Measuring Software Coupling

JARALLAH S. ALGHAMDI
Information & Computer Science Department
King Fahad University of Petroleum & Minerals
Dhahran, SAUDI ARABIA

Abstract: - Coupling is one of two fundamental attributes of software that significantly impact system quality. This work presents a method for measuring coupling in a software system using two matrices. The coupling calculation is performed in two major steps. The first step creates a matrix that captures the nature of the underlying software system. The second step produces a matrix that indicates the degree of coupling between each pair of system components, and also measures the overall system coupling.

Key-Words: - Coupling,components

1 Introduction

Coupling refers to the degree of interdependence between software system components. It is considered to be one of the two fundamental attributes of software that have a major impact on the quality of a system, and therefore its measurement is of great importance. Software designers are expected to determine, trace and manage the factors that contribute to coupling as a means of developing reliable and maintainable software and reducing costs.

Related work is discussed briefly in the next section. Then two examples of coupling metrics are provided, followed by a broad classification of coupling measures. Our coupling measure, which is calculated in two major steps, is then presented. The first step creates a matrix that describes the nature of the underlying software system. The second step produces a matrix that indicates the degree of coupling between each pair of system components, along with the overall coupling of the system. Some results and comparisons of various metrics are then presented, followed by some concluding remarks and observations.

2 Related Work

In their seminal work, Stevens, Myers, and Constantine introduced the concept of coupling in procedural programming [14]. Six levels of coupling based on the Myers classification were then defined in [16]. We redefine these types of coupling as binary relations on a pair of system components, x and y; these classifications are shown here in order from worst to best:

- Content coupling relation $R_5 : (x, y) \in R_5$ if x refers to the internals of y, i.e., it branches into, changes data, or alters a statement in y.
- Common coupling relation $R_4 : (x, y) \in R_4$ if x and y refer to the same global variable.
- Control coupling relation $R_3 : (x, y) \in R_3$ if x passes a parameter to y that controls its behavior.
- Stamp coupling relation $R_2 : (x, y) \in R_2$ if x passes a variable of a record type as a parameter to y, and y uses only a subset of that record.
- Data coupling relation $R_1 : (x, y) \in R_1$ if x and y communicate by parameters, each one being either a single data item or a homogeneous set of data items that does not incorporate any control element.
- No coupling relation $R_0 : (x, y) \in R_0$ if x and y have no communication, i.e., are totally independent.

This ordered classification has obtained general acceptance and has formed the basis for several software metrics such as the coupling metrics proposed by Fenton and Melton [8] and by Dhama [7], which we describe briefly.

2.1 Fenton and Melton Software Metric

Fenton and Melton [8] have proposed the following metric as a measure of coupling between two components x and y:

$$C(x, y) = i + n/(n+1) \text{ where,}$$

n = number of interconnections between x and y, and

i = level of highest (worst) coupling type found between x and y.

Coupling Type	Coupling Level	Modified Definition between components x and y
Content	5	Component x refers to the internals of component y, i.e., it changes data or alters a statement in y.
Common	4	Components x and y refer to the same global data.
Control	3	Component x passes a control parameter to y.
Stamp	2	Components x passes a record type variable as a parameter to y.
Data	1	Components x and y communicate by parameters, each of which is either a single data item or a homogenous structure that does not incorporate a control element.
No Coupling	0	Components x and y have no communication, i.e., are totally independent.

Table 1: Fenton and Melton Modified Definition for Myers Coupling Levels

The level of coupling type is based on the Myers classification and is assigned a numeric value, as shown in Table 1.

2.2 Dhama Coupling Metric

Dhama [7] proposes a coupling metric that measures the coupling inherent to an individual component C, and is equal to:

$$1/(i_1 + q_6 i_2 + u_1 + q_7 u_2 + g_1 + q_8 g_2 + w + r)$$

where,

q_6, q_7 and q_8 are constants assigned a value of 2 as a heuristic estimate, and

i_1 is a data parameter,

i_2 is a control parameter,

u_1 is a data return value, and

u_2 is a control return value.

