Measuring Coupling and Cohesion: An Information-Theory Approach

Edward B. Allen
Taghi M. Khoshgoftaar*
Florida Atlantic University
Boca Raton, Florida USA

Abstract

The design of software is often depicted by graphs that show components and their relationships. For example, a structure chart shows the calling relationships among components. Object-oriented design is based on various graphs, as well. Such graphs are abstractions of the software, devised to depict certain design decisions. Coupling and cohesion are attributes that summarizes the degree of interdependence or connectivity among subsystems and within subsystems, respectively. When used in conjunction with measures of other attributes, coupling and cohesion can contribute to an assessment or prediction of software quality.

Let a graph be an abstraction of a software system and let a subgraph represent a module (subsystem). This paper proposes information theory-based measures of coupling and cohesion of a modular system. These measures have the properties of system-level coupling and cohesion defined by Briand, Morasca, and Basili.

Coupling is based on relationships between modules. We also propose a similar measure for intramodule coupling based on an intramodule abstraction of the software, rather than intermodule, but intramodule coupling is calculated in the same way as intermodule coupling. We define cohesion in terms of intramodule coupling, normalized to between zero and one. We illustrate the measures with example graphs. Preliminary analysis showed that the information-theory approach has finer discrimination than counting.

Keywords*: software metrics, coupling, cohesion, call graph, information theory, entropy, excess entropy*

*Readers may contact the authors through Taghi M. Khoshgoftaar, Empirical Software Engineering Laboratory, Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA. Phone: (561)297-3994, Fax: (561)297-2800, Email: taghi@cse.fau.edu, URL: www.cse.fau.edu/esel/.

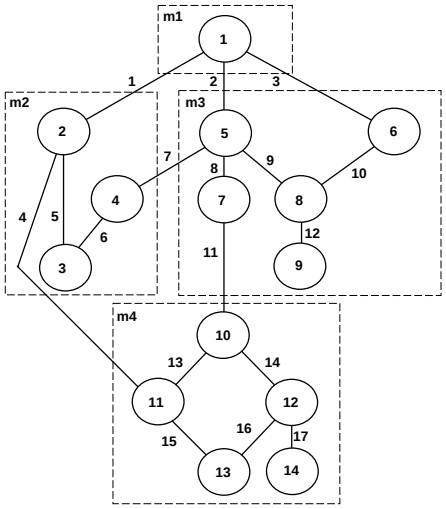
1. Introduction

The essence of high-level software design is identification of components and their relationships. The existence of nodes in a software design diagram and the structure of relationships depict design decisions. For example, software architectures are often represented by a variety of graphs [32]. Another example is a structure chart, which depicts the hierarchy of control among procedural components, where the calling component invokes the called component.

The software engineering community often discusses software design in terms of vague concepts, such as complexity, coupling, or cohesion. Software-design text books admonish would-be designers to "reduce coupling among modules and improve cohesion within modules" [27, 29]. Only in hindsight can one identify good and bad decisions. However, a balanced set of software design measurements early in the design process can give insight into design attributes, and may be inputs to models that predict software quality. If the attributes or predicted quality are unsatisfactory, then one can revise the design before changes become too expensive. Although many software complexity metrics have been proposed, fewer coupling and cohesion measures have been proposed [7, 8, 15]. Recent research has focused on various kinds of coupling and cohesion in object-oriented designs [6, 9, 10, 13, 16, 17].

Briand, Morasca, and Basili, propose a set of properties in a formal framework to define *coupling* and *cohesion* more precisely [11, 12]. *Coupling* and *cohesion* are the names of families of measures on graphs that have certain properties in common. More recently, Morasca and Briand extended their framework from graphs (binary relations) to relations in general [25].

Information theory is a promising foundation for software measurement that accounts for patterns in software attributes [1, 2, 3, 19]. This paper proposes specific measures based on information theory, namely,



This is similar to Figure 2 in [11].

Figure 1. Example of a Modular System

intermodule coupling, *intramodule coupling*, and *cohesion*. Intermodule coupling is ordinary coupling, as defined by Briand, Morasca, and Basili [11]. Intramodule coupling is similar, but measures a different subgraph. Cohesion is defined in terms of intramodule coupling, normalized to between zero and one, as prescribed by Briand, Morasca, and Basili [11]. These measures can be used to measure many kinds of software design abstractions, and thus, many kinds of software coupling. We illustrate the measures with examples.

