A Unified Framework for Software Coupling Measurement

Xiao-dan Li
School of Reliability and System Engineering
Beihang University
Beijing, China
Daniellee@dse.buaa.edu.cn

Yong-feng Yin
Science &Technology on Reliability &Environment Engineering Laboratory
Beihang University, Beijing, China
yyf@buaa.edu.cn

***Abstract*—With the development of the computer technology, the size and complexity of software increase sharply. It is necessary to have further researches into the softwarequality measures. However, as one of the most important measures, the coupling frameworks are too subjective and ambiguous. There is also a lack of coupling framework applicable to systems of different programming paradigms. In this paper, aunified framework for software coupling measurement is presented. The advantages and disadvantages of framework for object-oriented systems and procedure-oriented systems are taken into consideration comprehensively, and the coupling types are complemented and unified. At last, effectiveness of the framework is demonstrated through an experiment.**

Keywords- software quality; software measurement; software coupling

I. INTRODUCTION

Recently, much more emphasis on software quality has led to an increasingly large body of work being performed in evaluating the quality of software[1]. Stevens and Myers *et al.*[2] present the particular coupling framework for procedure-oriented systems. As to object-oriented systems, Briand *et al.*[3] provide framework for coupling measurement in object-oriented systems by comparing measures and their potential use. In [1], Briand *et al.* also propose a comprehensive suite of measures to quantify the level of class coupling during the design of object-oriented systems. Virdi *et al.*[4] study the different types of coupling and how the metrics perform under different environments. Gethers *et al.* [5] propose a new coupling metric which uses Relational Topic Models (RTM) to capture latent topics in source code classes and relationships among them. Eder[6] provides comprehensive taxonomy for coupling property and the suggestions to improve it. Újházi[7] presents a novel conceptual metric for measuring coupling in software systems which is based on the well-known CBO coupling metric. Rathore[8]propose a novel Class, Object and Inheritance based Coupling Measure (COICM) to find better OOP Paradigm. Offutt [9]focuses on the software coupling after the execution of software.However, there are many defects among the extant coupling framework. The coupling metric frameworks are too subjective; the definition of coupling types is ambiguous and incomplete; the mapping relationship between coupling types and coupling mechanism is fuzzy; the direction of coupling is not clear; there is also a lack of software coupling framework applicable to the software of different programming paradigms.

According to the issues above, this paper presents a unified framework for software coupling measurement which is applicable to both object-oriented systems and procedure-oriented systems. Our proposed approach thus fills a critical gap in the existing literature by introducing the unified framework for software coupling measurement to analyze the software coupling.

II. A UNIFIED FRAMEWORK FOR SOFTWARE COUPLING MEASUREMENT

In this section, we first introduce the coupling types of our framework. The coupling mechanism and direction are described. Then the mappings between coupling mechanism and coupling types are detailed. At last we illustrate the way to calculate the coupling degree.

A. Coupling type

According to the degree of coupling, the coupling types are divided into 11 kinds from the highest to the lowest as shown in Table 1.