For global coupling:

g_1 is the number of global variables used as data, and

g_2 is the number of global variables used for control.

For environment coupling:

w is the number of other components called from component C, and

r is the number of components calling component C; it has a minimum value of 1.

The following observations can be made concerning these two coupling metrics:

1. The Fenton and Melton metric is a direct quantification of the Myers coupling levels, whereas the Dhama metric considers the number of variables or parameters belonging to categories that are less directly influenced by the Myers classification.
2. The highest coupling level between two components is the main determinant of their coupling value in the Fenton and Melton metric. The coupling value approaches the value of next coupling level as the number of interconnections between the two components increases.
3. The Fenton and Melton metric considers all types of interconnections between components to have the same complexities and have the same effects on coupling.
4. The Dhama metric considers the effect on coupling of a parameter to be the same as the effect of a global variable, which is a major deviation from the Myers classification scheme.
5. The Fenton and Melton metric is an example of an inter-modular coupling metric, which calculates the coupling between each pair of components in the system. The Dhama metric is an example of an intrinsic coupling metric, which calculates the coupling value of each component individually.

3 Classification of Coupling Measures

Existing coupling measures can be broadly classified into the following two groups:

1 Procedural programming coupling measures: these measure the coupling of software components that are implemented in procedural programming languages; examples include metrics proposed by Lohse and Zweben [11], Huches and Basili [10], Fenton and Melton [8], Offut, Harold and Kotle [12], and Dhama [7]. This class of metrics is heavily influenced by the Myers classification of coupling levels.

2 Object-oriented coupling measures: these measure the coupling of software components that are implemented in object-oriented programming languages; examples include metrics proposed by Henry and Li (1992), Tegarden and Sheetz (1992), J-Y Chen and J-F La (1993), Lorenz and Kidd (1994) [9], and Chidamber and Kemerer [6].

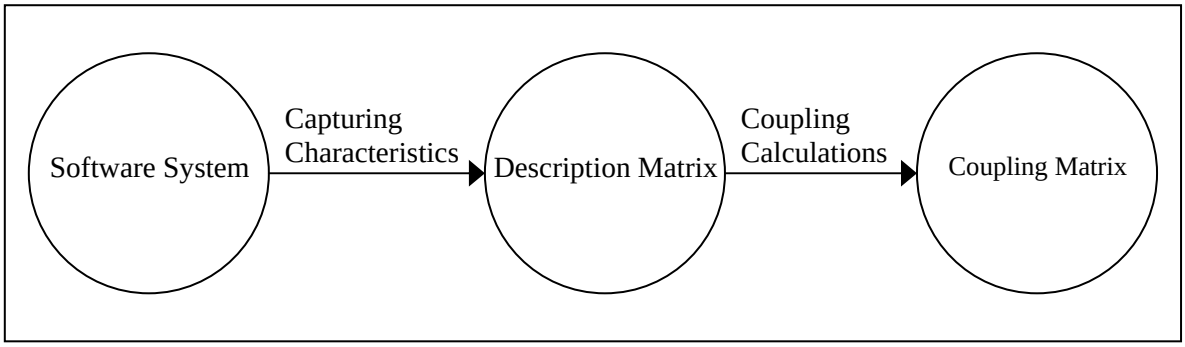

Fig.1 Major Steps of the Framework

This paper proposes a framework that can be applied to both of the above paradigms. Each paradigm requires a different process to deal with the first step of collecting coupling data from either the system design or the code. The second step, which calculates the actual coupling values, operates in an identical manner regardless of the paradigm used.

4 The Proposed Coupling Metric

The general approach of other coupling metrics is to calculate the coupling values for a system in one step. The approach of the framework outlined in this paper involves breaking the calculation of coupling into two steps, as shown in Figure 1. The first step is to generate a description matrix that captures the factors that affect coupling in a system. The second step is to calculate the coupling between each two components of the system from the description matrix to produce a coupling matrix.

