ISSN 2347-3983

International Journal of Emerging Trends in Engineering Research, Vol.4. No.10, Pages : 188-193 (2016)
Special Issue of ICACSSE 2016 - Held on September 30, 2016 in St. Ann's College of Engineering & Technology, Chirala, AP, India
http://www.warse.org/IJETER/static/pdf/Issue/icacsse2016sp34.pdf

NEW METRICS FOR SYSTEM MODIFIAILITY OF INHERITANCE HIERARCHIES

Mr.P.S.Naveen Kumar,G.Subba Rao and Mr.D.N.V.Syam Kumar*
Asst.Professor,Dept.of MCA,St.ann's College of Engineering & Technology,Chirala,,Ap.,India
Asst.Professor,Dept.of MCA,St.ann's College of Engineering & Technology,Chirala,,Ap.,India
*Assoc.Professor,Dept.of CSE,St.ann's College of Engineering & Technology,Chirala,,Ap.,India

ABSTRACT

Modifiability is the term stands for the how much amount of the information would be modified in the class and system level of the object-oriented system with inheritance hierarchies. Previously there is some metrics were existed for modification of the system in the issue of maintainability. Existed metrics were giving the more complexity values means time taken for the modification of the system is more. In this paper we are proposing new metrics for the modifiability at class and system levels. Our proposed metrics were evaluated through the weyker's properties. These proposed modifiability metrics gives the lowest complexity values when comparing with previous existed metrics. The manager of the system takes less time to modify the class and entire system for easy maintenance of the system.

KEYWORDS: software metrics, object-oriented, inheritance hierarchy, DAG, system modifiability, system maintenance, weyker's properties.

INTRODUCTION

A metric would be needed for measurement of the given software program either it may be traditional program or object oriented program ….etc. In the measurement of the inheritance hierarchy several object –oriented metrics were existed. Several studies were existed [1, 2] on software metrics for improving the software quality. Several programming languages and fields [3] were utilized the object-oriented metrics in effective manner. In object-oriented techniques inheritance causes to reduce the redundancy and system maintenance to improve the efficiency of the system[17,18,19,20,21,22]proved by many researchers in their researches. In the inheritance hierarchy several object-oriented inheritance metrics were existed [18,4,5,6,7,8,24].

The software metric used to measure the software program have to show its mathematical and theoretical background by fulfilling the some of the well-known properties. For developing the good software metrics weyker's[9] proposed nine properties which have to be satisfied by every proposed metric. The weyker's properties are evaluated by different developers [18, 4, 10, 11, 12, 13, 23, 25] against their proposed inheritance metrics. In these weyker's properties most of the properties were satisfied by the well-known inheritance metrics developed by the developers. Some of the weyker's properties were not satisfied by these well-known properties also because that metrics were utilized on the traditional programming [23]. Most of the object oriented metrics were use the classes only not the inside data of the classes. Hence many of the object-oriented metrics were not suited for some of the weyker's properties.

This paper is organized in following manner. Next topic covers the literature survey for the proposed metrics. In this we were mentioned so many well-known metrics like DIT,NOC,NAC,NDC ,AID and AM. Our proposed metrics were ACM and ASM were discussed in detail in the topc3. In topic4 weyler's properties were tested with our new metrics. Previous existed metrics comparison with our proposed metrics was done in the topic5. The results comparing with AM and ASM were placed in topic6. The finalized conclusion and future scope of this paper was written in the topic and topic 8 respectively.

LITERATURE SURVEY

In this literature survey we were discussed well known inheritance metrics likely DIT, NAC, NOC, NDC, AID and modifiability metrics of the system is AM in detail manner.

Depth of Inheritance Tree (DIT) developed by chidember-kerner [14,15,18] state that maximum depth from the root node to the present node. Here in this DIT technique ambiguity may be raised in many situations. To solve this problem W.Li[6] proposes the new metric called Number of Ancestor

Classes(NAC) measure the number of classes inherited by the individual class in the object-oriented inheritance hierarchy.

Chidember-kerner [14, 15, 18] proposes another metric named Number of Childs(NOC). W.Li [6] proposes new metric called Number Descendent Classes (NDC) to consider total number of sub classed in to the account.

The Average Depth of Inheritance (AID) metric was developed by Henderson-sellers[5] for applying the average complexity values in the DIT metric. Sheldon-jerath [8] proposed metric called Average Modifiability [AM] for system modifiability by considering the understandability and successors of the individual classes.