TABLE 1 THE DEFINITION OF UNIFIED COUPLING TYPE

The unified coupling type		The definition of unified coupling type
Extended circular coupling		Extended circular coupling means that two classes (modules) constitute coupling of any typewith each other which forms a circular.
Content coupling		If module (class) B access to private (internal) part of module (class) A directly, then there is the content coupling from A to B.
Common coupling	Loose Common coupling	If the module (class) Aonly writes data into an unstructured, global, shared data space, while the module (class) B just takes data from the data space, there is loose common coupling from A to B. If the modules (classes) A and B both write and readthe data of an unstructured, global, shared data space, there is tight common coupling from A to B.
	Tight common coupling	
External coupling	Generalexternal coupling	If class A and Class B transfer the transient data by the instance variable method of these twoclasses or modules A and B are associated with an external environment, there is general external coupling from A to B. If m is implemented at A and m' defined at B communicate by a public instance variables which are inherited by class B from C, and A is not as same as B, A is neither a superclass nor a subclass of B, there is inherited external coupling from A to B.
	Inherited external coupling	
Control coupling	Conditioncontrol coupling	If Module (class) A control the internal logic of module (class) B by controllingthe condition statement, there is conditioncontrol coupling from A to B. If module (class) A controls the traverse logic of elements in the setinside module (class) B by changing the set type, there is cycliccontrol coupling from class A to class B
	Cyclic control coupling	
Stamp coupling	Passing stamp coupling	Ifmethod of module (class)A passes the whole data structure to the method of module (class) B as parameters althoughonly part of the data structuresuffice, there is passing stamp couplingfrom class A to class B. Ifmethod of module (class) Baccesses the composite data structure directly in module (class) A, althoughonly part of the data structuresuffice, there is accessing stamp couplingfrom class A to class B.
	Accessing stamp coupling	
Inheritance coupling(Not included in the procedure - oriented framework)	Modified Inheritance coupling	If class B is a direct or indirect subclass of class A: if class B not only adds new information, but also to change or delete any inheritance information arbitrarily, there is modified Inheritance coupling from class A to class B. If class B adds new information and alsochanges inheritance information according to predefined rules, there is refinedinheritance coupling from class A to class B. If class B only adds new information without modifying and refining any inheritance information, there is extendedinheritance coupling from class A to class B.
	Refined Inheritance coupling	
	Extended Inheritance coupling	
Instance coupling		If the module B (Class B) directly calls an instance method (the implementation)of module A (instance class A) instead of class Interface, there is instance coupling from class A to class B
Data coupling		If the module A (method of class A) passes to the module B (method of classB) parameters (basic data types or complex data type), and these parameter is are relevant as a whole, there is data coupling from class A to class B.
Interface coupling		If the module B (Class B) directly calls a class interface of module A (class A), there is interface coupling from class A to class B
No direct coupling		There is No direct coupling between class A and class B iftwo modules (classes) are completely independent.

B. The coupling mechanism of unified framework

1) Object–oriented coupling mechanism

Assume software includes class a, class b, class c; class a has method Ma1, Ma2; class b has method Mb; class c hasmethod Mc; class a, class b and class c own attributes Aa, Ab, Ac respectively. Coupling mechanism is defined as follows in Table 2:

TABLE 2 OBJECT–ORIENTED COUPLING MECHANISM

Coupling object	Coupling mechanism
Coupling between classes and methods	Oc1. Class a is the type of theparameter of method Mb in class b; Oc2. Class a is the type of theparameter of method called by Mb; Oc3. Class a is the type of the local variable in Mb; Oc4. Class a is the return type of method Mb; Oc5. Class a calls an instance method of class b directly;
Coupling between classes and attributes	Oc6. Class a is the type of attribute Ab; Oc7.Method Ma of class a references to attributes Ab; Oc8. Method Ma references to attribute Ab (Ab is complex data type), but Ma only uses part of the value in the Ab;
Coupling between methods	Oc9. Method Ma calls method Mbdirectly; Oc10. Method Ma and methodMb share data space; Oc11. Method Ma receives a pointer to method Mb; Oc12. Method Ma and methodMb share unstructured global data structures; Oc13. Method Ma and methodMb use the public attributes(instance variables) to transmit the transient data; Oc14. Method Ma1 and methodMa2 use the instance variable Aa to exchange data, and Aa is inherited from class b; Oc15. Method Ma and methodMb use the instance variable Aa to exchange data, and Aa is inherited from class c by class a; class a is different from class, and a is neither a superclass nor a subclass of b; Oc16. The input parameter of Ma is defined by class b. Ma only utilizes some of the components (attributes and methods) in the object parameter, rather the object as a whole; Oc17. Method Ma and methodMb communicate by the parameters of complex data type, and they are utilized as a whole;
Coupling between classes	Oc18.Only the interface inheritanceexists between class a and class b; Oc19. The Implementation inheritanceexists between class a and class b; Oc20. Class a and class b interact by interface;

2) Procedure-oriented coupling mechanism

Assume software includes module x and module y;

