ISSN 2320 – 2602
Volume 3, No.8, August 2014
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/pdfs/2014/ijacst03382014.pdf

A Design for Problem Decomposition Models In The Development Of Software Intensive System

Saravana Moorthy R[1], Prof.Dr.Ashish Sharma[2]

[1]PhD Scholar; [2]PhD Guide; Bundelkhand University, Jhansi, India

ABSTRACT

Decomposition in computer science is also known as *factoring*, refers to the process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program, and maintain. We formally define the problem of data model decomposition as follows: The initial system (or *problem state*) is a data model (*D*), consisting of a set of entities (*E*) and a set of relationships (*R*). Each relationship in R defines an association between two entities in *E*, although the entities may not be distinct (i.e. recursive relationships are allowed). The terminal system (or *solution state*) is a hierarchy of *n* subject areas (*S*), organized into a finite number of levels (*L1*, *L2*, …). Successive levels in the hierarchy represent increasing levels of abstraction from the original data model.

Keywords: Decomposition, LOC Technique, FP Technique, Cohesion, Intensive systems

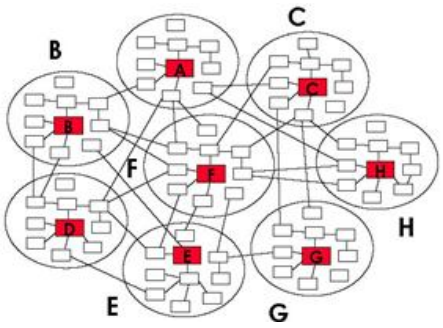

Figure 1.shows Level 1 Decomposition

Each subject area will consist of either a subset of entities in *E* (L1) or a subset of subject areas at the next level down. At the lowest level (*L*1), each subject area is defined as a subset of entities in *E*. Each Level 1 subject area is named after one of the entities it contains, called the *central entity* (see Figure 1). At higher levels, each subject area is a subset of subject areas at the area in L2 is an aggregation of subject areas in L1 and so on.

This results in a hierarchy in which elements at each level are groupings of elements at the next level down This is called a *multi-level structure system* (Klir, 1985) .

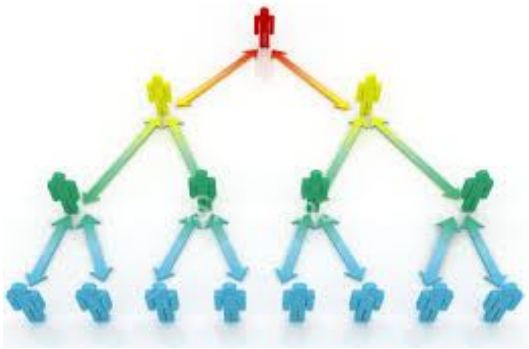

Figure 2 shows. Multi-Level Structure System

1. COMPLETENESS

This principle requires that each entity must be assigned to *at least one* subject area—in other words, the decomposition should *cover* the entities in the underlying model. This principle should be applied at each level of the hierarchy (i.e. each element at a particular level of the hierarchy belongs to at least one subsystem at the next level up).The objective of decomposition is to reduce the complexity of a system while

preserving all information in the original system (Davis and Olson, 1982; Klir, 1985). In the context of data model clustering, this means that all entities and relationships in the original data model should be preserved in the decomposition process. This principle ensures that the decomposition is *lossless* (Weber, 1997).

2. NON-REDUNDANCY

This principle requires that each entity must be assigned to *at most one* subject area. This ensures that subject areas form disjoint subsets of E. This principle should be applied at each level of the hierarchy (i.e. each element at a particular level of the hierarchy belongs to at most one subsystem at the next level up).This principle minimises redundancy between subject areas. This reduces maintenance effort because changes to each entity can be made in a single place. It also improves understanding because overlap between subject areas can lead to confusion in user validation (Moody, 1997).

3.INTEGRATION

This principle requires that each subject area forms a *fully connected sub graph* of the original model (D). This means that each entity on the subject area must be related to all other entities on the subject area via an unbroken sequence of internal relationships. This principle ensures that each subject area forms a fully integrated cluster of entities. This improves under-standability by making sure that each subject area can be understood as an meaningful whole. This principle effectively defines a "minimum cohesion" condition for each cluster (Weber, 1997).