The next two sections of this paper discuss the generation of the description and the coupling matrices.

5 Generation of the Description Matrix

The objective of generating a description matrix is to create a structure that captures all of the characteristics of a software system that relate to coupling, which can then be used to calculate coupling information for that system. The description matrix, shown in Table 2, is a m components by n members matrix. Each component of the software system is represented by a row of the description matrix; these are classes in an object-oriented system, or functions, procedures, and subroutines in a procedural system. Columns of the description matrix represent members, which are methods and instance variables in an object-oriented system, or variables and parameters in a procedural system.

Component \ Element	E_1	E_2	...	E_n
C_1	d_{11}	d_{12}		d_{1n}
C_2	d_{21}	d_{22}		d_{2n}
...				
C_m	d_{m1}	d_{m2}		d_{mn}

Table 2: The Description Matrix

Each entry d_{ij} in the description matrix shows the weight of a member E_j with respect to component C_i. The greater the value of d_{ij}, the more influence member E_j will have on the coupling value between components C_i and C_k, when $d_{xj} > 0$. A zero value for d_{ij} indicates that E_j is inaccessible from C_i. When developing a software metric based on our framework, the greatest effort required is in constructing an accurate description matrix; the coupling matrix described next is automatically generated from this description matrix. The accuracy of the description matrix very much depends on how accurately the factors influencing coupling are represented within it. This criterion is mainly considered while populating the description matrix.

Therefore, the weighting scheme utilized for the description matrix is of great significance. This weighting scheme can use the values from an existing coupling metric (such as assigning a weight of 2 to an integer parameter as was used by some researchers), or a new scheme can be adopted.

6 Generation of the Coupling Matrix

The coupling matrix for a software system of m components is a matrix of order $m \times m$, where each row and each column represents a component of the system, as shown in Table 3. The value of each entry C_{ij} of the coupling matrix indicates the extent to which the row component, C_i, is coupled to the column component, C_j. The values can be calculated from the description matrix in various ways; for example, the degree of coupling between two

components can be the sum of the weights of all members shared by the two components. Can and Ozkarahan similarity measure [4] was shown to be more effective and efficient method for information retrieval systems [1] due to role of their coupling calculation scheme that calculates coupling between different documents. Therefore, we have chosen to use their coupling calculation scheme to calculate the coupling between each two components.

Component	C_1	C_2	...	C_m
C_1	C_{11}	C_{12}		C_{1m}
C_2	C_{21}	C_{22}		C_{2m}
...				
C_m	C_{m1}	C_{m2}		C_{mm}

Table 3: The Coupling Matrix

The formula used to calculate each entry C_{ij} of the coupling matrix is:

$$C_{ij} = \left(\sum_{k=1}^{n} d_{ik} \times d_{jk} \times \beta_k \right) \Big/ \alpha_i$$

where,

d_{ik}, and d_{jk} are entries of the description matrix,

α_i is the sum of the entries of the *i*th row of the description matrix, and.

β_k is the reciprocal of the sum of the kth column of the description matrix. Mathematically they are:

$\alpha_i = \sum_{j=1}^{n} d_{ij}$ and $\beta_k = 1 \Big/ \sum_{i=1}^{m} d_{ik}$.

7 Properties of the Coupling Matrix

Each entry in the coupling matrix has the following meaning:

- C_{ij} is the extent to which component C_i is coupled to component C_j, for $i \neq j$, and
- C_{ii} is the extent to which component C_i is decoupled from the other components.

We can observe that the following properties hold for the coupling matrix:

1. $0 \le C_{ij} \le 1$.
2. The sum of the values in each row is equal to 1.
3. If $C_{ij} = 0 \rightarrow C_{ji} = 0, C_{ji} > 0$ if and only if $C_{ij} > 0$ and C_{ij} may or may not be equal to C_{ji}.
4. $C_{ii} = C_{jj} = C_{ij} = C_{ji}$ if and only if all entries in d_i and d_j are identical.