NEW INHERITANCE METRICS

Our proposed metrics in this paper are Average Class Modifiability (ACM) and Average System Modifiability (ASM). In our representations of the inheritance hierarchies two more diagrams were utilized. Figure1 involves with multiple inheritance situation. So, figure1 was represented with Directed Acyclic Graph (DAG) with no loops [16]. Figure2 was represented with the normal tree hierarchies.

In the complexity values identification preferred way is average case of representation. Coming to our metrics modification consider the sub classes of the specified class and that specified class also for modification. In the best case of modification all the classes information has to be modified. In the worst case no one class has to be modified. If we consider the Average case of half of the total class information has to be modified.

ACM states that the modifiability of the given class is the number of subclasses by adding the given class division by 2. That gives average modification value for the given class. ASM states that modifiability of the given system is the sum of Average Class Modifiability (ACM) of the individual classes with the number of classes in the system.

Average Class Modifiability is
ACM = (Number of Sub Classes +1) / 2
Average System Modifiability is
ASM = $\sum_{i=1}^{n} ACM_i$ / n
ACM_i = Average Class Modifiability of the Class i.
n= Number of classes.
Applying the above metrics on figure1

ACM(H)=3 ACM(G)=3.5 ACM(F)=2.5
ACM(E)=1 ACM(D)=0.5 ACM(C) =1.5
ACM(B)=ACM(A)=0.5
ASM of figure1 = 1.62
Applying the above metrics on figure2
ACM (P) =3.5 ACM (Q) =ACM(R) =1.5
ACM(S) =ACM (T) =ACM (U) =ACM (V)=0.5
ASU of figure2 = 1.21.

PROPOSED METRICS EVALUATION WITH WEYUKER'S PROPERTIES

The statistical evaluation of the software metrics can be done against the satisfaction of the weyker's[9] properties. This may leads to good metrics for measuring the system quality in better way. Even some of research persons criticizing the weyker's properties these properties judge the software metrics for effective system maintenance and quality features. The most object-oriented well known metrics not satisfies the some of the weyker's properties (5, 7, 9) [23].

Property-1:- Non-Coarseness –

For example class A and class B are having the proposed metric M the Non-Coarseness found that M (A) ≠ M (B).

The figure1 state that ACM of class H is different from ACM of class G. Here ACM (H) =3 and ACM (G) =3.5 .So ACM(H) ≠ACM (G).consider the figure2 ACM of class P is different from ACM of class Q. Here ACM (P) =3.5 and ACM (Q) =1.5. It means ACM (P) ≠ACM (Q). Hence ACM metric satisfies the weyker's property-1.The figure1 and figure2 poses two different ASM values for the both of the figures. ACM value of figure1 is 1.62 is different from the figure2 ACM value 1.21. It means that ASM metric satisfies the weyker's property-1.

Hence our proposed metrics ACM and ASM were satisfied the weyker's first property Non-Coarseness successfully.

Property-2:- Granularity –

It means that there is a metric value for the finite number of programs.Ccomplexity value given by the number of programs. Non-Negative value must be taken for the complexity value.

Every object-oriented metric represented with class level hierarchy must be satisfied this property [4] because every object-oriented inheritance hierarchy must having the class levels , that metric satisfies this property. Our proposed metrics also follows the inheritance hierarchy with class levels. So, our proposed ACM and ASM were satisfied the weyker's second property.

Property-3:- Non-Uniqueness-

The proposed metric M value must be same for the two different classes A and B. it means M(A)=M(B).Consider the figure1 ACM value of class D is same as ACM values of class B. Here ACM(D)=0.5 and ACM(B)=0.5. . It means that ACM(D)=ACM(B). Consider the figure2 ACM value of the class Q is same as the class R. Here ACM(Q)=1.5 and ACM(R)=1.5.So ACM(Q)=ACM(R). Hence ACM was satisfies the third property.

At the system level consider the another figure ASM values with the figure1. we found that both the figurse shows the similar ASM value is1.6..Hence ASM was satisfies the weyker's third property Non-Uniqueness successfully.

Property-4 :- Design Implementation-

If two designers design the same class it has to show the two different metric values. The designed class must be utilized in the proposed metric. Our proposed metrics ACM and ASM were satisfied the weyker's fouth property because if two designers develop the same class of the same program they may follow the different inheritance hierarchy and different class levels. Hence the designs of the systems would be different. Our metrics ACM and ASM also follows the different designs for different designers. Hence ACM and ASM metrics were satisfied the weyker's fourth property successfully.

