entialfeatures.Refactoringcontinuouslyimprovesthedesignofthesoftwareandhelpstheevolutionandincrementaldev elopmentofthesoftware.

Differentstepsfortheactivityofrefactoringhavebeenproposedbydifferentresearchers.Tom Mens*etal.*,[4]describedsixactivities intheprocessofrefactoring.Theseare:Identifying;whererefactoringshouldbedone,determiningwhichrefactoring(s)s houldbeappliedtotheidentifiedplaces,ensuringthattheappliedrefactoringpreservesbehavior,applyingtherefactoring, assessingtheeffectsofrefactoringonthequalityofthesoftwareandprocess,assessingtheconsistencybetweenthecodean dothersoftwareartifacts

Badsmellsaredesignflawsorstructuralproblemofsoftwarethatcanbehandledthroughrefactoring.Thetermrefactoring wasfirstproposedbyKentBeckwhilehelping martin Fowler[1].LaterFowlerdidmuch work in thiscontext and this work isstillinprogress.

In[7]smelltaxonomyispresentedbyMartinFowler.Heidentified22badsmellsranging from ordinary bad smells like code duplication and Long parameter list to morecomplexsmellslikeGodClassandFeatureEnvy.Hedividedthese22codesmellsinto 7categoriesbasedontheirsimilarity.Forexampledispensableisoneofthose7categoriesthatincludeslazyclass,dataclass ,duplicatedcodedeadcodeandspeculativegenerality smellsbasedonthe similaritythatallthese incorporateredundantdata.Otherresearchersalsoclassifiedcodesmellsindifferentways.Onewayistoclassify the code smellsinto two categories:smellswithin a class(*e.g.*, aslong method,switchstatement,commentsandcodeduplication)andsmellsoutsidetheclass(*e.g.*,asdataclassandmiddleman*etc.*,)[6].

Avariety ofsoftware toolshavebeendeveloped forthe automateddetectionofbadsmellsandtheydifferintheircapabilitiesandapproaches.Determiningwhethersomepieceof codecontainsbadsmell(s)issomewhatsubjectiveandstillthereisalackofstandards.

Inthiswork,acomparativestudyiscarriedoutregardingtwobadsmelldetectiontoolsnamelyJDeodorantandinCode.The irdetectionmethodologyisdiscussedingreaterdetailandvariationsinresultsarenoted.WeselectedFeatureEnvyandGo dclasscodesmellstodoworkwith.Bothtoolsareevaluatedonthesetwosmells.

II. Literature Review

Steve*etal.*,[8]describedanalysisofcodesmells.Theyusedbadsmelltaxonomydescribedin[7,9]andabespokesoftwaretool[10]tofindnumberofrefactoringrequiredforeachof22badsmells.Accordingtohisanalysis,Bloatersisthesmellcategorythatneedsmosteffortbecauseitrequireslargernumberofrefactoringtoremove from code. The Change Preventers needs least effortbecause it requires lowestaverage number of refactoring to eradicate from code. They claimed that bad smells aredeceptive innaturemean that they hidethe real effortthatwouldberequiredtoremovethat smell when nested refactoring are involved. Bad smells that requires most effort toremovearebestunderstoodbydevelopersand,conversely,badsmellsrequireslesseffortoremoveareleastunderstood bydevelopers.

In[11]Foutse*etal.*,presentedthatclasseswithcodesmellsaremorechangepronethen the classes withoutcode smells. They furthershowed thatthe correlation betweenparticulartypeofcodesmellandchangeproneness.TheyusedDECORE[12]approachtodetectcodesmellsandapplieddifferentmathematicaltechniquestoconcludetheresults.Twosystemsfromdifferentdomainswereselectedforexperimentation.BothwereopensourcesystemsnamelyAzureusandEclipse[11].From13releasesofEclipseand9releasesofAzureustheyshowedthatclasseswithbadsmellsaremorechangepronethen theclasseswithoutbadsmells.They furtherprovedthataparticularkindofbadsmellsleadtochange-proneness.Howeverthisstudydoesnotconcernthetypeandamountofchangeanditislimitedtoonlytwosystems.Ifitisgeneralizedtomoresystemsitmaygivedifferentoutcomes.