2. Intermodule Coupling

Briand, Morasca, and Basili [11] propose a set of properties that defines the concept *coupling of a modular system* where a module is a subsystem. Figure 1 is an example of a modular-system graph. The modules (dashed boxes) partition the nodes in the system. Table 1 summarizes the properties of the coupling of a modular system [11]. We reserve the term *coupling of a modular system* for measures that have these properties. The discussion below indicates how these properties influenced the design of our measure.

2.1. Measurement protocol

A *measurement protocol* is a set of conditions that assures consistent repeatable measurement of an attribute [22]. A valid protocol is independent of the measurer and the measurement environment. It is also compatible with the desired unit of measure and the purpose of the measurement.

Table 1. Properties of Coupling

Concept/Properties
Coupling of a Modular System:
1. Nonnegativity. Coupling of a modular system is nonnegative.
2. Null value. Coupling of a modular system is null if its set of intermodule edges is empty.
3. Monotonicity. Adding an intermodule edge to a modular system does not decrease its coupling.
4. Merging of modules. If two modules, m_1 and m_2, are merged to form a new module, $m_{1\cup 2}$, that replaces m_1 and m_2, then the coupling of the modular system with $m_{1\cup 2}$ is not greater than the coupling of the modular system with m_1 and m_2.
5. Disjoint module additivity. If two modules, m_1 and m_2, which have no intermodule edges between nodes in m_1 and nodes in m_2, are merged to form a new module, $m_{1\cup 2}$, that replaces m_1 and m_2, then the coupling of the modular system with $m_{1\cup 2}$ is equal to the coupling of the modular system with m_1 and m_2.

We begin with any measurement protocol that results in a graph representing some aspect of software design. Many design methods produce graphs as artifacts. Software is the accumulation of a large number of decisions on many levels, from those very close to the code to those that constitute the system architecture. A design abstraction represents a set of design decisions that are related to each other, and each decision is an element of information. An artifact created during the design phase represents the design decisions made at that time, and embodies the abstraction of the software that was used by the designer.

Various protocols can result in different abstractions of software, and thus, result in different measures. For example, Briand, Daly, and Wüst survey numerous proposed measures of coupling and cohesion for object-oriented designs [9, 10]. The measures are based on various abstractions, such as class inheritance, class type, method invocation, and class-attribute references, and thus, measure various software attributes. Our approach is compatible with all of these abstractions, as well as graphical abstractions, such as call graphs, for the procedural development paradigm.

Given an abstraction of software represented graphically as a modular system, MS, with n nodes partitioned into modules [11], we make a further abstraction for coupling. It is not necessary for MS to be a directed graph. We represent the system's environment by one additional disconnected node to represent the fact that

interfaces to the environment of the system are not of interest here. Consider the subgraph, S, consisting of all nodes in MS plus the environment node, and all intermodule edges. The end points of an *intermodule* edge are not in the same module. Without loss of generality, index the environment node as $i = 0$, and the nodes in MS as $i = 1, \ldots, n$. It is not necessary for subgraph S to be a connected graph. Because diagrams in software engineering, such as structure charts, identify each node with a component name, we choose an abstraction that maintains a distinct identity for each relationship. For example, Figure 2 depicts the intermodule-edges subgraph, S, for the modular system in Figure 1. (The table in Figure 2 is discussed below.)

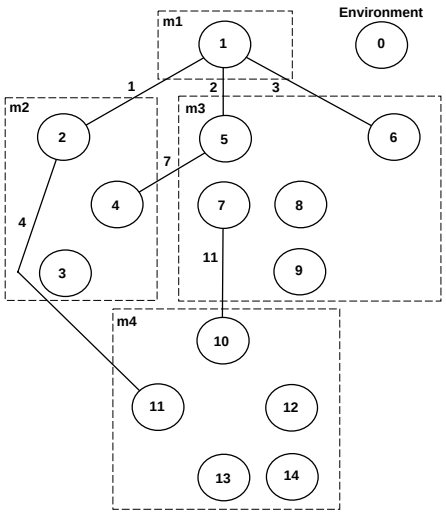
We choose this subgraph for measurement because the properties of coupling in Table 1 focus on intermodule edges. In particular, the Monotonicity Property specifies the incremental effect of adding just one intermodule edge. Conversely, the Disjoint Module Additivity Property requires the absence of intermodule edges between two modules. Moreover, the Merging of Modules Property allows for conversion of an intermodule edge into an intramodule edge upon merger.