Property-5:- Monotonicity –

The metric value of the combination of two different metric valued classes is greater than or equal to the given individual classes. Suppose A and B classes are having the two different metric values ,the combination of both denoted as A+B metric value is greater than or equal to the individual A and B classes metric values. It means that M(A+B)≥M(A) and M(A+B)≥M(B).

In object-oriented inheritance hierarchy every metric has to fulfill the three possible cases for satisfying the weyler's monotonicity property.

1. If class A and class B are siblings.

Consider the figure1(b) shows that ACM(A)=0.5 and ACM(B)=0.5.if we combine the both siblings into one as A+B the finalized metric of the ACM(A+B)=0.5 , which is equal to the both the metrics of A and B. Consider figure2(b) shows that ACM(Q)=1.5 and ACM(R)=1.5. If we can combine the both siblings into one as Q+R, the finalized result of the metric of the ACM (Q+R)=2.5 which is greater than both the metrics of classes Q and R individually.

Hence case-1 was successfully satisfied by the ACM metric.

2. If one class is the child of another class.

Consider the figure 1(c) gives the ACM(C)=1.5 and ACM(B)=0.5.The Combination of the C+B gives metric value as ACM(C+B)= 1 which is equal to class E and less than class C. Consider the figure2(c) gives the values of the metric ACM(Q)=1.5 and ACM(S)=0.5 . The combination of the Q+S gives the value of metric as ACM (Q+S)=1,which is greater than the metric value of S and less than Q.

Hence case-2 was not satisfied by our proposed metric ACM, because the combination of the one class is the child of another class both the classes are combined and treated as one unit. So ACM is not satisfied the monotonicity property. The well-known inheritance metrics like DIT,NAC,NDC,AID and AM also not satisfying the weyker's fifth property[23] because they were also focused on the class only not the inside matter of the class.

3. If class A and class B are neither siblings nor children of each other.

Consider the figure 1(d) shows that the ACM (C) =1.5 and ACM (E) =1. The combination of C+E gives the metric value ACM(C+E) =1.5, which is equal to class C metric value and greater than class E metric value. Consider the figure 2(d) shows that ACM (Q) =1.5 and ACM (V) =0.5. The combination of both the classes as (Q+V) metric value is ACM (Q+V) =1.5, which is equal to the class Q metric value and the greater than the class V metric value.

Hence case-3 is satisfied by ACM metric successfully.

ISSN 2347-3983

International Journal of Emerging Trends in Engineering Research, Vol.4. No.10, Pages : 188-193 (2016)
Special Issue of ICACSSE 2016 - Held on September 30, 2016 in St. Ann's College of Engineering & Technology, Chirala, AP, India
http://www.warse.org/IJETER/static/pdf/Issue/icacsse2016sp34.pdf

Only one case of weyker's monotonicity property was not satisfied by our proposed metrics. Hence our proposed ACM and ASM metrics were not satisfies this property similar to the evaluation of the well known inheritance metrics like DIT, NAC, NDC, AID and AM.

Property-6:- Non-Equivalence of Interaction-

If suppose class A and class B shows the same metric value combine these individual classes with another class C the finalized metric values of A+C not equal to the metric values of B+C.

Consider the figure1 gives the metric values of ACM(B)=ACM(D)=0.5. Here combining these classes with another class C as displayed in figure1(e) and 1(f) would give the finalized metric values as ACM(B+C)=1 and ACM(D+C)=1.5 both are not equal. So ACM(B+C)≠ACM(D+C). Consider the figure2 gives the metric values of ACM (Q) = ACM (R) =1.5. Here combining these classes with class T as displayed in figure2 (e) and figure2 (f) would give the finalized metric values of ACM (Q+T)=1 and ACM(R+T)=1.5 both are not equal. So ACM (Q+T) ≠ACM(R+T). Hence our proposed metric ACM was satisfies the weyker's sixth property. At the system level modifiability metric ASM also satisfies the weyker's sixth property.

Property-7:- Significance of Permutation-

If program B is formed with program A's permuting order of statements then the measured metrics for the individual metric values were not equal.

This metric was suitable for traditional programming where the inside matter of the program taken major place in the selection of the metrics. This metric not suitable to most of the object-oriented metrics because these metrics were consider the class as single unit not the inside data of the class. Well-know metrics like DIT, NOC, NAC, NDC, AID and AM also not satisfies this property [23]. Hence our proposed metrics ACM and ASM also not supporting the weyker's seventh property.