Evolutionofthreebadsmellsontwoopensourcesystemwasstudiedin[19].ConsecutiveversionofjFlexandjFreechartweretakenintoaccounttoperformtheanalysis.ThebadsmellsnamelyLongmethod,FeatureenvyandStatecheckingwereextracted using JDeodorant. The result showed that the number of bad smells in systemincreaseasthesystemevolvesoverthetime;manyofthebadsmellsremaininthecodeup tolatestversionsof thesystem;someof thebadsmells that areremovedwerenotaresultoftargetedrefactoringapplicationrathertheyareremovedinaneffortofadaptivemaintenance.

Fontana*etal.*,[20]presentedareportononsmelldetectiontools.Theystatedresultofsix bad smelldetection tools; experimentationwasperformed on 5versionsofGanttProject [21]. As none of the bad smell shared by all tools, that’s why they perform theiranalysisonasubsetofthetool.Onthebasisoftheirexperimentsandresultstheyconcluded that;main criticalaspect is to have no access todetection rules andmetricsthresholds.They proposedthatbadsmelldetectiontoolsshouldhavethepossibility tosetandchangethemetricsthreshold.

1. Experimental Work

WeperformedouranalysisonGUIbasedmultimediaapplicationusingtwocodesmelldetectiontools.Thescenarioofexperimentationisgiveninsubsequentparagraphs.

1.1. SelectionofBadSmellDetectionTools

Avarietyofsmelldetectiontoolsareavailableforcodesmellsdetection.Belowisthedescriptionofthosetoolsthatwehave used inourcomparativestudy. Thesetoolscanapplicableonlyforjavacode.

1.1.1. JDeodorant:JDeodorant[15,16]isaneclipsepluginthatissofaridentifiesfourkindsofbadsmells,namely"FeatureEnvy”,"TypeChecking”,“LongMethod”and“Godclass”.ThistoolusestheASTParserAPIofeclipseJavaDevelopmentToolto detect bad smell from the source code and ASTRewrite API to apply refactoring. Inadditiontodetectsmellsandapplyrefactoring,JDeodorantencompassesmanyotherfeaturesincludingtransformation ofexpertknowledge tofullyautomated processes,preevaluationofeffectforeachsuggestedsolution,userguidanceincomprehendingthedesignproblemsanduserfriendliness[14].CurrentlyanewfeaturehasbeenembeddedinJDeodorant,whichisexecutionofthistoolinbatchprocessingmode[13].Inthismode,nointeractionwitheclipseinterfaceisrequired.RatherallopenJavaprojectscanbeanalyzedwithoutuserinteraction[2,5].

1.1.2. inCode:inCodeisalsoaneclipseplug-inforsmellsdetection.Themainfeatureofthistoolistosupportprogrammerstoprogramincodesmellsawareprogrammingenvironment.Itworksinthebackgroundofeclipse.Duringprogramming,ifprogrammerwritesanybadstructure,thanit showsthesesmellsas,“eclipseshowerror”andwarningsintheshapeofredcolorblocksalongwithcode.inCodedetects4bad smells *i.e*. Feature envy, God class, Duplicate code and Data class. These bad smelldetectionsarebasedonobject-orientedmetrics.Thegreaterpartisthatuserdoesn’thavetointeractwiththeunderlyingmetricsdirectly.Metricsdotheirjo bunderthehood,whileyouzipdirectlytotheusefulconclusions.Moreover,italsohasfunctionalitiesofconstructionofMetricsPyramid[17].Thepyramidshowsthevaluesofdifferentmetricsandtheircorrespondencewithinthesoftware.