4.UNITY

Each subject area should be named after one of the entities on the subject area, called the central entity. The central entity forms the "nucleus" of the subject area. It helps to ensure the *unity* of the subject area that is, that all entities in the subject area relate to a single business concept or subject. Central entities should be chosen as the entities of greatest business significance to ensure that clusters are as meaningful as possible (Moody, 1997).

We proposed that *connectivity* (the number of relationships an entity participates in) be used as a surrogate measure of business importance. The psychological justification for this is based on two theories of human memory: *semantic network theory* (Collins and Quillian, 1969, 1972) and *spreading activation theory* (Anderson and Pirolli, 1984). According to these theories, semantic memory is structured as a network of related concepts. The concept of spreading activation says that nodes in a semantic network remain in a quiet state until they are activated or "primed". The activation then spreads with decreasing intensity along all pathways connected to the initial node. The level of activation decays exponentially as a function of the distance that it spreads. Spreading activation theory predicts that recall accuracy will be highest and response latency will be lowest for concepts with large numbers of connections to other concepts, because they will receive higher levels of priming. In the case of a data model, entities with large numbers of relationships would therefore be more likely to be recalled. If we assume that "ability to recall" equates to importance, we can conclude that entities with the most relation-ships will also be perceived as the most important. Experimental evidence has confirmed that connectivity is highly correlated with perceived importance (Moody and Flitman, 1999).

5. COGNITIVELY MANAGEABLE

This principle requires that each subject area is of cognitively manageable size. We operationalise this principle by requiring that each subject area consists of a maximum of nine concepts the upper limit of human cognitive capacity. There is universal agreement among cognitive psychologists that due to limits on short term memory, the human mind can only handle "seven plus or minus two" concepts at a time (Miller, 1956; Baddeley, 1994). Once the amount of information exceeds these limits, it must be organised into larger and larger chunks, each containing more information and less detail (Uhr et al, 1962).

Limiting the size of subject areas helps to overcome both the limitations of the human mind in dealing with large amounts of information (*understanding*) and the restrictions of physical sheets of paper (*documentation and maintenance*). If a maximum of nine concepts is used for subject areas at each level, diagrams can be easily drawn on standard sized paper, and the need for reduced fonts and/or crossed lines is virtually eliminated.

6. FLEXIBILITY

An important characteristic of the quality of a decomposition is its flexibility to change. Systems need to adapt to changes over time, and should therefore be organised in a way which is resilient to change (Davis and Olson, 1985; Wand and Weber, 1990; Simon, 1982). Data models tend to increase in size over time, as new requirements are added or the system expands in scope. The partitioning of the data model into subject areas should therefore allow adequate capacity for growth. A data model which consists of subject areas that are all of the maximum size (nine) will have to be repartitioned if even a single entity is added.

We operationalise this principle by requiring that the average size of subject areas is as close as possible to seven entities. This allows, on average, 30% capacity for growth. This reduces the need for future re-partitioning of the model, which in turn simplifies documentation and maintenance. Note that choosing a lower optimal size would reduce the complexity of individual subject areas, but would increase the number of subject areas and the number of levels required. This increases the structural complexity of the model (which is deter-mined by the number of subsystems) and the need to navigate between subject areas.

There is also a strong cognitive justification for using seven as the optimum number of concepts for each sub-ject area. Recall that the limits on short-term memory. This means that some people will have a limit of five concepts, others will have a limit of nine concepts, while most people will be around the average (seven). Therefore to maximise understandability to all people, it is preferable to use the average rather than the upper limit of human cognitive capacity as the optimal size of clusters.

7. EQUAL ABSTRACTION

Another important requirement of a good decomposition is the principle of *equal abstraction* or *balancing* (De Marco, 1978; Klir, 1985; Francalanci and Pernici, 1994). This states that each subsystem should be approxi-mately equal in scope. In the context of a levelled data model, this means that all subject areas should be similar in size. Equal abstraction is an important principle in hierarchical organisation (Klir, 1985). We operationalise this principle by defining the *minimum size* of subject areas as five entities. An alternative metric which could be used is the standard deviation in size of subject areas, but a minimum size constraint is much easier to apply in practice.

