ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Automated Methodology to Design a Clustered Class

Ajeet A. Chikkamannur*
Department of Computer Science and Engg.
Sri Venkateshwara College of Engineering
Bangalore,India
ac.ajeet@ gmail.com

Dr. Shivanand M. Handigund
Department of Computer Science and Engg.
Bangalore Institute of Technology
Bangalore,India
smhandigund@gmail.com

Abstract: Object oriented design process for an application is a bottom-up approach in which the class or object of class are designed at the bottom level of design granularity. The current practice of designing a class is modeled with a class diagram by the Unified Modeling Language (UML) but in a design the grouping of an attributes and the functionalities pertaining to a class is made arbitrarily by the intuition of a designer with his/her expertise. The design process depends on the expertise and perception of a designer without any strong foundation, which may lead to the design and development of imperfect information systems. Hence, there is a need of sound and correct design methodology to bifurcate the attributes and the function's signatures applied to an attributes for a class. In the literature, we have not observed any scientific and sound methodology to design implicitly a class with related attributes and function's signature.

This paper proposes a methodology that utilizes the dependency matrix constituted by an attributes and functional dependencies among attributes taken from a Software Requirements Specification (SRS). Then the attributes and functional dependencies of an individual class are structured by the aid of pseudo-transitive axiom and subset theory of mathematics. This procedure is automated to purge the ambiguity of designer(s) decisions and developed based on the harnessing of the axiom and mathematical rigor, which authenticates the sound and correctness.

Keywords: attributes, functional dependency, class, relation, normalization, pseudo-transitive

I. INTRODUCTION

The object oriented design process for an application is a bottom up approach in which the classes or objects of classes are designed at the lowest level of design granularity. The pragmatic practice of designing a class uses four approaches [1], viz. noun phrase approach, common class pattern approach, use case driven approach, and 'classes, responsibilities, and collaborators approach. Then the UML class diagram models the abstracted class. While depicting the UML class diagram, it is observed that the attributes and function's signatures to a particular class is designated by the designer depending on his/her expertise and domain knowledge. This may lead to ambiguity in the design when more number of designers is involved in the decision. Hence, there is a need of sound and correct design methodology for a class to bifurcate the attributes and their function's signature from the group of attributes and functional dependencies.

This paper proposes an automated procedure to cluster the attributes and functional dependencies of a class by blending the good database design principles and the mathematical rigor. The sound theory "normalization" is the way of good database design principle in which the attributes are grouped from the large set of attributes to minimize the redundancy and many researchers [2, 3, 4, 5, 6, 7, 8, 9, 10] have provided the algorithms for the normalization process with mathematical rigor. Further, the authors [20] suggested that the normalization process could be cautiously used to cluster the attributes for a class. At present, in the literature, we have not observed sound and scientifically proved methodology of grouping attributes and the function's signature operating on the grouped attributes.

Hence, a methodology is designed and developed for grouping of an attributes and the functional dependencies are bifurcated pertaining to the grouped attributes, by the blend of second normal form of good database design principle and subset theory of mathematics. Then each categorized group is designated as one class with attributes and the functional dependencies.

II. TAXONOMY

This section discusses the definition and axioms used for the design of methodology

Independence Axiom:

The "independence axiom" states that each function of a system and the design choice that satisfies it should not interfere with other functions of the system. In object technology, this axiom states, "the components are to be maintained independently" [1].

Information Axiom:

The "information axiom" states that when choosing between two designs with similar functional properties, the design with the highest probability of success is the best. In object technology, the axiom states, "the information content of the component design is to be minimized" [1].

Pseudo-transitivity Axiom:

The pseudo-transitivity axiom [2] is equivalent to three axioms viz. reflexivity, augmentation and transitivity. The axiom of pseudo-transitivity between attributes is depicted below.

$$\textbf{If } X \rightarrow Y \textbf{ and } WY \rightarrow Z \textbf{ then } WX \rightarrow Z \qquad \textbf{(1)}$$

Functional Dependency

The relationship among attributes is modeled by the mathematical theory of "functional dependency" (FD). The functional dependency [13, 14] over a set of attributes U asserted from an application is a statement or function, Y = f(X) where X, Y $\subset$ U are attributes sets. A set of non-trivial functional dependencies F represented [17] with following:

$$\mathbf{E = \{(X, Y) \mid Y = f(X) \in F \text{ and } Y \not\subset X\}} \qquad \textbf{(2)}$$