In order to analyze the pattern of intermodule relationships, we label each node with the set of edges that are incident to it. A convenient representation of the graph is a nodes $\times$ edges table where each cell indicates whether the node is an end point of the edge, or not. Then each node's label is the binary pattern of values in a row of the table. Figure 2 also shows the nodes $\times$ edges table of S. (Column $p_{L(i)}$ is explained below.) The abstraction of a nodes $\times$ edges table is an instance of an object-predicate table, where nodes are objects and each predicate is of the form, "Is this node related to another node by this edge?" Van Emden shows that object-predicate tables are useful for analysis of complex relationships among objects [33].

In summary, we begin with any protocol that results in a graphical abstraction of a modular system, and then translate that into an nodes $\times$ intermodule-edges table. Because object-predicate tables are suitable for relations in general, future research will extend this to more general measurement protocols [25].

2.2. Measurement instrument

A *measurement instrument* is a tool for mapping an abstraction of software to a number or category [22]. An instrument is usually designed to detect a particular unit of measure, and assures that each unit of an attribute is equivalent within the constraints of measurement error. We adopt information theory [14, 31]



		Edge						
Module	Node	1	2	3	4	7	11	$p_{L(i)}$
Environ.	0	0	0	0	0	0	0	0.467
m_1	1	1	1	1	0	0	0	0.067
m_2	2	1	0	0	1	0	0	0.067
	3	0	0	0	0	0	0	0.467
	4	0	0	0	0	1	0	0.067
m_3	5	0	1	0	0	1	0	0.067
	6	0	0	1	0	0	0	0.067
	7	0	0	0	0	0	1	0.133
	8	0	0	0	0	0	0	0.467
	9	0	0	0	0	0	0	0.467
m_4	10	0	0	0	0	0	1	0.133
	11	0	0	0	1	0	0	0.067
	12	0	0	0	0	0	0	0.467
	13	0	0	0	0	0	0	0.467
	14	0	0	0	0	0	0	0.467

Figure 2. Example Intermodule-Edges Graph

as the basis for our measurement instrument, because we view the design decisions embodied by the graph as information [2].

Software design is an intellectual process. Since human short-term memory capacity is limited [24], we suspect that one important source of mistakes is information overload. A designer may not consider all relevant information when making a decision, because the amount of information is too large to work with in short-term memory. Our working hypothesis is that assessment of designs in terms of amounts of various kinds of information can indicate potential information overload. Information is stored in human short-term and long-term memory when a person comprehends a

design abstraction. The amount of such information may have useful statistical relationships with quality factors [18]. Future empirical research will investigate the relationship between information overload and software quality. Information theory-based modeling methods are promising [1, 20, 21].

Most traditional software metrics count artifact features, as though each item is equal in the mind of a designer. For example, Briand, Daly, and Wüst [10] propose an integrated measurement framework for object-oriented coupling metrics, based on counts. The framework successfully encompasses many metrics proposed in the literature. Each qualifying metric is equivalent to the number of edges of an appropriate graph. We suspect that an approach that is more sophisticated than counting will be useful. Information theory is attractive, because it measures symbolic content in a way that accounts for redundancy and patterns. Regular patterns are easy for designers to remember, but counts do not reflect this. According to information theory, regular patterns have low information content, because the symbolic content can be compactly described [14]. The foundation of our measurement instrument is information theory, rather than just counting.

The notion of coupling is based on relationships among nodes. Our data consists of a nodes $\times$ edges table that represents a subgraph, S, with $n+1$ nodes. Let us model the graph S as a probability distribution on the labels (row patterns), estimated by the proportions of distinct labels, $p_l, l = 1, \ldots, n_S$, where n_S is the number of distinct labels [33]. *Entropy* is the average information per node. The entropy of the distribution of node labels [31] is the following.

$$H(S) = \sum_{l=1}^{n_S} (-p_l \log p_l) \quad (1)$$

$$H(S) = \sum_{i=0}^{n} \frac{1}{n+1} \left(-\log p_{L(i)}\right) \quad (2)$$

where all logarithms are base two, and $L(i)$ is a function that gives the pattern index, l, of the i^{th} row. Figure 2 lists $p_{L(i)}$ for each row of the example; each $p_{L(i)}$ is the proportion of that row's pattern out of fifteen rows. The unit of measure is a bit. Entropy is always nonnegative. The example in Figure 2 has $H(S) = 2.46$ bits per node.

If a graph S_0 has no edges, its nodes $\times$ edges table is equivalent to one with an arbitrary number of columns and zero in all cells, and thus, all patterns are the same (zeros), $p_l = 1$ and $H(S_0) = 0$.

