3-1-2006

The Cohesion-Based Requirements Set Model for Improved Information System Maintainability

Nelson Barnes Jr.
nbarnes@cba.ua.edu

David P. Hale

Joanne E. Hale

THE COHESION-BASED REQUIREMENTS SET MODEL FOR IMPROVED INFORMATION SYSTEM MAINTAINABILITY

Nelson Barnes, Jr.
The University of Alabama
nbarnes@cba.ua.edu

David P. Hale
The University of Alabama
dhale@cba.ua.edu

Joanne E. Hale
The University of Alabama
jhale@cba.ua.edu

Abstract

The concept of cohesion, which is normally associated with software design, is commonly used to measure the degree to which elements of a module are related. Systems constructed adhering to the principle of cohesion are expected to be more maintainable. It is proposed in this research that it may be more advantageous to apply the principle of cohesion at an earlier phase of the software development life cycle, thus placing more responsibility on the analyst who has a better understanding of the business. This paper proposes the Cohesion-Based Requirements Set (CBRS) model for improved information system maintainability. Using the CBRS technique, one may be able to positively affect the overall maintainability of the resulting system by applying a synthesis or expansion approach when gathering requirements rather than using an approach based on analysis or reduction.

Keywords: Cohesion, requirements structuring, requirements validation, requirements engineering

Introduction

Cohesion has been commonly defined as a measure of the degree to which elements of a module belong together (Mallens, 1997). More specifically, it describes the degree to which the tasks performed by a single module are functionally related. The concept of cohesion is normally considered during software design and development, during which developers strive for components which are highly cohesive (Beiman and Kang, 1995). The cohesiveness of the system is frequently used to measure characteristics such as the separation of responsibilities of the components, the independence of the components, the quality of the software design, the fault proneness of the system, the modularization of the components, and the amount of reuse that a component provides (Etzkorn et al, 1997; Haley et al, 2004; Lee et al, 2001; Mallens, 1997). This article proposes applying these same general principles of cohesion at an earlier phase of the software development life cycle. By doing this, the decision of determining the level cohesion in the system modules becomes the responsibility of the system analyst, who has more knowledge of the work system and thus has the ability predict what requirements are more likely to change, instead of the software designer who has less contextual knowledge of the domain. Using this technique, the analyst may be able to positively affect the overall maintainability of the system by applying a synthesis or expansion approach when gathering, structuring, and communicating requirements rather than using an approach based on analysis or reduction.

Background

Cohesion

Cohesion is a qualitative measure which falls into one of the following categories in order of lowest to highest: *Coincidental, Logical, Temporal, Procedural, Communication, Sequential, and Functional* (Yourdon and Constantine, 1979). High cohesion leads to code that is easier to comprehend. Comprehension is the often cited as the most time consuming component of the maintenance activity. In object-oriented programming, developers normally assign responsibilities to classes with the main objective of keeping the level of cohesion high. This increases the likelihood of reuse and creates components whose complexity is kept manageable. The level of cohesion decreases if the responsibilities of a class are unrelated -- in other words, they perform a wide range of disparate actions or operate on diverse types of data.

Although significant progress has been made in the area of developing more extensive metrics for measuring the cohesion of modules during the design phase, no one cohesion method currently exists that is accepted as a standard. Under the category of structural cohesion metrics, Chidamber and Kemerer (1994) originally proposed LCOM (Lack of Cohesion of Methods) which has become one of the most widely known cohesion metrics for object-oriented systems. Subsequent research has produced various similar metrics such as LCOM2, LCOM3, LCOM4, and LCOM5 (Beiman and Kang, 1995; Chidamber and Kemerer, 1994; Hitz and Montazeri, 1995; Lee et al, 2001), as well as TCC (Tight Class Cohesion) and LCC (Loose Class Cohesion) (Beiman and Kang, 1995). Other cohesion metrics that have been proposed include specialized metrics such as ontology cohesion metrics (Yao et al, 2005), knowledge-based metrics (Kramer and Kaindl, 2004), and metrics for distributed systems (Cho et al, 1998). While these metrics have proven to be effective for measuring modules during the design phase, no research was found that took the notion of cohesion and synthesized it to an earlier phase in the life cycle.