1.2. Subject System

XtremeMediaPlayer[18]isafreecross-platformmediaplayer,licensedunderGPLv2.Itsupports differentaudio formatsandplaylistformats.Italsoprovide remotemusicplaybackviaURL.Itcanreadgeneraltaginfoandspecifictaginfodependingontheaudioformat.Attractiveinterfacehavingdifferentplayingmodesand

Multilanguagesupportisprovided.Playlistmanagerandkeyboardshortcutsarealsoprovided.Usercanupdatetheapplicationautomatically.
Itsfirstversionwas0.5.1releasedonMarch27,2008afterthatmanyversionshavebeen released.Latestversion is0.7.0released on September 12,2011. Theversionwehaveselectedforourcasestudyisxtrememp-0.6.3releasedonJanuary,31,2009andxtrememp-0.6.6releasedonOctober23,2010[18].
MetricsforbothversionofXtremeMPisgiveninTable1.ThesemetricsarecalculatedwithatoolnamelyMetrics[23].

Table1.Xtreme-MPProjectMetrics

Metrics	XtremeMP-0.6.3	XremeMP-0.6.6
Totallinesofcode	6797	7956
NumberofPackages	13	17
NumberofClasses	63	71
Numberofinterfaces	4	8
NumberofMethods	459	498
Methodlinesofcode	4146	5142
Weightedmethodsperclass	1142	1341
Numberofstaticmethods	60	70

1.3. SelectionofBadSmells

Manyresearchgroupsareworkingonthedevelopmentofautomatedrefactoringtools.Currently, differentrefactoring toolsareavailablewhich coverdifferentaspectofbadsmells.Therearealmost93refactoringavailableintheliterature.Though,manytoolsprovide someoftheserefactoring,noneofthetoolprovidesalltheserefactoring.
Weselectedtworefactoringtools(detailsgivenintheIastsection).Thesetoolscovermultiplebadsmellsdetection,butinthescopeofourstudy,wehaveselectedonly bad smells named, “Feature Envy” and “God Class”. Short description ofboth badsmellsisgiveninlaterparagraph.

1.3.1. FeatureEnvy: FeatureEnvyisaviolationoftheprincipleofobjectorientedaboutgroupingbehaviorwithrelateddata.“Itoccurswhenamethodismoreinterestedin aclass other than theone it actually is in”. [1] Most importantcures to eradicate badsmellsareMoveMethodandMoveFieldrefactoring.[1]

1.3.2. GodClass: GodClassisaviolationoftheprincipleofobjectorientedthataclassshouldimplementonlyoneconcept[22].Itisoftentryingtodomuch;itoftenshowsupastomany instancevariables[1].Duetoimportantrole in thesystem,manyothersystemclassescanbedependentonit.AsconsequencechangesinGodClassduringsoftwaremaintenanceandevolutioncanleaddeveloperstoproblems.TherefactoringthatusetoremovebadsmellareExtractClass,ExtractSubclass,ExtractInterfaceandDuplicateObservedData[1].

2. Results Findings

Experimentationwasperformedwithdefaultthresholdvalues.DetectionresultsarewritteninTable2.

Table2.#ofFoundSmellswithSelected ToolsinbothversionsofXtremeMP

Tool	FeatureEnvy		GodClass	
	Ver.0.6.3	Ver.0.6.6	Ver.0.6.3	Ver.0.6.6
JDeodorant	1	3	14	16
InCode	2	2	2	2

2.1. Feature EnvyD iscussion

Onexaminingtheresults,wefoundthattheinstancesoffeatureenvybadsmellsdetectedbybothtoolsareentirelydifferent. FeatureEnvysmellsdetectedbyInCodearestaticmethods,theydonotmanipulateanydataoftheirsourceclassbuttheyprocess data of other system classes. According to object oriented design heuristics andprinciples;methodmustbeplacedintheclass,whichdataitmanipulatesmore.ThatisthereasonInCodedetectsthesemethodsasbadsmell.

TheJDeodorantdoesnotdetectthesestaticmethodsasbad/codesmell.TheJDeodorantdetectsthreenonstaticmethodsas featureenvybadsmells.Butthesedetectedbadsmellsarenotpresentintheresultsof InCode.