We multiply entropy by the number of nodes to calculate the estimated *minimum description length* [14].

$$I(S) = (n+1)\, H(S) \quad (3)$$

$$I(S) = \sum_{i=0}^{n} \left(-\log p_{L(i)}\right) \quad (4)$$

The minimum description length, $I(S)$, is the total amount of information in the structure of the graph. The example in Figure 2 has $I(S) = 36.95$ bits.

Consider the subgraph, S_i, consisting of all the nodes in S and the edges of S that have the i^{th} node as an end point. Disconnected nodes are included in this subgraph. Similar to S, we label each node with the set of edges incident to it. We also model S_i as a probability distribution on the labels, estimated by the proportions of distinct labels, p_l. Similar to Equation (2), the entropy of the distribution of node labels is the following.

$$H(S_i) = \sum_{j=0}^{n} \frac{1}{n+1} \left(-\log p_{L_i(j)}\right) \quad (5)$$

where $L_i(j)$ is a function that gives the pattern index, l, of the j^{th} row of S_i. Like $H(S)$, the unit of measure is a bit. The minimum description length of S_i is the subgraph's total information.

$$I(S_i) = \sum_{j=0}^{n} \left(-\log p_{L_i(j)}\right) \quad (6)$$

Because each row of each S_i is a subset of the corresponding row of S, S represents the joint distribution of all the S_i. Information theory states that the entropy of a joint distribution is less than or equal to the sum of the entropy of the components.

$$\sum_{i=0}^{n} H(S_i) \geq H(S) \quad (7)$$

Watanabe shows that the difference of the two sides of this inequality is a measure of the relationships among the components [35].

$$C(S) = \sum_{i=0}^{n} H(S_i) - H(S) \quad (8)$$

Van Emden calls this *excess entropy* [33]. Excess entropy is the average information in relationships. Excess entropy is always nonnegative. This supports the Nonnegativity Property in Table 1. The excess entropy of a graph with no edges is zero, supporting the Null Value Property in Table 1. The end points are related by the presence of an edge and the other nodes are related by the absence of that edge. If the S_i are highly correlated with each other by many edges and many common disconnected nodes, then the excess entropy

will be high. The example in Figure 2 has $C(S) = 3.82$ bits per node.

Multiplying excess entropy by the number of nodes yields the estimated minimum description length [14] of the relationships, in other words, the total information in the relationships.

$$(n+1)C(S) = \sum_{i=1}^{n} I(S_i) - I(S) \qquad (9)$$

Note that $I(S_0) = 0$, because a disconnected node, such as the environment node, $i = 0$, has no subgraph S_i. Because it is an information theory-based measure, total information accounts for patterns in relationships, and consequently, discriminates among graph structures in a way that counting measures cannot.

2.3. Coupling of a modular system

Given a modular system, MS, and its intermodule-edges subgraph, S, the intermodule *coupling of a modular system* is the minimum description length of the relationships in S.

$$Coupling(MS) = \sum_{i=1}^{n} I(S_i) - I(S) \qquad (10)$$

By Equation (9), this is equivalent to the excess entropy of S times the number of nodes, $n + 1$. In [5], we have proven that $Coupling(MS)$ has the properties specified in Table 1, and thus, is a member of the family of measures called "coupling of a modular system". The example in Figure 2 has $Coupling(MS) = 57.28$ bits.

3. Intramodule Coupling

Intermodule coupling, discussed above, measures attributes of intermodule relationships. We propose a family of measures called *intramodule coupling of a modular system* which is closely related to the coupling family above.

We transformed the properties of intermodule coupling in Table 1 into properties of intramodule coupling. To conserve space, the following is a summary of the differences. Given the properties in Table 1, (1) we substitute *IntramoduleCoupling* for *Coupling*; (2) we substitute "intra-" for "inter-"; and (3) we change the direction of the inequality in the Merging of Modules Property, so that instead of "not greater than", it reads "not less than". This inequality accounts for the possibility that an intermodule edge between two modules may become an intramodule edge when they are merged. Otherwise, the properties of *IntramoduleCoupling* are the same as those of *Coupling* of a modular system.