Dependency Matrix:

The "dependency matrix" [15, 17, 18] is matrix representation of attributes and functional dependencies, in which each row i represents the attribute position in the existing functional dependencies and each column j represents a functional dependency. Thus a_{ij} represents the status of attributes in the j^{th} functional dependency such that;

$$a_{ij} = \begin{cases} 1 & \text{If attribute exists in left hand side of fd} \\ 0 & \text{If attribute exists in right hand side of fd} \\ x & \text{otherwise} \end{cases} \qquad \textbf{(3)}$$

III. FRAME WORK

The design of business process in object-oriented paradigm [19, 20] commences with design of class at the lowest level of design granularity in the bottom up approach. The UML provides the modeling of the class by the class diagram perceived, designed and developed by the human expertise. The class diagram consist of three parts, viz. class name, attributes of the class, functions/methods pertaining to the class, which is shown in the figure 1.

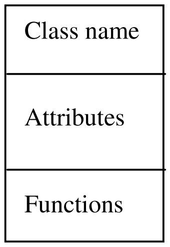

Figure1. UML class diagram

The designer domain knowledge and the expertise assign the contents in the three components of a class. This may lead to wrong judgments or decisions because of the art or skill involved in the bifurcation process from the large set of attributes and functions. Further, the use of human intelligence in the design process will lead to an ambiguity when more number of designers is involved. The view of abstraction of a class, the categorization of attributes and method's signature is dissimilar and herculean task in the large set by the human effort.

Hence, a methodology is developed to group the attributes and their functional dependencies for a clustered class by presuming that the attributes, functional dependencies among the attributes are abstracted from the SRS [14] are available. To group an attributes and their corresponding functional dependencies, a methodology is designed by the aid of "pseudo-transitive axiom". *The axiom is harnessed to identify the pseudo-transitive link between two functional dependencies i.e. the subset relationship between right-hand side attribute of an one functional dependency and the left-hand side attribute set of another functional dependency and the new functional dependency is derived by the substitution. Then the functional dependencies are merged with exclusively identical left-hand side attribute set of another functional dependency to constitute attribute set and the functional dependencies operating on them.*

The stepwise explanation of designed methodology is as follows:

Step 1. The methodology developed by Shivanand et al [14] endows with set of functional dependencies and attributes of an application as an input to our methodology.

Step 2. The functional dependencies are refined to a canonical form, i.e. there is only one attribute on the right-hand side. Then the assignment of each attribute to a column and functional dependency to each row assembles a dependency matrix. By using the symbols of the Boolean algebra, the assignment of an element value is made depending on the type of attribute in a functional dependency, viz. the determinant attribute assigns the value as 1, the dependent attribute assigns the value 0 and the non-existence of attribute assigns the 'x'.

Step 3. The sorting of rows in ascending order is carried out depending on the number of determinant attributes in each functional dependency.

Step 4. In this process, the *pseudo-transitive link* between the two functional dependencies is known by the existence of element values as 1 -1 in different rows, and the same column. The devised method is identifying the condition to provide the true or false state. On the true condition, the determinant attributes of test functional dependency are included in the linking functional dependency by eliminating the linking attribute. For example, consider the functional dependencies $X \rightarrow Y$, $WY \rightarrow Z$ and $WX \rightarrow Z$. The pseudo-transitive link is exists between $X \rightarrow Y$ and $WY \rightarrow Z$ due to the presence of Y as determinant and dependent attribute in two different functional dependencies. Hence, the functional dependency $WY \rightarrow Z$ is revamps to $WX \rightarrow Z$ by substituting the X in a functional dependency $WY \rightarrow Z$.

Step 5. This process recognizes the *exclusively identical* determinant attribute(s) in two or more functional dependencies to take the union of the attribute set of functional dependencies, e.g. Consider the functional dependencies, f_1: $X \rightarrow Y$, f_2: $WX \rightarrow Z$ and f_3: $WY \rightarrow Z$ to illustrate the process. By the execution of step 4 the functional dependency f_2: $WX \rightarrow Z$ is revamping to the f_2: $WY \rightarrow Z$. The functional dependency f_2 and f_3 have the identical determinant attributes and hence, the attributes W, X and Y are grouping for the functional dependencies f_2 and f_3. Then the merged rows are marked as "traced".