Requirements Structuring

The significance of representing requirements so that the system behaves and evolves as intended has increased as systems have become more integrated and complex. The task of effectively organizing and categorizing requirements can be surprisingly complex in a fluid and dynamic environment (Haley et al, 2004). The difficulty involved in developing a powerful yet flexible classification scheme is partly due to the heterogeneity of factors usually considered during the requirements elicitation phase.

Requirements structuring mechanisms include techniques such as use cases, creating a system of subsystems, decision trees, and Entity Relationship Diagrams (ERD). In addition, techniques such as Rapid Application Development (RAD) and Object-Oriented Analysis (OOA) have requirements structuring and analysis techniques embedding in the frameworks. Although each of these techniques are effective within the frameworks in which they are used, no common as well as flexible architecture or tools exist for structuring requirements.

Requirements Classification and Mapping

The goal of this research stream is to determine if applying the concept of cohesion to requirements gathering, structuring, and communication results in improved system maintainability. In order to do this, the patterns related to how requirements are defined must first be identified and then mapped to the principles of software-based cohesion. Business information system functional requirements can be categorized as (Davis, 1990; Ferdinandi, 2001; Pressman, 1992; Von Halle, 2001):

Validation – Ranges of values, optional values, and mandatory values.

Restriction/Constraint – Limits for the data that is stored in the system.

Action/Processing – The order or sequence of the algorithm for a set of tasks the system must perform.

Presentation – How work and tasks are organized as well as how the system should appear to the user.

Behavioral – How the system should perform in certain situations and what is automated.

Deductive – How information within the system should be calculated or derived.

These categories cover the majority of possible types of requirements. During the requirements elicitation phase, it is imperative for the analyst to obtain additional information about each requirement. This meta-data is necessary to ensure that the requirements are correctly classified.

Using the concept of modularization, the objective is to construct comprehensive functional Requirements Sets that provide an integrated means for improved maintainability. To do this, the different classes of requirement types must be accurately mapped to cohesion levels that closely define the relationships, behaviors, or characteristics that are desired. Table 1 maps Requirements Sets based on the defined requirements categories to the cohesion levels. It is important to note that this mapping represents a typical classification of most requirements within a specific category; it is not expected to be accurate for all functional requirements. Logical and Coincidental cohesion are explicitly not included because they typically should be avoided due to their weak association nature.

To better understand the logic behind the mappings, consider both the requirement type and its paired cohesion level. For example, processing requirements specify the algorithm for a task or set of tasks the system must perform. Grouping requirements based on common processing most closely resembles the characteristics desired when designing a system with a high degree of procedural cohesiveness. Therefore, all related requirements of this type will grouped into procedural sets. Another example can be found with presentation requirements. Creating Requirements Sets based on common presentation would entail structuring requirements according to how tasks are organized. This structuring mechanism is similar to the objective of temporal cohesiveness in which elements that are processed with a certain time period are grouped together. Therefore, using this principle alone, all related requirements of this type will grouped into temporal sets. By mapping the requirement types to the cohesion categories in this manner, a significant step has been taken towards the ultimate goal of structuring the requirements according to the principle of cohesion.

Table 1: Requirement Type to Cohesion Level Mappings

Requirement Type	Cohesion Level
Validation	Communication
Restriction/Constraint	Communication
Action/Processing	Procedural
Presentation	Temporal
Behavioral	Functional
Deductive	Sequential

The Cohesion-Based Requirements Set Model

After successfully developing a classification for the different types of requirements, the next step is to develop a scheme for structuring the functional requirements. Currently, neither an overall architectural rationale nor a business architecture driven approach for structuring requirements exists for the analyst. This research begins to address this gap with the proposed Cohesion-Based Requirements Set (CBRS) model. The most general CBRS model is depicted in Figure 1 as a non-directed graph in which each requirement is represented as a node and the arcs represent potential binary precedence. Underlying the proposed method is the proposition that there exists a logical structuring order of requirements derived from the fully enumerated *general model* that minimizes maintenance costs of the resulting system.