8. COUPLING

Coupling is defined as the strength of association *between* different subsystems, and is widely accepted to be one of the most important measures of the quality of a decomposition (Simon, 1982). In the context of data model decomposition, minimising coupling means minimising the number of relationships between entities from different subject areas (called *boundary relationships*).

Coupling should be minimised to increase the independence of the parts of the system (Wand and Weber, 1990; Flood and Carson, 1993). Systems that have low coupling are generally easier to maintain because subsystems can be maintained relatively independently of each other (Yourdon and Constantine, 1979; Davis and Ol-son, 1985; Weber, 1997; Flood and Carson, 1993). The fewer the interactions between subsystems, the less likely changes to one subsystem will affect other subsystems. In addition, minimising coupling improves under-standability by reducing the need to navigate between subject areas.

9. COHESION

The complementary concept to coupling is *cohesion*, which is defined as the strength of association *within* each subsystem. Cohesion should be *maximised*, to increase independence of subsystems. In the context of data model decomposition, maximising cohesion means maximising the number of relationships between entities on the same subject area (called *internal relationships*).

Subsystems which are highly cohesive are likely to be more independent of each other, which simplifies maintenance (Yourdon and Constantine, 1978; Flood and Carson, 1993). It is also believed that subsystems that are highly cohesive are easier to understand. Presumably this is because they can be encoded as a single integrated "chunk" of information rather than a set of relatively independent concepts which must be separately encoded (Eysenck and Keane, 1992; Weber, 1997). Grouping together entities which are strongly related together is likely to result in a unit of information which can be understood as a meaningful whole.

Coupling vs Cohesion

Note that the total cohesion of a decomposition (the number of internal relationships) plus the total coupling of a decomposition (the number of boundary relationships) will always equal the total number of relationships in the model. As a result, increasing coupling will decrease cohesion by an identical amount. Therefore maximising coupling will minimise cohesion, so these two principles are logically *dependent*. As a result, following the rule of parsimony, we can eliminate one of them.

Alternatively, we can combine the two principles together into a new concept called *relative cohesion,* which is the *ratio of cohesion to coupling* of the decomposition (the number of internal relationships divided by the number of boundary relationships in the decomposition). Relative cohesion provides a means of comparing the quality of decompositions independent of the size of the underlying data model. As a general rule, the level of cohesion should be at least twice the level of coupling (internal forces twice as strong as external forces).

There are four different approaches to the sizing problem:

The SLOC technique is language-dependent. The effort required to calculate source lines of code may not be the same for all languages.

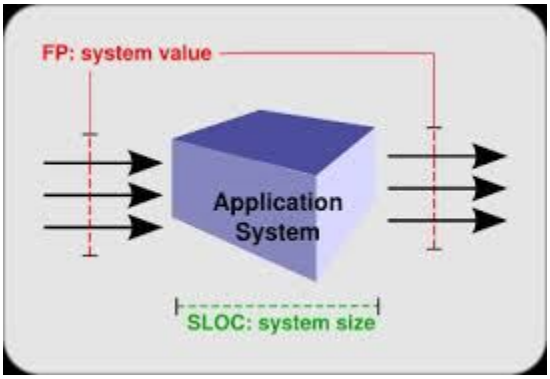

Figure 3 shows App System vs system value

For example, to conceive and write 8 lines of code that accomplish a task in the assembly language may require 15 minutes. However, you may need only five minutes to complete the same lines of code if it is written in Visual Basic.

FP Technique

The FP technique is a direct indicator of the functionality of a software application from the user's perspective. This is the most popular technique used to estimate the size of a software project. This fact is further supported by a quote of Capers Jones, chairman of Software Productivity Research, Inc. in Burlington, Massachusetts, on page 1 of Computer Finance brought out in November 1997. He quotes *"80% of the Fortune 500, are using function points, at least somewhere in their application development organizations".*

By using FP technique to estimate the total size of a project. The total size of a project is estimated as a single FP value. After calculating the total size of a project in FP, you divide the total FP into the different phases of the SDLC. This way, you can determine how much effort per FP is required in that particular phase. For example, the testing phase is planned for 20 FP of work. The project managers, based on their past project experience, determine the amount of effort in man/person months required in the testing phase.