Property-8:- No Change of Remaining-

If the class name is renamed with another name the metric values of the classes need not changed with the previous values.

This property must be satisfied by every object-oriented inheritance metric because changing of the name of the class not shown the effect on the metric value. Here our proposed metrics were also object-oriented inheritance hierarchy metrics. Hence our proposed ACM and ASM metrics were satisfied the weyker's eighth property successfully.

Property-9:- Interaction complexity-

Suppose two classes A and B combination denoted as A+B metric value is greater than the summation of the individual classes A and B. It means

$$M(A+B)>M(A)+M(B)$$

Our proposed metrics not supported the weyker's ninth property because these metrics follows the object-oriented design [23]. Hence our proposed metrics ACM and ASM were also not satisfying the weyker's ninth property.

METRICS COMPARISION

In this topic we were evaluated our proposed ACM and ASM metrics against weyker's properties and compare with the well-known metrics DIT,NOC,NAC,NDC,AID and AM. For the DIT, NOC, NAC, NDC and AID metrics previously existed results were taken with respect to the weyker's properties. Weyker's properties were tested on the proposed metrics and results were displayed in below table

Property	DIT	NOC	NAC	NDC	AID	AM	ACM	ASM
1	√	√	√	√	√	√	√	√
2	√	√	√	√	√	√	√	√
3	√	√	√	√	√	√	√	√
4	√	√	√	√	√	√	√	√
5	×	√	×	×	×	×	×	×
6	√	√	√	√	√	√	√	√
7	×	×	×	×	×	×	×	×
8	√	√	√	√	√	√	√	√
9	×	×	×	×	×	×	×	×

Table1: Measurement of Inheritance Metrics in view of Weyker's properties.

√ - weyker's property satisfied by the metric.
× - weyker's property not satisfied by the metric.

From the above table we may understand that most of the inheritance metrics which were focused on the classes only not the inside information of the class ,that metrics need not to satisfy all the properties suggested by the weyker's .

RESULTS

International Journal of Emerging Trends in Engineering Research, Vol.4. No.10, Pages : 188-193 (2016)
Special Issue of ICACSSE 2016 - Held on September 30, 2016 in St. Ann's College of Engineering & Technology, Chirala, AP, India
http://www.warse.org/IJETER/static/pdf/Issue/icacsse2016sp34.pdf

In this topic our main focus is to reduce average modifiability of the system when comparing with previously existed metric AM. In this regard we were got the very much reduced results than the AM results. The finalized results of the Average system modifiability of our proposed metrics ASM by comparing with the AM

Figure	AM	ASM
1	4.37	1.62
2	3.1	1.21

Table2: Modifiability complexity values at system level for figure 1&2.

CONCLUSION

In this paper we were mentioned various inheritance metrics and discussed their existence with the weyker's properties. Here our man concentration is on the modifiability of the class with in minimum amount of time. For this reason we want to reduce the average system modifiability complexity value by comparing with existed metric AM. In this connection we were drastically reduced the finalized values of the average system modifiability complexity values and compared them with AM values.

FUTURE WORK

Our proposed metrics were got the good results rather than the previously existed metrics for average modifiability. Yet we were not concentrated on the inside data of the class. In future we want to focus on the class data and methods and study their behavior on the class and system level modifiability.