Thenonstaticmethodsdetectedasfeatureenvybadsmellby JDeodorant;useattributes(datamember)ofsourceclassbuttheseattributesareofreferencetype.Typeofthese attribute isother system classes.So,thesemethodswith the help ofreferencetypedattributesmanipulatedataofothersystemclasses.

ThatisthereasonJDeodorantidentifiesthesemethodsusingitsdistancebasedapproach.Inwhich,itmeasurethedistancebetweenthemethodandtheclass,expressing thedissimilaritybetweenthesetofentities(methodandattributes)accessedbymethodandsetofentitiesbelongingtoclass.

TheInCode does not detect thesemethods as feature envy bad smell.Here a questionarisesfromtheseresultsofInCodeforfeatureenvy;whicharegivenbelow:

DoesInCodedetectnonstaticmethodsasfeatureenvysmellornot?

Currentresultsdonotgivemuchinformationaboutthisquestion.WemakeaseparateversionofthisAPIbyapplyingtheGodClassrefactoringontheXtreme-MP 0.6.6withJDeodorant.BydetectingFeatureEnvy badsmell from the refactoredversionofAPI withInCode, wegetsomeresultswhich somehowgivesomesatisfactory resultsabout this question. By these result we found thatInCode detects non-static methods asfeature Envy bad smell. Due to no access to InCode's documentation for its bad smelldetectionrulesandmetricsthreshold;weareunabletounderstandthereasonwhyInCodedoesnotdetectnon-staticmethodsdetectedbyJDeodorant.

2.1.1. ConcludingRemarksonFeatureEnvy'sResults:

AstheMartinFowlerwroteinhisbook:"Methodmustbeonthatobjectwhichdataituses."

Aswe see thisguideline, thenweget thought of only non-staticmethods as featureenvy;becauseonlynon-staticmethodsareontheobject.Thestaticmethodsarenotassociatedwithanyparticularobject,becausetheyareshareable behaviorsofallobjects. To access them we do not need object of the class in which it resides. They areaccessedwiththeirclassnamesinthesystem.

No doubt theguidelineof Martin, hassense that consider only non-staticmethods asfeatureenvy.Butifweseethemainphilosophybehindthiswhichistopreventtherippleeffectsofchangesinsoftwaresystem.Toimplementthisphilosophyinsystemdesign, wegroupedmethodsto thatclasswhich data itmostuses.Due to thisreason,staticandnon–staticmethodsareequallyimportant.

So,asJDeodorantdetectsveryefficientlynon-staticasfeatureenvy.Itmustincorporatethenon-staticmethodsalsointhecategoryoffeatureenvy.TheInCodealsoneedsimprovementsinitsmethodologytodetectfeatureenvysmell.Inthecasewhenthenon-staticmethodsusethereferencetypefields(attributes)toaccessthedataofothersystemclasses.

At last none of both has concrete process to detect feature envy bad smell. Both havedeficienciesintheiralgorithms.Themissingconceptsofbothareimportant;iftheyconsiderthemtheycanimprovethequalityofsoftware.

2.2. GodClass Discussion

Byevaluatingtheresultofbothtoolswefoundlargedifferencebetweentheirresults.TheInCodedetectstwoandJDeodorantdetectfourteenGodClassesfromthebothversionsofXtreme-MP.

Onfurtherindepthanalysiswefoundthatbothtoolshaveentirelydifferentapproachestoidentify GodClasses.

TheGodClassesdentifiedbyJDeodorantalsohasthosetwoGodClasseswhichidentifiedbyInCode.ButtheyranksomeotherdetectedGodClasssmellsmoreimportantforthedesignqualityoftheexaminedprojectwiththehelpofEntityPlacementMetrics[23].

The InCode use the metrics based approach to detect God class smells. It checks theencapsulationthroughcohesionandcouplingmetricsandthecumulativecomplexityofthemethodsoftheclasstoidentifytheGodClasssmells.