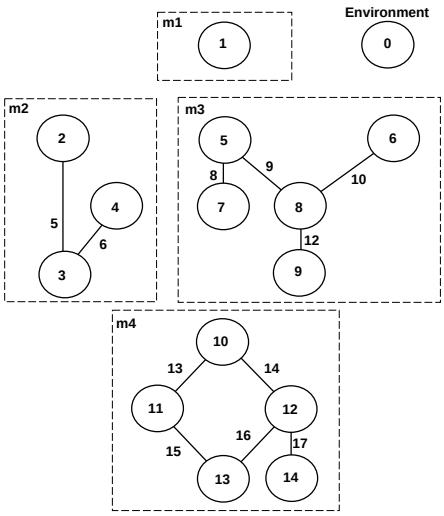


Figure 3. Example Intramodule-Edges Graph

Intramodule coupling is applicable to the same software engineering abstractions as intermodule coupling. Our measurement protocol for intramodule coupling is similar to intermodule coupling's protocol, except that we consider intramodule edges instead of intermodule edges. Given a modular system, MS, represented by a graph with n nodes partitioned into modules, and its environment, represented by one additional disconnected node, consider the subgraph, S', consisting of all nodes in MS plus the environment node, and all intramodule edges. The end points of an intramodule edge are both in the same module. Without loss of generality, index the environment node $i = 0$, and the nodes in MS as $i = 1, \ldots, n$. Like intermodule coupling, we use nodes $\times$ edges tables as a convenient representation. For example, Figure 3 depicts the intramodule edges subgraph, S', for the modular system in Figure 1, and Table 2 shows its nodes $\times$ edges table, and the proportion of each row's pattern, $p_{L(i)}$. Our measurement instrument for intramodule coupling is the same as for intermodule coupling, based on information theory.

The intramodule coupling of a modular system, MS, is the minimum description length of the relationships in S'.

$$IntramoduleCoupling(MS) = \sum_{i=1}^{n} I(S'_i) - I(S') \qquad (11)$$

By Equation (9), this is equivalent to the excess entropy of S' times the number of nodes, $n + 1$. In [4],

Table 2. Example Intramodule Edges Graph

		Edge											
Module	Node	5	6	8	9	10	12	13	14	15	16	17	$p_{L(i)}$
Environ.	0	0	0	0	0	0	0	0	0	0	0	0	0.133
m_1	1	0	0	0	0	0	0	0	0	0	0	0	0.133
m_2	2	1	0	0	0	0	0	0	0	0	0	0	0.067
	3	1	1	0	0	0	0	0	0	0	0	0	0.067
	4	0	1	0	0	0	0	0	0	0	0	0	0.067
m_3	5	0	0	1	1	0	0	0	0	0	0	0	0.067
	6	0	0	0	0	1	0	0	0	0	0	0	0.067
	7	0	0	1	0	0	0	0	0	0	0	0	0.067
	8	0	0	0	1	1	1	0	0	0	0	0	0.067
	9	0	0	0	0	0	1	0	0	0	0	0	0.067
m_4	10	0	0	0	0	0	0	1	1	0	0	0	0.067
	11	0	0	0	0	0	0	1	0	1	0	0	0.067
	12	0	0	0	0	0	0	0	1	0	1	1	0.067
	13	0	0	0	0	0	0	0	0	1	1	0	0.067
	14	0	0	0	0	0	0	0	0	0	0	1	0.067

we have proven that this measure has properties corresponding to those specified in Table 1, and thus, is a member of the family of measures called "intramodule coupling of a modular system". The example in Figure 3 has $IntramoduleCoupling(MS) = 113.40$ bits.

4. Cohesion

Briand, Morasca, and Basili [11] propose a set of properties that defines the concept *cohesion of a modular system*. Table 3 summarizes the properties of this family of measures [11]. We reserve the term *cohesion of a modular system* for measures that have these properties.

Intramodule coupling, discussed above, measures information in intramodule relationships. Intramodule coupling and cohesion are measures of different attributes of relationships within modules; intramodule coupling is a quantity of bits, but cohesion is a normalized fraction. Cohesion is applicable to the same software engineering abstractions as intramodule coupling.

Our measurement protocol for cohesion is the same as for intramodule coupling. Given a modular system, MS, represented by a graph with n nodes, the intramodule-edges subgraph, S', is defined as above. Consider the modular system $MS^{(n)}$, consisting of a complete graph of n nodes in one module, where every node is connected to every other node. In effect, $MS^{(n)}$ is the "most cohesive" system possible with n nodes. Also, consider the subgraph, $S^{(n)}$, derived from $MS^{(n)}$.

Table 3. Properties of Cohesion

Concept/Properties
Cohesion of a Modular System:
1. Nonnegativity and Normalization. Cohesion of a modular system belongs to a specified interval, $Cohesion(MS) \in [0, \text{Max}]$.
2. Null value. Cohesion of a modular system is null if its set of intramodule edges is empty.
3. Monotonicity. Adding an intramodule edge to a modular system does not decrease its cohesion.
4. Merging of modules. If two unrelated modules, m_1 and m_2, are merged to form a new module, $m_{1\cup 2}$, that replaces m_1 and m_2, then the cohesion of the modular system with $m_{1\cup 2}$ is not greater than the cohesion of the modular system with m_1 and m_2.