REFERENCES

[1] Amjan Shaik,C. R. K. Reddy, Bala Manda, Prakashini. C, Deepthi. K Metrics for Object Oriented Design Software Systems: A Survey Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS) 1 (2): 190-198.
[2] M.S.Ranwat,A.Mittal,S.K.Dubey Survey on impact of software metrics on software quality (IJACSA)International journal of Advanced Computer Science and Applications, Vol.3,No.1,2012.
[3] Darcy, D.P.—Kemerer, C. F.: OO Metrics in Practice. IEEE Softw. 22,6 November 2005, pp. 17–19. DOI: http://dx.doi.org/10.1109/MS.2005.
[4] Abreu, F.B.—Carapuca, R.: Candidate Metrics for Object-Oriented Software within a Taxonomy Framework. Journal of System Software, Vol. 26, 1994,pp.87–96
[5] Henderson-Sellers, B.: Object Oriented Metrics: Measures of Complexity. Pren-tice Hall PTR: Englewood Cliffs, NJ, 1996; pp. 130–132.
[6] Li, W.: Another Metric Suite for Object-Oriented Programming. Journal of Systems and Software, Vol. 44, 1998, pp. 155–162.
[7] Lorenz, M.—Kidd, J.: Object-Oriented Software Metrics. Prentice Hall 1994,ISBN: 013179292X.
[8] Sheldon, F.T.—Jerath, K.—Chung, H.: Metrics for Maintainability of Class In-heritance Hierarchies. Journal of Software Maintenance 14, 3 May 2002, pp. 147–160.
[9] Weyuker, E. J.: Evaluating Software Complexity Measures. IEEE Transactions on Software Engineering, Vol. 14, 1988, No. 9, pp. 1357–1365.
[10] Cherniavsky, J.—Smith, C.: OnWeyukers Axioms for Software Complexity Mea-sures. IEEE Transaction on Software Engineering, Vol. 17, 1991, No. 6, pp. 636–638.
[11] Gursaran, G.R.: On the Applicability of Weyuker Property Nine to Object-Oriented Structural Inheritance Complexity Metrics. IEEE Transaction on Software Engineering, Vol. 27, 2001, No. 4, pp. 361–364.
[12] Sharma, N.—Joshi, P.—Joshi, R.K.: Applicability of Weyuker's Property 9 to Object-Oriented Metrics. IEEE Transaction on Software Engineering, Vol. 32, 2006, No. 3, pp. 209–211.
[13] Deepti Mishra: New Inheritance complexity metricsfor object – oriented software systems:An evaluation with weyker's properties Computing and Informatics, Vol. 30, 2011, 267–293.
[14] Chidamber, S.R.—Kemerer, C.F.: To wards A Metrics Suite for Object Oriented Design,OOPSLA'91,pp. 197-211,1991.
[15] Chidamber, S.R.—Kemerer, C.F.: A Metrics Suite for Object Oriented Design, M.I.T.Solan School of Management 1993.
[16] wang CC, shih TK ,paiWC An automatic approach to object – oriented software testing and metrics for c++ inheritance hierarchies, proceedings International Conference on Automated Software Engineering (ASE'97), IEEE Computer Society press 1997;934-938
[17] Basili VR,Biand LC Melo WL A validation of object-oriented metrics as quality indicators, Technical Report,University of Maryland, Department of computer science,1995; 242-249.
[18] Chidamber, S.R.—Kemerer, C.F.: A Metrics Suite for Object Oriented Design.IEEE Transactions on Software Engineering, Vol. 20, 1994, No. 6, pp. 476–493.
[19] Ghassan alkadi, Application of a revised DIT metric to Redesign an OO Design, Journal of Object technology , Vol. 2,Issue 3,pp 897-910,2005.
[20] Basili, V.R.:Viewing Maintenance As Reuse Oriented Software Development. IEEE Software, Vol. 7, 1990, No. 1, pp. 19–25.
[21] Cartwright, M.—Shepperd, M.: An Empirical Analysis of Object Oriented Soft-ware in Industry. In: Bournemouth Metrics Workshop, April, Bournemouth, UK1996.
[22] Li, W.—Henry, S.: Object-Oriented Metrics That Predict Maintainability. Journal of Systems and Software, Vol. 23, 1994, No. 2, pp. 111–122.
[23] Sanjay Misra and Ibrahim Akman :Applicability of weyuker's Properties on OO Metrics: Some Misunderstandings , ComSIS Vol. 5, No. 1, June 2008.
[24] K. Rajnish, A. K. Choudhary, A. M. Agrawal, "Inheritance Metrics for Object-Oriented Design", *IJCSIT*, Vol. 2 No.6 , December2010,pp.13-26.
[25]K. Rajnish and V. Bhattacherjee, "Class Inheritance Metrics-An Analytical and Empirical Approach", *INFOCOMP-Journal of Computer Science*, Federal University of Lavras, Brazil, Vol. 7 No.3, pp. 25-34, 2008.

ISSN 2347-3983

International Journal of Emerging Trends in Engineering Research, Vol.4. No.10, Pages : 188-193 (2016)
Special Issue of ICACSSE 2016 - Held on September 30, 2016 in St. Ann's College of Engineering & Technology, Chirala, AP, India
http://www.warse.org/IJETER/static/pdf/Issue/icacsse2016sp34.pdf

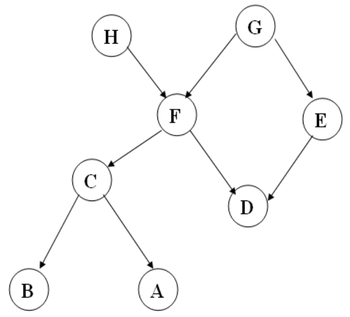

FIG 1(a) : Class Inheritance Hierarchy

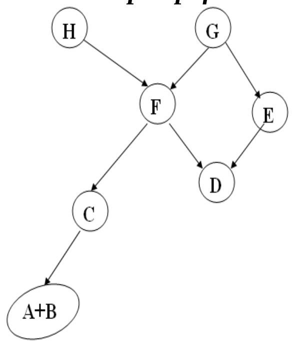

FIG 1(b) : A+B combined Class Inheritance Hierarchy(case-1)property-5.