TheJDeodorantusestheagglomerativeclusteringtechnique.Inwhichitextractinformationfromthesourcecodeoftheexaminedprojecttocalculatethedistancebetweentheentities(methodsandattributes)oftheclassesandthenapplythecluste

ringalgorithmtoidentifythecoherentgroupofentitiesastheopportunitiesasGodClass[13].
A large difference exists in the scope andgoal of the approaches used by both tools.InCodegoalisto identify suchsystemclassesasGodClasswhich haslack ofcohesionandhighcomplexity.TorefactortheGodClassesdetectedbyInCodeweneedsomefurtherprocessingtodeterminewherewecanapplywhichrefactoring.Forthisweneedtodetermine;thisGodClassiseitherDataGodClassorbehavioralGodClass.AfterclassificationweapplyoneoftherefactoringfromMoveFieldRefactoring,Movemethodrefactoring,ExtractclassRefactoring.
ButallthesethingsdescribedinaboveparagraphcannotbeattainedautomaticallyfromthecurrentmethodologyusedbyInCode.ForthiswemustneedsomemanualinteractionindetectionandclassificationofGodClass,whenweusingInCodetodetectandrefactorGodClasses.
GoalofthemethodologyusedbyJDeodorantforGodClassistoidentifytheopportunitiesofextractclassrefactoring.InthemechanismandwayofrepresentationoftheresultoftheGodclasssmelldetectiontosoftwareengineerswithEntityPlacementMetricsisbest.
ButtheextractclassrefactoringisnotthesolutioninallcasesandconditionsfortheGodClass.
WhileperformingexperimentationforfindingFeatureenvysmellthrough InCode,wefoundaninterestingresultaboutGodclass.InthisexperimentwerefactorsingleGodClasssmell(xtrememp.XtremeMP)withJDeodorantbyextracting aclass(Xtreme-MP.XtremeMPProduct).AfterthisrefactoringwhenweagaindetectGodClassbadsmellformthisrefactoredversionofAPIwithbothtools;theydetectnogodclass.On thebasisofthisfindingonepointisclearerthatInCodedetectsonlytwoclassesasGodClassbuttheyaretheonesthathavegreatimpactonqualityofcode.

2.2.1. ConcludingRemarksonGodClass'sResults: Bothtoolsdonotprovidecomprehensivesolution.ThereisaneedofsuchapproacheswhichidentifytheGodClassesandthenidentifyopportunitiesofalltypesofrefactoringsolutionsthatproposedbyauthorofrefactoring[1]forGodClassindifferentcasesofGodClass.

3. ConclusionandFutureWork

Inthisworkweanalyzetwocodesmelldetectiontoolsandcomparisonispresented.Weevaluatedwhytoolsdepictvariationinresultsandwhatspecificparameterstheyusedfordetectingaparticularsmell.
Wefound thatbasic difference in results liesdue to useofdifferentapproaches forsmelldetection.Bothtoolshavetheirownefficienciesanddeficiencies,whichweendeavoredtocomprehendinthiswork.Onthebasisofourfindingwesaythatthough,thereislackoftoolsupportincontextofsmelldetectionandrefactoring,availabletoolsarenotmatureenough.Besidesdevelopmentofnew toolscurrent toolsalsoneedtoberevised.
Inthefuture,weareinterestedininvestigatingthechangepronenessofcodehavingGodClassandFeatureEnvybadsmellsandimpactofthischangepronenessonmaintainabilityofsoftware.Aprototypeisinprogresswhichwillprovidestate-of-the-artsolutionoftheproblemsidentifiedinthisstudyandinpriorresearch.