Step 6. This process identifies the *subset relationship* between among the determinant attributes of two or more untraced functional dependencies for elimination of dependent attribute from linking functional dependencies, e.g. Consider the functional dependencies f_1: X → S, f_2: WX → ZS for demonstration. The subset relationship among f_1 and f_2 is exists due to the attribute X and hence, the attribute S is discarded from the WX → ZS. The resulting functional dependency is WX → Z.

Step 7. Here the process is assigning the attributes of each untraced rows to the attribute section and the functional dependencies to the method section of a class. Hence, the constitution of number of classes depends on the number of untraced rows of the dependency matrix.

The algorithm for designing class with attributes and their functional dependencies is shown in the figure 2.

```
Input: attributes and functional dependencies
Output: class with attributes and methods

 Represent dependency matrix in accord with
  equation 3
for (i =0; i++; i < no of functional dependencies)
 {Identify pseudo transitive link between i and i+1
   row by element's value 0 in i row and 1 in i+1
   row but in a same column
     if link is existing then
       eliminate attribute causing the link in i+1 row
        & substitute it by LHS attributes of i row
 }
for (i =0; i++; i < no of functional dependencies)
  {Identify exclusively identical set of LHS
    attributes in i and i+1 row
     if link is true then
          group the functions of i and i+1 row
   }
for (i =0; i++; i < no of functional dependencies)
  {Identify subset link among LHS attributes in i
     and i+1 row
     if link is true then
          discard the RHS attribute in i+1 row
          corresponding to RHS attribute of ith row
   }
Construct a class corresponding to each functional
dependency's attributes and function's signature
with merged functional dependencies.
```

Figur 2. Algorithm to construct a class

IV. CASE STUDY

Consider the attributes property No (A), iDate (B), iTime (C), pAddress (D), comments (E), staffNo (F), sName (G) and carReg (H) and the functional dependency among attributes are F_1: A→D, F_2: F→G, F_3: BF→H F_4: AB→CEFGH and F_5:BCH→ADEFG [16, 17, 18]. The dependency matrix for the attributes and functional dependencies is depicted in the figure 3.

The pseudo-transitive link exists among {4, 9, 10, 11, 12, 13}, {8, 9, 10, 11, 12, 13} and {6, 2, 3}. The attribute C is discarded by revamping the element's value as 'x' and substituting the element's value as '1' for attributes A, B in rows {9, 10, 11, 12, 13}. Similarly for the case {8, 9, 10, 11, 12, 13} the attribute F is discarded by changing element's value as 'x' attributes A, B is added to the rows 9, 10, 11, 12, 13 by changing the element's value as '1'. The revamped dependency matrix is shown in figure 4.

	A	B	C	D	E	F	G	H	
1	1	x	x	0	x	x	x	x	F_1
2	x	x	x	x	x	**1**	0	x	F_2
3	x	1	x	x	x	**1**	x	0	F_3
4	1	1	**0**	x	x	x	x	x	F_4
5	1	1	x	x	0	x	x	x	F_4
6	1	1	x	x	x	**0**	x	x	F_4
7	1	1	x	x	x	x	0	x	F_4
8	1	1	x	x	x	x	x	**0**	F_4
9	0	1	**1**	x	x	x	x	**1**	F_5
10	x	1	**1**	0	x	x	x	**1**	F_5
11	x	1	**1**	x	0	x	x	**1**	F_5
12	x	1	**1**	x	x	0	x	**1**	F_5
13	x	1	**1**	x	x	x	0	**1**	F_5

Figure3. Dependency Matrix of attributes and functional dependency

	A	B	C	D	E	F	G	H	
1	1	x	x	0	x	x	x	x	F_1
2	1	1	x	x	x	**x**	0	x	F_2
3	1	1	x	x	x	**x**	x	0	F_3
4	1	1	**0**	x	x	x	x	x	F_4
5	1	1	x	x	0	x	x	x	F_4
6	1	1	x	x	x	**0**	x	x	F_4
7	1	1	x	x	x	x	0	x	F_4
8	1	1	x	x	x	x	x	0	F_4
9	**1**	1	**x**	x	x	x	x	**x**	F_5
10	**1**	1	**x**	0	x	x	x	**x**	F_5
11	**1**	1	**x**	x	0	x	x	**x**	F_5
12	**1**	1	**x**	x	x	0	x	**x**	F_5
13	**1**	1	**x**	x	x	x	0	**x**	F_5