Pc1. x access the internal data of y directly

Pc2. x go into y by abnormal entrance;

Pc3. x and y share some code (only possible in assemble language);

Pc4. x has many entries;

Pc5. x and y share data space. x only writes the data into the shared data space, while y just take the data from the data space;

Pc6. x and y share an unstructured, global, shared data space which both write and read the data in data space;

Pc7. x and y are both associated with the same external environment;

Pc8. x and y exchange the information by parameters which include some control information;

Pc9. x and y transmit part of the value of the complex data structure by the interface;

Pc10.x calls the method or function of y directly;

Pc11.When accessing y, x exchange information via the data parameters (ie parameter table);

Pc12.x and y interact by interface;

C. The coupling direction

The coupling direction can be one-way transmission and two-way transmission. Most of the coupling direction can be determined by definition in Table 1.The direction of common coupling and external coupling is more complex to determine which should be defined solely as Table 3.

TABLE 3 DIRECTION DEFINITION OF COMMON COUPLING AND EXTERNAL COUPLING

Coupling type	Direction of coupling
common coupling	If module (class) reads data space, direction is in; otherwise, direction is out.

Module(class)A / Module(class) B	out	in	in-out
out	No common coupling	Loose common coupling from A to B	Loose common coupling from A to B
in	Loose common coupling from B to A	No common coupling	Loose common coupling from B to A
in-out	Loose common coupling from B to A	Loose common coupling from A to B	Tight common coupling between B and A

external coupling	The direction definition of external coupling is similar to that of common coupling. The public instance variables or external environment in the external coupling are the counterparts of the shared data space in common coupling.

D. The mapping between coupling mechanism and coupling type

The mapping is described between the mechanism for object-oriented (e.g.Oc1,Oc2) and procedure-oriented (e.g.Pc1,Pc2) coupling and corresponding coupling types. The mapping is shown in Fig.1.

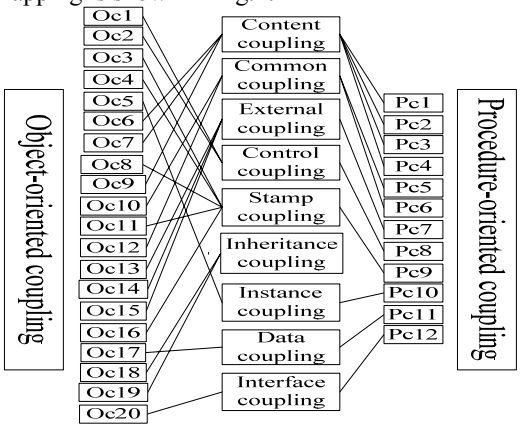

Figure 1.Mapping diagram for unified coupling framework

E. Coupling degree calculation

1) Coupling weight function

According to the different strength of coupling types, we assign different value to coupling types. The coupling weight function F(*x*,*y*) and the subtype coupling weight function are shown as in (1)(2).

$$P_{ij}=P_i\cdot\alpha_{ij} \quad (\alpha_{ij}\in[0,1]) \tag{1}$$

P_i is the weight for the coupling type. $P_{i,j}$ is the weight of *j*th coupling subtype of *i*th coupling type. α_{ij} is the coefficient for the *j*th coupling subtype of *i*th coupling type.

In order to set the reasonable weight for each couplingtype, we have the rules defined as follows:

a) In term with the procedure-oriented system, $P_{inheritance}$ is zero, and there is only one subtype of external coupling;

b) As to the coupling type with only one subtype, α_{ij} is one;