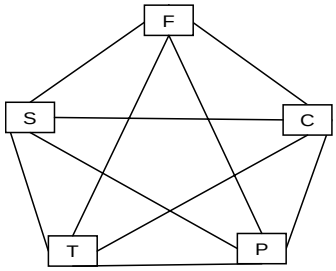

Figure 1: General CBRS Model for Requirements Structuring

The number of different paths through the cohesion-level requirements sets in the general model increases exponentially as the number of requirements increase. One configuration may consist of first grouping the presentation requirements, then the deductive requirements, followed by the validation requirements, behavioral requirements, restriction/constraint requirements, and finally the action/processing requirements. The chosen configuration for a specified target system would be dependent upon various factors such as application size and type, the amount of code reuse that is desired, the expected requirements volatility, and the application domain.

Experience and intuition may suggest that requirements should be structured in a linear fashion based on the levels of cohesion that are desired between the attributes, methods, and objects during the design phase. An example of this linear sequential CBRS model is shown in Figure 2. In this model, the requirements are structured in a sequential fashion based on the levels of cohesion. If the design goal is to produce a highly cohesive system, then the rules dictate that we should initially begin to combine requirements at the highest level of cohesion and subsequently proceed down the cohesion levels. This model is built on the hypothesis that the relative desirability of design-phase cohesion levels map directly to the requirements phase. In this case, behavioral requirements are grouped and validated first, and then the deductive requirements are added, followed by the validation or restriction/constraint requirements, then the action/processing requirements, and finally the presentation requirements. Although requirements are only added to a set in this sequence, the actual number of each requirement type may vary.

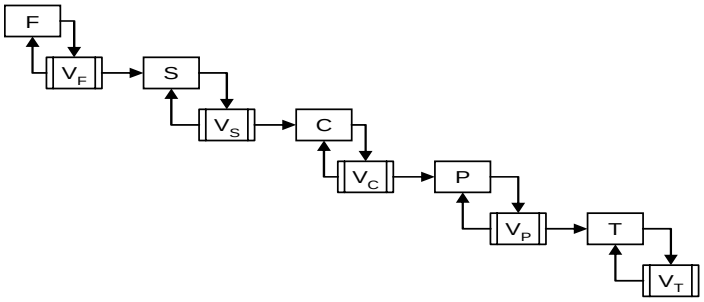

Figure 2: Linear CBRS Model for Requirements Structuring

We have generalized the usage of the linear CBRS model into two distinct techniques which we have designated as Type I and Type II usages. One scenario illustrating how the Type I technique could be applied to the linear CBRS model is shown in Figure 3. In this Type I usage, the first step is to find all requirements that when grouped are related functionally, grouping the related requirements into sets. From Table 1, it follows that this step may focus heavily on the behavioral requirements given that they have been mapped to functional cohesion. The second step is to find all requirements that when grouped are related sequentially, grouping the related requirements into sets. Based on the mappings from Table 1, this would include related deductive requirements. These requirements sets are then added to the appropriate existing functional sets. Similar steps are continued for the next three cohesion levels. Again, Logical and Coincidental cohesion are explicitly not included because they typically should be avoided due to their weak association nature.

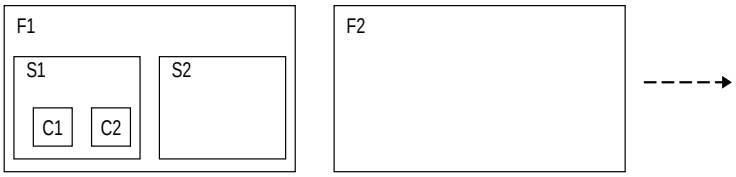

Figure 3: Type I Usage of the Linear CBRS Model

In some situations, the analyst may be unable or unwilling to combine requirements sets. In either case, the analyst would employ the Type II technique which results in groups of tightly related requirement sets that are divided based on cohesion levels. For instance, these modules may be used to design isolated components that are constructed solely for the purpose of modeling a set of volatile requirements for maintenance or testing purposes. With the knowledge that 80% of the total cost of ownership of a system is in maintenance, the analyst may opt to extract certain related requirements and place them into segregate modules. This added insight as to the context in which the system will ultimately function allows the analyst the ability to specify how these volatile requirements are functionally related to the overall system. An example of this case is shown in Figure 4. One final variation on the linear model is a hybrid in which the analyst still follows the order of the linear model, but some requirements types may be skipped or relaxed based on the unique needs of the business.