Like the coupling measures, we use nodes $\times$ edges tables as a convenient representation for S' and $S^{(n)}$. For example, Figure 3 depicts the intramodule-edges subgraph, S', for the modular system in Figure 1.

As with coupling, most measures of cohesion in the literature are based on counts of artifact features, as though each feature is equal in the mind of a designer. For example, Briand, Daly, and Wüst [9] propose an integrated measurement framework for object-oriented cohesion metrics, based on counts. The framework successfully encompasses many metrics, and each qualify-

ing metric is equivalent to the number of edges of an appropriate graph divided by the maximum number of possible edges of interest. As with coupling, we suspect that a more sophisticated approach than counting will be useful. Thus, our measurement instrument for cohesion is also based on information theory, rather than just counting.

Cohesion of a modular system, MS, is the minimum description length of the relationships in S' divided by the minimum description length of the relationships in $S^{(n)}$.

$$Cohesion(MS) = \frac{IntramoduleCoupling(MS)}{IntramoduleCoupling(MS^{(n)})} \quad (12)$$

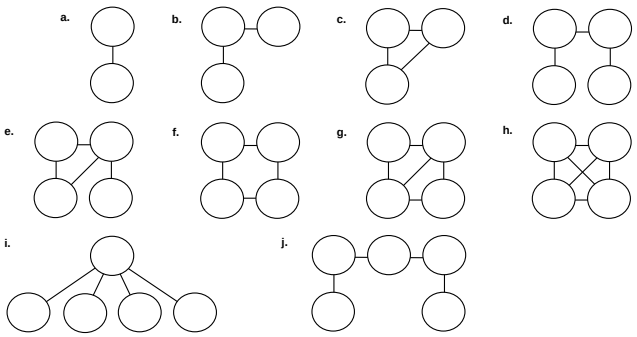
This is equivalent to the ratio of excess entropies, $C(S')/C(S^{(n)})$, because S' and $S^{(n)}$ have the same number of nodes. In [4], we have proven that $C(S^{(n)}) = (n-1)\log(n+1)$ for $n > 2$ and that $Cohesion(MS)$ has the properties specified in Table 3, and thus, is a member of the family of measures called "cohesion of a modular system". The example in Figure 3 has $Cohesion(MS) = 0.149$.

5. Discussion

The information-theory approach to software metrics differs from counting in the way patterns in a graph can have distinct measurements. We model the pattern of edges incident to each node as a random variable. The distribution of that random variable is the basis for calculating entropy, excess entropy, and information. In software engineering terms, we are modeling both the design decisions to connect each pair of nodes, and the decisions not to connect the others. The benefit of this approach is a measure that is sensitive to patterns of connections. This is attractive, because software engineers recognize patterns as well.

To illustrate this, let us take a closer look at some examples of coupling. The number of intermodule edges is a simple measure that conforms to Briand, Morasca, and Basili's definition of coupling [11]. Figure 4 shows several example graphs and their measurement for $Coupling(MS)$ when each node is considered a module, and thus all edges shown are intermodule edges. (The environment node is assumed, but not shown.) The metric shows good discrimination among similar graphs compared to the number of edges.

Consider examples i. and j. in Figure 4. Both graphs have the same number of intermodule edges. However, $Coupling(MS_i)$ has a lower value than $Coupling(MS_j)$ due to the fact MS_i has a more extensive pattern of nodes with only one incident edge, which implies a slightly shorter minimum description length. An



Note: environment node is assumed.

Graph	Nodes	Edges	$Coupling(MS)$
a.	2	1	2.76
b.	3	2	8.00
c.	3	3	16.00
d.	4	3	17.32
e.	4	4	24.07
f.	4	4	26.83
g.	4	5	30.83
h.	4	6	34.83
i.	5	4	22.04
j.	5	4	27.78

Figure 4. Examples

information-theory approach to measuring coupling is able to discriminate between some graphs that the number of intermodule edges cannot, due to the pattern of edges.

The same conclusions apply to $IntramoduleCoupling(MS)$ and $Cohesion(MS)$. However, space does not allow for additional examples here. Preliminary analysis explored examples of cohesion [3].