$$F(x,y)=\begin{cases} P_{ndc} & \text{no-direct coupling between x and y}\\ P_{interf} & \text{interface coupling between x and y}\\ P_{data} & \text{data coupling between x and y}\\ P_{instance} & \text{instance coupling between x and y}\\ P_{inheritance}\begin{cases} P_{M\text{-}inheritance} \\ P_{R\text{-}inheritance} \\ P_{E\text{-}inheritance} \end{cases} & \begin{matrix}\text{modified Inheritance coupling between x and y}\\ \text{refined Inheritance coupling between x and y}\\ \text{extended Inheritance coupling between x and y}\end{matrix}\\ P_{stamp}\begin{cases} P_{P\text{-}stamp} \\ P_{V\text{-}stamp} \end{cases} & \begin{matrix}\text{passing stamp coupling between x and y}\\ \text{accessing stamp coupling between x and y}\end{matrix}\\ P_{control}\begin{cases} P_{L\text{-}control} \\ P_{C\text{-}control} \end{cases} & \begin{matrix}\text{condition control coupling between x and y}\\ \text{cyclic control coupling between x and y}\end{matrix}\\ P_{external}\begin{cases} P_{inherited\text{-}external} \\ P_{normal\text{-}external} \end{cases} & \begin{matrix}\text{inherited external coupling between x and y}\\ \text{general external coupling between x and y}\end{matrix}\\ P_{common}\begin{cases} P_{T\text{-}common} \\ P_{L\text{-}common} \end{cases} & \begin{matrix}\text{tight common coupling between x and y}\\ \text{loose common coupling between x and y}\end{matrix}\\ P_{content} & \text{content coupling between x and y}\\ P_{ex\text{-}cir} & \text{extended circular coupling between x and y}\end{cases} \tag{2}$$

c) If a pair of components have formed extended circular coupling, the weight is P_{ex-cir} instead of the weight of their own coupling type;

2) Coupling degree definition

Next we will calculate the coupling degree. When there is no-direct coupling, the coupling degree is zero. Otherwise, the coupling degree between modules (classes) m and n, $t_{m,n}$is calculated by (3). $n_{i,j}$ is the number for the *j*th coupling subtype of *i*th coupling type. S denotes the set of the sub-type coupling of every coupling type, C is set of coupling type.

$$t_{m,n}=\begin{cases}\dfrac{\sum_{i\in C}\sum_{j\in S}(P_i\cdot\alpha_{i,j})\cdot n_{i,j}}{1+\sum_{i\in C}\sum_{j\in S}P_i\cdot n_{i,j}} & m\neq n\\ 1 & m=n\end{cases} \tag{3}$$

III. ILLUSTRATION

In this section, we will analyze the software witheight components programmed by C++. In this software, the

parameters are passed among different components mainly by external variables. Firstly, we obtain the static call graph of the software by *Testbed*, and analyze the call direction and the call number among various components. Secondly, the components sharing the same external variables are identified to calculate the common coupling. Then we have conducted some analysis about other coupling types. Finally, according to various types of coupling (The *F(x,y)* is defined below), we substitute the connection number of coupling and the corresponding weight into formula (3) to get the component coupling matrix **Corr**. There are also some function calls amongcomponents, relatively few inheritance relationships of different classes.*F*(*x*,*y*) is setas (4).The results are as follows:

$$F(x,y)=\begin{cases} 0 & \text{No-direct coupling between x and y} \\ 0.05 & \text{interface coupling between x and y} \\ 0.1 & \text{data coupling between x and y} \\ 0.2 & \text{instance coupling between x and y} \\ \begin{cases} 0.3\times 1 & \text{modified Inheritance coupling between x and y} \\ 0.3\times 0.8 & \text{refined Inheritance coupling between x and y} \\ 0.3\times 0.5 & \text{extended Inheritance coupling between x and y} \end{cases} \\ \begin{cases} 0.4\times 1 & \text{passing stamp coupling between x and y} \\ 0.4\times 0.8 & \text{accessing stamp coupling between x and y} \end{cases} \\ \begin{cases} 0.5\times 1 & \text{condition control coupling between x and y} \\ 0.5\times 0.7 & \text{cyclic control coupling between x and y} \end{cases} \\ \begin{cases} 0.6\times 1 & \text{inherited external coupling between x and y} \\ 0.6\times 0.8 & \text{general external coupling between x and y} \end{cases} \\ \begin{cases} 0.7\times 1 & \text{tight common coupling between x and y} \\ 0.7\times 0.8 & \text{loose common coupling between x and y} \end{cases} \\ 0.8 & \text{content coupling between x and y} \\ 0.9 & \text{extended circular coupling between x and y} \end{cases} \tag{4}$$