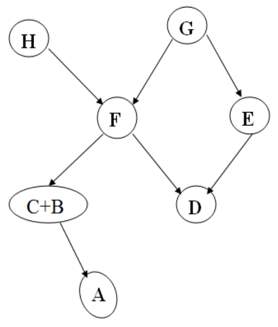

FIG 1(c) : C+B combined for Class Inheritance Hierarchy(case-2)property-5.

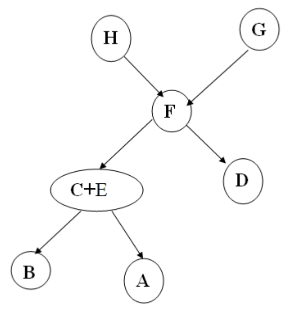

FIG 1(d) : C+E combined for Class Inheritance Hierarchy(case-3)property-5.

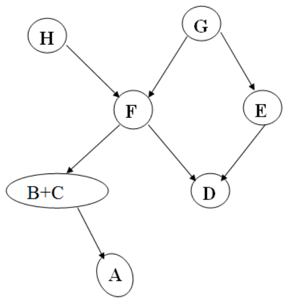

FIG 1(e) : B+C combined for Class Inheritance Hierarchy property-6.

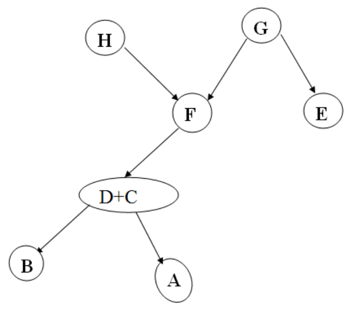

FIG 1(f) : D+Ccombined for Class Inheritance Hierarchy property-6.

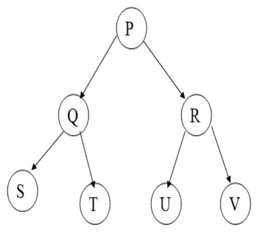

FIG 2(a) : Class Inheritance Hierarchy

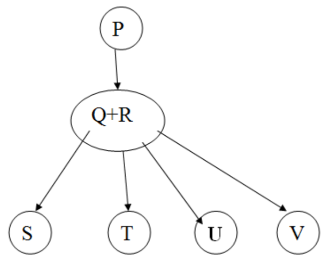

FIG .2(b) :Q+R combined for Class Inheritance Hierarchy(case-1)property-5.

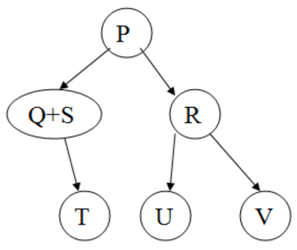

FIG 2(c) : Q+S combined for Class Inheritance Hierarchy(case-2)property-5.

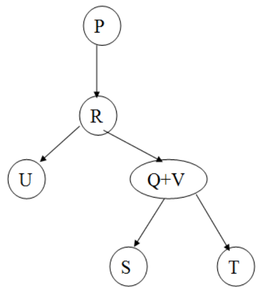

FIG 2(d) : Q+V combined for Class Inheritance Hierarchy(case-2 property-5.

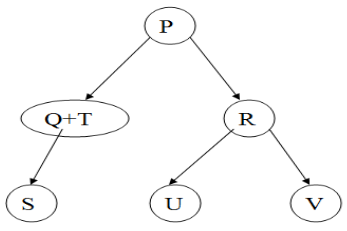

FIG 2(E) : Q+T combined for Class Inheritance Hierarchy property-6.

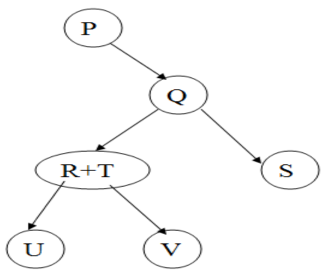

FIG 2(F) : R+T combined for Class Inheritance Hierarchy property-6