Kitchenham, Pfleeger, and Fenton delineate various properties that a theoretically valid software metric should have [22]. Following their criteria, it can be shown that: (1) different modular systems may have different amounts of $Coupling(MS)$; (2) $Coupling(MS)$ represents relationships among modules; (3) the unit of measure for $Coupling(MS)$ is a bit, which is an accepted unit of information, and different bits of information are considered equivalent when analyzing information overload; (4) different modular systems are allowed to have the same value of $Coupling(MS)$; (5) $Coupling(MS)$ is compatible with the ratio scale-type, and the measurement algorithm is a valid instrument for an information theory-based measure; and (6) given a graph, the protocol for abstracting subgraphs is unambiguous and self-consistent. Thus, $Coupling(MS)$ is a theoretically valid mea-

sure. *IntramoduleCoupling*(*MS*) is also a theoretically valid measure, due to its similarity to *Coupling*(*MS*). *Cohesion*(*MS*) is a dimensionless ratio of bits. It can be shown that *Cohesion*(*MS*) is also a theoretically valid measure, because *IntramoduleCoupling*(*MS*) is the numerator, and the denominator is a function of the number of nodes only.

6. Conclusions

Graphs are widely used in software architecture and design to depict many abstractions of software. For example, call graphs have been used by software developers for many years. The coupling and cohesion of a graph are just two attributes among many that should be considered when evaluating the quality of a design.

We define metrics that conform to the properties proposed by Briand, Morasca, and Basili [11], and therefore, we call them (intermodule) *coupling of a modular system*, *intramodule coupling of a modular system*, and *cohesion of a modular system*. The definitions are based on information theory, taking patterns of relationships into account, rather than just counts of graph attributes. Because excess entropy measures the average connectivity of a system per node, we define both kinds of coupling as the excess entropy times the number of nodes in the abstraction, which is the total information in the relationships. Cohesion is based on intramodule coupling, normalized to between zero and one.

These measures may be applied to any graph, and thus, are suitable for many software engineering abstractions. The practical interpretation of each measure depends on the underlying abstraction. Accordingly, the usefulness of each measure depends on the relevance of the abstraction to the development process.

The number of edges is also a measure that conforms to the coupling properties proposed by Briand, Morasca, and Basili [11]. The number of edges has been used extensively as a measurement instrument for coupling and cohesion in the software metrics literature. Our preliminary analysis indicated that our information-theory approach makes finer-grain distinctions among alternative graphs than counting.

Future research will validate the usefulness of coupling and cohesion in the context of models that predict software quality [30]. Future research will also apply information theory to other families of metrics such as size, length, and complexity, and various module-level metrics [11].

Acknowledgments

We thank Ye Chen for helpful discussions. We thank the anonymous reviewers for their thoughtful comments. This work was supported in part by the National Science Foundation grant CCR-9803853.

References

[1] E. B. Allen. *Information Theory and Software Measurement*. PhD thesis, Florida Atlantic University, Boca Raton, FL, Aug. 1995. Advised by Taghi M. Khoshgoftaar.

[2] E. B. Allen and T. M. Khoshgoftaar. Measurement of software design coupling. In H. Pham, editor, *Proceedings: Third ISSAT International Conference on Reliability and Quality in Design*, pages 247–253, Anaheim, CA, Mar. 1997. International Society of Science and Applied Technologies.

[3] E. B. Allen and T. M. Khoshgoftaar. Measurement of software design cohesion. In H. Pham, editor, *Proceedings: Fifth ISSAT International Conference on Reliability and Quality in Design*. International Society of Science and Applied Technologies, Aug. 1999. Invited paper. Forthcoming.

[4] E. B. Allen, T. M. Khoshgoftaar, and Y. Chen. Properties of cohesion of graph abstractions of software. Technical Report TR-CSE-99-5, Florida Atlantic University, Boca Raton, FL USA, Apr. 1999.

[5] E. B. Allen, T. M. Khoshgoftaar, and Y. Chen. Properties of coupling for graph abstractions of software. Technical Report TR-CSE-96-41, Florida Atlantic University, Boca Raton, FL, June 1999. Revised.

[6] V. R. Basili, L. C. Briand, and W. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, Oct. 1996.

[7] J. M. Bieman and B.-K. Kang. Measuring design-level cohesion. *IEEE Transactions on Software Engineering*, 24(2):111–124, Feb. 1998.

[8] J. M. Bieman and L. M. Ott. Measuring functional cohesion. *IEEE Transactions on Software Engineering*, 20(8):644–657, Aug. 1994.

[9] L. C. Briand, J. W. Daly, and J. Wüst. A unified framework for cohesion measurement in object-oriented systems. In *Proceedings of the Fourth International Symposium on Software Metrics*, pages 43–53, Albuquerque, NM USA, Nov. 1997. IEEE Computer Society.