Figure 4. Dependency Matrix of attributes and functional dependency

A	B	C	D	E	F	G	H	
1	x	x	0	x	x	x	x	F_1
1	1	0	**x**	0	0	0	0	F_2.F_3, F_4, F_5

Figure 5. Result Dependency Matrix

Then the functional dependencies with exclusively identical left-hand attributes are merged. Hence, the attributes of functional dependencies from 2 to 13 are grouped to form a set of attributes. The resulting matrix is shown in the figure 5.

There is a subset link between two rows of result dependency matrix shown in figure 5 by the attribute A. The attribute A determines the attribute D and hence it is to be discarded in the other functional dependency. This is depicted by marking element's value as 'x' for attribute D in the second row. Then the set attributes and functional dependencies corresponding to each row is utilized to constitute a class. The resulting classes with their attributes and their functions are given below

Class 1: attribute: A, D
functional dependency: F_1

Class 2: attributes A, B, C, E, F, G, H
functional dependency: F_2, F_3, F_4, F_5

V. DISCUSSION

The methodology eliminates the possible human intervention in the design of a class by automating the process with foundation of good database design principles and mathematical rigor. The methodology takes the number of attributes and functional dependencies as input to represent a dependency matrix. The two-dimensional array data structure, supported by many programming languages, satisfies the implementation process. The use of array data structure is quite efficient in the process of accessing the elements for a manipulation in accord with axiom and mathematical requirements. The time complexity of the designed algorithm is:

1. With number of functional dependencies n, the testing of each functional dependency for a transitive link with other functional dependencies takes the $n^* (n-1) \approx n^2$ comparisons.

2. For n number of functional dependencies, the testing of each functional dependency for recognition of exclusively identical determinant attributes and the subset association determinant attributes have the $n^* (n-1) \approx n^2$ number of comparisons.

Hence the time complexity of the algorithm is:

$$\boldsymbol{\Theta (n) = n^2 / 2 + n^2 /2 \approx n^2}$$

This methodology is developed based on the pseudo-transitive axiom, which is natural and sound. The compartment of attributes are done with amalgamation of "second normal form" [3] of good database design principle, which is sound , proven and minimizes the information content by categorizing the attribute set based on pseudo-transitive link, which is nothing but fulfillment of "Information axiom". The categorization of functional dependencies leads to function's signature, which are applied exclusively to a group of attributes, are abstracted. This is an independent maintenance of the class, which is fulfilling the "Independence axiom". Since the methodology is designed by the incorporation of axioms, hence it is sound and correct.

VI. CONCLUSION

In object oriented system design, the UML class diagram models the class. While depicting the class diagrams the attributes and function's signature of class is designated by the designer depending on his/her expertise and domain knowledge. This may lead towards the wrong design by inviting ambiguity in the design when more number of designers is involved in the decision.

The developed methodology utilizes the dependency matrix constituted by the functional dependencies and attributes. Then the matrix elements are manipulated in accord with pseudo-transitive link and subset link among key attributes of functional dependencies to categorize them. The scratch class is designed from the attribute(s) and function's signature from categorized functional dependencies. Since the methodology is developed by underpinning the axiom and mathematical rigor, hence it evidences the correctness. The methodology is illustrated with a case study. Further, the harnessing of other normal form design principles and the recognition of function's signature depending on the clustered functional dependencies are to be refined for a stronger foundation and design.

VII ACKNOWLEDGEMENT

We acknowledge the International Conference on Computing Business Applications and Legal Issues (ICCBALI 2011) held at Institute of Management and Technology Ghaziabad, India, where the paper is accepted for presentation and published.