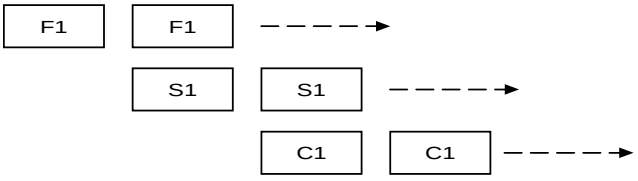

Figure 4: Type II Usage of the Linear CBRS Model

Conclusions and Future Work

This paper presented a conceptual model for structuring requirements using the notion of cohesion levels. The model was derived from the pairings of the requirement types to the cohesion levels. It is proposed that this general model, deemed the Cohesion-Based Requirements Set (CBRS) model, will improve the cohesion at the design stage as well as improve the overall quality of the system in the areas of reuse and maintainability.

The next step in this research is to conduct empirical studies to gain insight into which paths through the general model provide the optimal structure in terms of maximizing component reuse and decreasing maintenance costs when analysts can predict areas in which requirements are more likely to occur. There are numerous sequences that can be constructed through the CBRS model and these planned empirical studies will serve as a tool to determine the most effective sequences. Future phases will empirically identify sequence effectiveness based on project types as well as application domain.

References

Bieman, J. and Kang, B. K. (1995) Cohesion and reuse in an object-oriented system. *Proceedings of ACM Symposium on Software Reusability (SSR'95)*, 259-262.

Chidamber, S. R. and Kemerer, C. F. (1994) A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6), 476-493.

Cho, E. S., Kim, C. J., Kim, D. D., and Rhew, S. Y. (1998). Static and dynamic metrics for effective object clustering. *Proceedings of Asia Pacific International Conference on Software Engineering*, 78-85.

Davis, A. M. (1990) *Software Requirements Analysis and Specification*. Prentice Hall, Englewood Cliffs, NJ.

Etzkorn, L. H., Gholston, S. E., Fortune, J. L., Stein, C. E., Utley, D., Farrington, P. A., and Cox, G. W. (2004) A comparison of cohesion metrics for object-oriented systems. *Information and Software Technology*, 46(10), 677-687.

Ferdinandi, P. L. (2001). Software *Requirements: The Requirements Set*. Addison Wesley, Reading, MA.

Haley, D. T., Nuseibeh, B., Sharp, H. C., and Taylor, J. (2004) The Conundrum of Categorizing Requirements: Managing Requirements for Learning on the Move. *Proceedings of the 12th IEEE International Requirements Engineering Conference.*

Hitz, M. and Montazeri, B. (1995) Measuring Coupling and Cohesion in Object-Oriented Systems. *Proceedings of International Symposium on Applied Corporate Computing*, Monterrey, Mexico.

Kramer, S. and Kaindl, H. (2004). Coupling and cohesion metrics for knowledge-based systems using frames and rules. *ACM Transactions on Software Engineering and Methodology*, 13(3), 332-358.

Lee, J. K., Jung, S. J., Kim, S. D., Jang, W. H., and Ham, D. H. (2001) Component identification method with coupling and cohesion. *Proceedings of Eighth Asia-Pacific Software Engineering Conference*, 79-86.

Mallens, P. (1997). *Business Rules-Based Application Development*. Database Newsletter, 25(1).

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

Yao, H., Orme, A. M., Etzkorn, L. H. (2005). Cohesion Metrics for Ontology Design and Application. *Journal of Computer Science*, 1(1), 107-113.

Yourdon, E.; Constantine, L L. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Prentice-Hall, Englewood Cliffs, NJ.

Von Halle, B. (2001). *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. Wiley, New York.