References

[1]. M.Fowler,K.Beck,J.Brant,W.OpdykeandD.Roberts,"Refactoring:ImprovingtheDesignofExistingCode", Addison-Wesley,**(1999)**.
[2]. M.Munro,"ProductMetricsforAutomaticIdentificationof"BadSmell"DesignProblemsinJavaSource-Code", SoftwareMetrics,11thIEEEInternationalSymposium,**(2005)**,pp.15.
[3]. K.Beck,"ExtremeProgrammingExplained:EmbraceChange", Addison-Wesley.ISBN978-0-321-27865-4, **(1999)**.
[4]. T. Mens and T. Tourw´e, "A survey of software refactoring", IEEE Transactions on software engineering,vol.30,no.2,**(2004)**,pp.126-139.
[5]. N. Roperia, "JSmell: A Bad Smell detection tool for Java systems", Long Beach, CA, USA: ProQuestDissertation andTheses,**(2009)**.
[6]. CodingHorror:CodeSmells.**(2012)**http://www.codinghorror.com/blog/2006/05/code-smells.html[Accessed at:,**(2012)** March4]
[7]. M. Mantyla, J. Vanhanen and C. Lassenius, "A Taxonomy and Initial Empirical Study of Bad Smells inCode. **(2003)**",ProceedingsoftheInternationalConferenceon SoftwareMaintenance,pp.381.
[8]. S. Counsell, H. Hamza and R. M. Hierons, "An Empirical Investigation of Code Smell 'Deception' andResearchContextualisationthroughPaul'sCriteria",**(2010)**JournalofComputingandInformationTechnology-CIT18,**(2010)**,vol.4,March4,pp.333-340.
[9]. M. Mantyla, "Bad Smells in Software — A Taxonomy and an Empirical Study", Master's Thesis, HelsinkiUniversityofTechnology,SoftwareBusiness andEngineeringInstitute, **(2003)**.
[10]. H. Hamza, S. Counsell, G. Loizou and T. Hall, "Code Smell Eradication and Associated Refactoring",ProceedingsoftheEuropeanComputingConference(ECC),Malta, **(2008)**September.
[11]. F.Khomh,M. D. PentaandY. Guéhéneuc,"AnExploratory Study of theImpactof Code SmellsonSoftwareChangeproneness",Proceedingsofthe16thWorkingConferenceonReverseEngineering(WCRE), Lille,France,IEEEComputer SocietyPress,**(2009)**October 13-16.
[12]. N. Moha, Y. Guehenue, L. Duchien and F. L. Meur, "DECORE: A method for specification and detection ofcode and design smells", Software Engineering, IEEE Transactions, vol. 36, no. 1, **(2010)** January-February,pp.20-36.
[13]. Jdeodorant,http://jdeodorant.com/[Accessed: April.17,2012].
[14]. Eclipse Marketplace," online Available: http://marketplace.eclipse.org/content/jdeodorant [Accessed: April17,2012].
[15]. M. Fokaefs, N. Tsantalis and A. Chatzigeorgiou, "JDeodorant: Identification and Removal of Feature EnvyBadSmells",

IEEEInternationalConferenceon SoftwareMaintenance, **(2007)**October,pp. 519-520.
[16]. T.Chaikalis,N.TsantalisandA.Chatzigeorgiou,"JDeodorant:IdentificationandRemovalofType-Checking Bad Smells", 12th European Conference on Software Maintenance and Reengineering, **(2008)**, pp.329-331.
[17]. InCodewebsiteproblemdetection on fly[April.17,2012].
[18]. Sourceforge,http://xtrememp.sourceforge.net/[Accessed:May.13,2012].
[19]. A. Chatzigeorgiou and A. Manakos, "Investigating the Evolution of Bad Smells in Object-Oriented Code",QualityofInformationandCommunicationsTechnology(QUATIC),2010SeventhInternationalConference, pp.106-115,**(2010)**September 29-October2.
[20]. F. Fontana, E. Mariani, A. Mornioli, R. Sormani and A. Tonello, "An Experience Report on Using CodeSmells Detection Tools", ICSTW '11: Proceedings of the 2011 IEEE Fourth International Conference onSoftwareTesting,VerificationandValidationWorkshops,**(2011)**.
[21]. GanttProject,Available:http://www.ganttproject.biz/[Accessed:April12,2012].
[22]. R. C. Martin, "Agile Software Development: Principles", Patterns and Practices Prentice Hall, Upper SaddleRiver, NJ,**(2003)**.
[23]. Metrics1.3.6",http://metrics.sourceforge.net/,[July15,201].