Similarly, you can express the cost required to complete FP of work for a particular phase. At the end of a project, you can also express the number of defects reported in terms of per FP for a phase.

Features of Function Points

The total size of a software project is expressed in total function points. It is independent of the computer language, development methodology, technology, or capability of the project team developing the software project. The specific user functionality of the application is evaluated in terms of relation to what is delivered by the application and, not how it is delivered. Only user-requested and user-defined components are counted. To calculate FP for a project, some major components are required.

function point estimates for a project or a particular phase can be calculated by following four steps:

1.Identify the nonadjustable function points.
2.Calculate total GSC s.
3.Calculate Value Adjustment Factor (VAF)
4.Apply a formula to calculate Adjusted FP

CONCLUSION

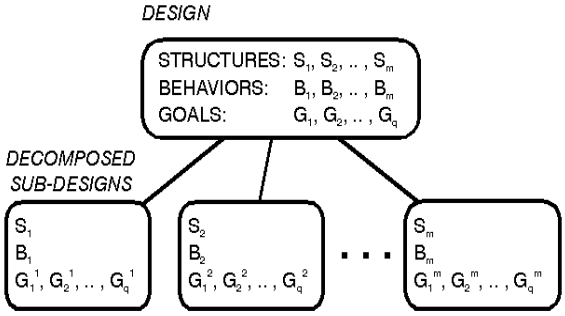

Figure 4 shows Decomposition Designs

The first step in estimation is to predict the size of the project. Typically, this will be done using either LOC (the direct approach) or FP (the indirect approach). Then we use historical data (on similar types of projects) about the relationship between LOC or FP and time or effort to predict the estimate of time or effort for this project. If we choose to use the LOC approach, then we will have to decompose the project quiteconsiderably into as many component as possible and estimate the LOC for each component.

The size *s* is then the sum of the LOC of each component. If we choose to use the FP approach, we don't have to decompose quite so much. In both cases, we make three estimates of size:

sopt an optimistic estimate

sm the most likely estimate

spess an optimistic estimate

and combine them to get a *three-point* or *expected value EV*

$EV = (sopt + 4sm + spess)/6$

EV is the value that is used in the final estimate of effort or time.

REFERENCES

[1] , "A genetic algorithm for scheduling and decomposition of mul-tidisciplinary design problems," *Trans. ASM*E, vol. 118, pp. 486–489,1996.

[2] R. Amen, I. Rask, and S. Sunnersjö, "Matching design tasks to knowl-edge-based software tools—When intuition does not suffice," in *Proc.ASME Design Engineering Technical Conf. (DETC*), Las Vegas, NV,1999.

[3] J. Andersson, "On engineering systems design: A simulation and op-timization approach," M. E. thesis, Linköpings Universitet, Linköping,Sweden, 1999.

[4] S. Austin, A. Baldwin, B. Li, and P. Waskett, "Development of the ADePT methodology: An interim report on the link IDAC 100 project," Loughborough University, Dept. of Civil and Building Engineering ,Loughborough, U. K., 1998.

[5] , "Application of the analytical design planning technique to con-struction project management," *Project Manage. J*., vol. 31, pp. 48–59,2000.

[6] S. Austin, A. Baldwin, and A. Newton, "A data flow model to plan and manage the building design process," *J. Eng. Des*., vol. 7, pp. 3–25,1996.

[7] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modu-larity*. Cambridge, MA: MIT Press, 2000, vol. 1.

[8] O. Becker, J. Ben-Asher, and I. Ackerman, "A method for system inter-face reduction using N charts," *Syst. Eng*., vol. 3, pp. 27–37, 2000.

[9] T. A. Black, C. F. Fine, and E. M. Sachs, "A method for systems designusing precedence relationships: An Application to automotive brake sys-tems,"MIT Sloan School of Management, Cambridge, MA, 3208, 1990.

[10] B. Bolton, "Polyhedral dynamics applied to design and project manage-ment,"*IEE Proc*., vol. 135A, pp. 241–244, 1988.

[11] T. R. Browning, "Systematic IPT integration in lean development pro-grams,"M.S. thesis , MIT, Cambridge, MA, 1996.