A more detailed description of these properties can be found in [5]. The above properties can be further elaborated as follows [3]:

- Property 1:
  a) $C_{ij} = 0$ if $\forall k(d_{jk} > 0 \rightarrow d_{ik} = 0)$,
  b) $C_{ii} = 1$ if $\forall k(d_{jk} > 0 \rightarrow$ there exists not $d_{jk} > 0$ where $i <> j))$, and
  c) $0 < C_{ij} < 1$ if $\forall k($ there exists some $(d_{ik} > 0)$ and there exists some $(d_{jk} > 0)$ where $i <> j$).
- Property 2: For each component in the coupling matrix, the sum of the entries in the corresponding row is equal to 1.
- Property 3:
  a) $(C_{ij} = 0) \rightarrow (C_{ji} = 0)$,
  b) $(C_{ij} > 0) \rightarrow (C_{ji} > 0)$, and
  c) There may exist $C_{ij} = C_{ji}$.
- Property 4:
  $\forall k(d_{ik} = d_{jk}) \rightarrow C_{ii} = C_{jj} = C_{ij} = C_{ji}$.

Because the sum of the values in each row of the coupling matrix sum to 1, these values are proportional to each other. An increase in the value of an entry C_{ij} in the coupling matrix indicates an increase in the coupling between the corresponding components C_i and C_j [3]. From this property we can deduce the following metrics:

1. The coupling value of component $C_i = 1 - C_{ii}$.
2. The overall coupling of the system $C_s = \sum_{i=1}^{m} C_i$.
3. The average coupling of the system $C_{avg} = C_s / m$.

The values in each row of the matrix are proportional to each other but the values in different rows are not because the coupling matrix as a whole is not normalized. Therefore, each row of the coupling matrix is stand alone with elements of a row being proportional to each other. Each entry of the coupling matrix C_{ij} reflects the extent of influence C_j has on the members of C_i. Hence, C_{ij} reflects the coupling of C_i to C_j. The diagonal entry for each row, C_{ii}, indicates how much component C_i is coupled to itself relative to its coupling to the other components of the system. In other words, C_{ii} reflects the extent to which C_i is decoupled from other components of the software system. Each row i can be divided by its corresponding diagonal entry, C_{ii}, to achieve a normalized coupling matrix [2].

Thus, it is evident that, once we build the description matrix, the coupling matrix can be easily derived from it [3]. Therefore the emphasis should

be placed on building a reliable description matrix that represents all of the members and components of a given system without losing any of the essential properties.

8 An Example

A description matrix for a system that consists of 4 components and 8 members is shown in Table 4.

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
C_1	0.7	0.9	0.0	0.5	0.3	0.8	0.1	0.2
C_2	0.2	0.0	1.0	0.0	0.7	0.4	0.5	0.1
C_3	1.3	2.0	0.0	0.2	0.1	0.1	0.3	0.3
C_4	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.7

Table 4: An example of a description matrix

Table 5 shows the coupling matrix derived from the description matrix of Table 4:

	C_1	C_2	C_3	C_4
C_1	0.420	0.158	0.381	0.041
C_2	0.190	0.496	0.118	0.195
C_3	0.310	0.080	0.548	0.062
C_4	0.084	0.333	0.158	0.425

Table 5: Coupling matrix derived from Table 4

Based on the values in Table 4 and Table 5, the Overall System Coupling = 2.111 and the Average System Coupling = 0.527.

The higher the value of an entry C_{ij} in the coupling matrix, the more the component of the corresponding row, C_i, is coupled to the component of the corresponding column, C_j. For example, component *C1* is more strongly coupled to component *C3* than it is to either *C2* or *C4*; this is reflected by the fact that *C13* > *C12* and *C13* > *C14*. The higher the value of a diagonal entry in the coupling matrix (e.g., *C11* or *C22*), the more the corresponding component is decoupled from the other components of the system; in other words, the less interdependent it is with the other components. The four main factors in determining the value of a diagonal entry C_{ii} are:

1. The number of components that share members with component C_i,
2. The weights of the members of C_i, which appear in row *i* of the description matrix,
3. The weights of the members of C_i that appear in other components' entries in the description matrix, and
4. The number of members of C_i that are shared by other components.