$$\mathbf{Corr}=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.64 & 0 & 0.59 & 0.78 & 0 & 0.66 \\ 0.47 & 0.87 & 1 & 0 & 0.65 & 0.76 & 0.65 & 0.94 \\ 0.80 & 0.64 & 0.75 & 1 & 0.45 & 0.67 & 0.25 & 0.85 \\ 0.65 & 0.84 & 0.88 & 0.67 & 1 & 0.77 & 0.25 & 0.77 \\ 0.45 & 0.90 & 0.80 & 0.75 & 0.86 & 1 & 0 & 0.77 \\ 0 & 0 & 0.43 & 0.20 & 0 & 0.45 & 1 & 0 \\ 0.35 & 0.83 & 0.85 & 0.70 & 0.82 & 0.80 & 0 & 1 \end{bmatrix}$$

As shown in **Corr**,the changes in components four and five, even the failures, will have a great impact on other components, whereas there are less couplings from components one to others, which satisfies the requirements of the program design. However, there are lots of couplings to component one from other components which will influence component one so much. Thus component one is very susceptible by others.

IV. CONCLUSION

Recently, software industry has developed rapidly. With the continuous progress of industrialization level, the requirements of the software quality are getting higher and higher, so we must have further investigation into softwaremetrics. By coupling metric we can have a more thorough understanding of the correlation among the components inside the software. In this paper, according to the inconsistencies of the current software coupling metrics norms, a unified framework for software coupling measurement is put forward. The advantages and disadvantages of framework for object-oriented Systems and procedure-oriented systems are taken into consideration comprehensively, and the coupling types are complemented and unified on the basis of the two kinds of framework; At the meantime, the mappings between the coupling types and coupling mechanism is refined, and the extant coupling metric formula is optimized; The coupling degree of the software components is analyzed quantitatively.Next, we will study the formal expression for softwarecoupling framework and develop a tool based on our theory.

V. ACKNOWLEDGMENT

This work was supported by Fundamental Research Funds for the Central Universities (YWF-11-03-Q-114) and National Natural Science Foundation of China (61202077).

REFERENCES

[1] L. C. Briand,J. W. Daly,J. K. Wust. “A unified framework for coupling measurement in object-oriented systems.” Software Engineering, IEEE Transactions on, 25(1) pp.91-121, 1999.

[2] W. P. Stevens, G. J. Myers,L. L.Constantine,“Structured design”. IBM Systems Journal, 13(2) pp.115-139, 1974.

[3] L. Briand, P. Devanbu,W. Melo, (1997, May). “An investigation into coupling measures for C++.” In Proceedings of the 19th international conference on Software engineering (pp. 412-421). ACM.

[4] H. S. Virdi, B. Singh, (2012, July). “Analysis of the software code based upon coupling in the software.”In Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on (pp. 1-4). IEEE.

[5] M.Gethers,D.Poshyvanyk, (2010, September). “Using relational topic models to capture coupling among classes in object-oriented software systems.” In Software Maintenance (ICSM), 2010 IEEE International Conference on (pp. 1-10). IEEE.

[6] J. Eder, G. Kappel, M.Schrefl, “Coupling and cohesion in object-oriented systems.” Technical Reprot, University of Klagenfurt, Austria, 1994.

[7] B.Újházi, R.Ferenc, D.Poshyvanyk,T. Gyimóthy, (2010, September). “New conceptual coupling and cohesion metrics for object-oriented systems.” In Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on (pp. 33-42). IEEE.

[8] M. N. P. S. Rathor, R.Gupta,“A Novel Class, Object and Inheritance based Coupling Measure (COICM) to Find Better OOP Paradigm using JAVA.” International Journal, 2011.

[9] J. Offutt, A.Abdurazik, S. R.“Schach, Quantitatively measuring object-oriented couplings.” Software Quality Journal, 16(4), pp.489-512, 2008.