[10] L. C. Briand, J. W. Daly, and J. K. Wüst. A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, 25(1):91–121, Jan. 1999.

[11] L. C. Briand, S. Morasca, and V. R. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–85, Jan. 1996. See comments in [12, 28, 37].

[12] L. C. Briand, S. Morasca, and V. R. Basili. Response to: Comments on "Property-based software engineering measurement": Refining the additivity properties. *IEEE Transactions on Software Engineering*, 23(3):196–197, Mar. 1997. See [11, 28].

[13] H. S. Chae and Y. R. Kwon. A cohesion measure for classes in object-oriented systems. In *Proceedings Fifth International Software Metrics Symposium*, pages 158–166, Bethesda, MD USA, Nov. 1998. IEEE Computer Society.

[14] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[15] H. Dhama. Quantitative models of cohesion and coupling in software. In *Proceedings of the Annual Oregon Workshop on Software Metrics*, Silver Falls, OR, Apr. 1994. Oregon Center for Advanced Technology Education, Portland State University.

[16] R. Harrison, S. J. Counsell, and R. V. Nithi. Coupling metrics for object-oriented design. In *Proceedings Fifth International Software Metrics Symposium*, pages 150–157, Bethesda, MD USA, Nov. 1998. IEEE Computer Society.

[17] R. Harrison, S. J. Counsell, and R. V. Nithi. An evaluation of the MOOD set of object-oriented software metrics. *IEEE Transactions on Software Engineering*, 24(6):491–496, June 1998.

[18] L. Hatton. Reexamining the fault density — component size connection. *IEEE Software*, 14(2):89–97, Mar. 1997.

[19] T. M. Khoshgoftaar and E. B. Allen. Applications of information theory to software engineering measurement. *Software Quality Journal*, 3(2):79–103, June 1994.

[20] T. M. Khoshgoftaar and E. B. Allen. An information theoretic approach to predicting software faults. *International Journal of Reliability, Quality and Safety Engineering*, 5(3):227–248, Sept. 1998.

[21] T. M. Khoshgoftaar, E. B. Allen, and D. L. Lanning. An information theory based approach to quantifying the contribution of a software metric. *Journal of Systems and Software*, 36(2):103–113, Feb. 1997.

[22] B. A. Kitchenham, S. L. Pfleeger, and N. E. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, Dec. 1995. See comments in [23, 26].

[23] B. A. Kitchenham, S. L. Pfleeger, and N. E. Fenton. Reply to: Comments on "Towards a framework for software measurement validation". *IEEE Transactions on Software Engineering*, 23(3):189, Mar. 1997. See [22, 26, 36].

[24] G. A. Miller. The magical number 7 plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1957.

[25] S. Morasca and L. C. Briand. Towards a theoretical framework for measuring software attributes. In *Proceedings of the Fourth International Symposium on Software Metrics*, pages 119–126, Albuquerque, NM USA, Nov. 1997. IEEE Computer Society.

[26] S. Morasca, L. C. Briand, V. R. Basili, E. J. Weyuker, and M. V. Zelkowitz. Comments on "Towards a framework for software measurement validation". *IEEE Transactions on Software Engineering*, 23(3):187–188, Mar. 1997. See [22, 36].

[27] D. L. Parnas. On the criteria to be used in decomposing systems. *Communications of the ACM*, 15(12):1053–1058, Dec. 1972.

[28] G. Poels and G. Dedene. Comments on "Property-based software engineering measurement": Refining the additivity properties. *IEEE Transactions on Software Engineering*, 23(3):190–195, Mar. 1997. See [11].

[29] R. S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, 1992.

[30] N. F. Schneidewind. Methodology for validating software metrics. *IEEE Transactions on Software Engineering*, 18(5):410–422, May 1992.

[31] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL, 1949.

[32] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik. Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering*, 21(4):314–335, Apr. 1995.

[33] M. H. van Emden. Hierarchical decomposition of complexity. *Machine Intelligence*, 5:361–380, 1970. See also [34] for details.

[34] M. H. van Emden. *An Analysis of Complexity*. Number 35 in Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam, 1971.

[35] S. Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1):66–82, Jan. 1960.

[36] E. J. Weyuker. Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 14(9):1357–1365, Sept. 1988.

[37] H. Zuse. Reply to: "Property-based software engineering measurement". *IEEE Transactions on Software Engineering*, 23(8):533, Aug. 1997. See [11].