These four factors should be reflected in metrics that measure the coupling of software systems. They are naturally obtained in our framework by the mapping from the description matrix to the coupling matrix, which is one of the key strengths of this framework.

Changing the weight of a member with respect to a particular component in the description matrix will result in changed values in the coupling matrix. To illustrate these effects, let us remove the first member, *E1*, from the second component, *C2*. This means changing d_{21} in the description matrix from 0.2 to 0.0. The resulting coupling matrix can be seen in Table 6. The effects of this change are:

1. The overall system coupling and the average system coupling decreased.
2. The coupling values of C_{12} and C_{21} decreased because the two components C_1 and C_2 no longer share d_{21} and therefore have one less member in common.
3. The coupling values of C_{23} and C_{32} decreased for the same reason.
4. The values of C_{11}, C_{22}, and C_{33} increased, which indicates that each of these three components has become more decoupled from the other components of the system.
5. All of the coupling values in the second row changed, reflecting the fact that the relative degree of coupling between C_2 and other components changed.
6. Coupling values are changed only for components that originally included E_1. In particular, note that d_{41} was zero in the original description matrix, which means that C_4 did not include E_1; therefore, C_{14}, C_{34}, and all of the coupling values in the 4th row have not changed.

Thus, it can be observed that all changes to the coupling matrix occurred in accordance with the expected behavior.

	C_1	C_2	C_3	C_4
C_1	0.427	0.139	0.393	0.041
C_2	0.181	0.526	0.083	0.210
C_3	0.320	0.052	0.566	0.062
C_4	0.084	0.333	0.158	0.425

Table 6: The coupling matrix after d_{21} is changed to zero.

9 Results

The framework described above has been applied to software systems created within the procedural paradigm [13, 15]. Two sets of test cases were used to compare the results of our metric against three other metrics. The first set consists of six small programs with sizes of 29, 66, 69, 117, 104, and 154 LOC. For each test case, the system's coupling was analyzed manually according to Myers' levels, and the expected behavior was recorded. The coupling

	Remove Proc	Combine Procs	Replace Proc	Inc Elm Weight	Use Param	Total
Our metric	1	1	1	1	1	5
Fenton and Melton	1	1	0	0	0	2
Chen and Lu	1	1	1	0	0	3
Offut, Harrold and Kotle	1	1	0	0	0	2
Dhama	1	1	1	1	0	4

Table 7: The coupling matrix after d_{21} is changed to zero.

values for each test case were then calculated using our metric and the three other metrics and compared against the results from the manual analysis [13, 15]. The results of applying our software metric were consistent with the Myers' classification protocol in all six test cases. The Fenton and Melton [8] metric was consistent in 5 test cases, as was the Offut, Harold and Kotle [12] metric. The Dhama [7] metric was only consistent with Myers' levels in three of the six test cases.

The objective of the second set of test cases was to test the ability of the four metrics to capture the effect on coupling when atomic changes are made to the system. Accordingly, the following five modifications were made to the system in separate experiments:

1. Remove a procedure from the program so that the coupling for all procedures, as well as the overall system coupling, is expected to decrease.
2. Combine two procedures into one in a program where again the expected change is that the coupling for all procedures and the system as a whole are expected to decrease.
3. Replace a procedure with another one such that the coupling of the replaced procedure, of procedures calling the replaced procedure, and the overall system are expected to increase, while the coupling of all other procedures should decrease.
4. Increase a member weight so that the procedure that contains that member and the overall system coupling should increase.
5. Replace the use of global variables with parameters so that the coupling of all procedures and of the system should decrease.