VI. REFERENCES

[1] Ali Bahrami " Object Oriented Systems Development using UML", 2nd Edition, Tata McGraw-Hill New Delhi, 2008

[2] Abraham Silberschatz, Henery F. Korth, S. Sudarshan, "Database System Concepts", 5th Edition, McGraw-Hill International Edition, 2006

[3] Elmasri, Navathe, "Fundamentals of Database Systems", 5th Edition, Pearson Education, 2008

[4] C. J. Date, A. Kannan, S. Swaminathan, "An Introduction to Database Systems", 8th Edition, Pearson Education (Dorling Kindersley (India) Pvt. Ltd.), 2008.

[5] George C. Philip "Teaching Database Modeling and Design: Areas of Confusion and Helpful Hints" Journal of Information Technology Education Volume 6, page 481-497, 2007

[6] W. Kent, "A Simple Guide to Five Normal Forms in Relational Database Theory", Communications of the ACM, Vol. 26, No. 2, page 110-114, 1983.

[7] Patric O'Neil, Elizabeth O'Neil, "Database: principles, programming and performance", 2nd Edition, Morgan Kaufmann, page 369, 2001.

[8] Fangjie Xu, Huichuan Duan, "A CAI Tool for the Theory of Relation Normalization", 1-4244-1285-0/07 IEEE page 532-534, 2007

[9] Tauqeer Hussain, Shafay Shamail, Mian M. Awais, "Eliminating process of Normalization in Relational Database design" Proceedings IEEE INMIC, page 408-412, 2003.

[10] Amir Hassan Bahmani, Mahmoud Naghibzadeh, Behnam Bahmani "Automatic database normalization and primary key generation" IEEE CCECE/CCGEI May 5-7, Niagara Falls, Canada, 2008.

[11] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. ACM 12 (6), June 1970, page 377-387.

[12] E.F. Codd, "Normalized Data Base Structure: A Brief Tutorial", ACM SIGFIDET Workshop on Data Description, Access, and Control, San Diego, California, 1971

[13] Simon Bennett, Steve McRobb, Ray Farmer, "Object Oriented Systems Analysis and Design using UML", Second Edition, Tata Mc-Graw Hill Pvt. Ltd., New Delhi, page 456, 2009

[14] S M. Handigund, "Reverse Engineering of Legacy COBOL systems", Ph. D. thesis Indian Institute of Technology Bombay, 2001

[15] A. A. Chikkamanur, S. M. Handigund, "Categorization of Functional Dependencies for a Minimal Cover", ICSTC, San Diego, USA, page 213-217, 2008

[16] Thomos Connolly, Carolyn Begg, "Database Systems: A practical approach to design, implementation and management", Third Edition, Pearson Education, 2005.

[17] A. A. Chikkamannur, S. M. Handigund "An ameliorated methodology to design normalized relations" ACS / IEEE sponsored, 7th international Conference on Computer Systems and Applications (AICCSA09), Rabat, Morocco, 2009, page 861-864.

[18] A. A. Chikkamannur, S. M. Handigund "Design of Normalized Relation: An Ameliorated Tool" International Journal of Computing and Information Sciences (IJCIS), ISSN 1708-0460, in press.

[19] Mark Priestley " Practical Object-Oriented Design with UML", 2nd edition, Tata McGraw-Hill, 2008

[20] Simon Bennet, Steve McRobb, Ray Farmer " Object-Oriented Systems Analysis and Design Using UML", 2nd edition , Tata McGraw-Hill, page 365, 2009.

Ajeet A. Chikkamannur received his M. Tech. degree in Computer Science and Engineering in 2001 from the Visvesvaraya Technological University, India. Currently pursuing the Ph.D. and the research is focused on "Design of Fourth Generation Language". His research interests are Object Oriented System Development, Database Management Systems, System Simulation and Modeling. Presently working as Professor, Department of Computer Science and Engineering and teaching for graduate courses for last twenty-two years.

Prof. Shivanand M. Handigund received his Ph.D. degree from Department of Computer Science & Engineering, Indian Institute of Technology, Bombay in 2001. Currently, he is working as a full time Professor and Head, Super Computer, M. Tech. CSE Programme and Research Centre at Department of Computer Science and Engineering, Bangalore Institute of Technology, Bangalore. His research interests are Software Engineering, Reverse Engineering, Database Management Systems, Object Technology and Computer Graphics. He teaches several courses to Academia and Industry engineers. He has organized number of conferences and delivered keynote addresses & invited talks at several conferences. He is a Ph. D. referee and IEEE technical papers reviewer.