Table 7 summarizes the results of the experiments and shows the comparison between the proposed metric and four other metrics in terms of their sensitivity to Atomic Modifications. An entry of 1 in the table indicates that the corresponding metric was able to reflect the expected change in coupling for that experiment; an entry of 0 indicates otherwise [15].

A tool to measure the inheritance coupling in an object-oriented system based on the framework presented in this paper has also been implemented. Further experiments using the proposed metric are currently underway.

10 Conclusion

A framework for measuring coupling between program components was presented. Instead of tracing all coupling factors of all members in each component, this framework makes the measurement of coupling easier by breaking it down into two major steps and provides a systematic procedure for each step. These steps are the representation of the system using a description matrix, and the mapping from the description matrix to a coupling matrix. This division into two steps also enables progress to be made independently on each step. In addition, this framework can be easily applied to both procedural and object-oriented paradigms. A metric was implemented for procedural systems, and experimental results using this metric were summarized. Research aimed at adapting this framework to the object-oriented paradigm is currently underway. As expected from coupling metrics, this framework should assist in the identification of risky areas of a software system that require redesign or further testing.

References:

[1] J. AlGhamdi, *Implementing and testing the cover coefficient based information retrieval system*, MS Thesis, Arizona State University, 1986.

[2] J. Al-Ghamdi, *Programming Languages: A Qualitative Methodology for Assessments Software Metrics to Measure Syntactic Properties*, Ph. D. Thesis, Arizona State University, August 1994.

[3] J. Al-Ghamdi, Measuring Interactions between Objects, *Technical Report,* King Fahd

University of Petroleum and Minerals, pp. 1- 6, 1995.

[4] F. Can and E. Ozkarahan, "Concepts of the cover coefficient-based clustering methodology," *Proceedings of the sixth annual ACM SIGIR Conference*, June 1985, pp. 1-14.

[5] Fazli Can and Esen A. Ozcarahan, "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases," *ACM Transactions on Database Systems*, Vol 15, No 4, Dec 1990, pp. 483 - 517.

[6] Shyam R. Chidamber and Chirs F. Kemerer, "A Metrics Suite For Object Oriented Design," *IEEE Transactions On Software Engineering*, Vol. 20, No. 6, Jun. 1994, pp 476- 493.

[7] H. Dhama, Quantitative Models of Cohesion and Coupling in Software, *Journal of System and Software*, Vol. 29, No. 1, pp. 65-74, Apr. 1995.

[8] Norman Fenton and Austin Melton, "Deriving Structurally Based Software Measures," *J. System Software*, 1990, Vol. 12, pp 177-187.

[9] Brian Henderson-Sellers, *Object-Oriented Metrics : Measures of Complexity*, Prentice Hall PTR, 1996.

[10] D. H. Hutches, and V. R. Basili, System Structure Analysis: Clustering with Data Bindings, *IEEE Transactions on Software Engineering*, Vol. 11, no. 8, pp. 749-757, Aug. 1985.

[11] J. B. Lohse, and S. H. Zweben, Experimental Evaluation of Software Design Principles: An Investigation into the Effect of Module Coupling on System Modifiability, *Journal of System and Software*, Vol. 4, No. 1, pp. 301-308, Feb. 1984.

[12] A. J. Offut, M. J. Harrold, and P. Kotle, A Software Metric System for Module Coupling, *Journal of System and Software*, Vol. 20, No. 3, pp. 295-308, Mar. 1993.

[13] S. H. Al-Nasser, *Measuring the Coupling of Imperative Computer Programs*, Master Thesis, King Fahd University of Petroleum & Minerals, Appendix A, June 1997.

[14] E. Yourdon, and L. L. Constantine, *Structured Design*, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[15] J. AlGhamdi, M. Shafique, S. Al-Nasser, and T. Al-Zubaidi, "Measuring the Coupling of Procedural Programs," *Proceedings of the AICCSA Conference*, June 2001.

[16] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*. 2nd Edition. Reading, 1997.