A Methodology for the Abstraction of Design Components from the Software Requirement Specification to the Object Oriented System

Syed Naimatullah Hussain and Nisar Hundewale
College of Computers and Information Technology
Taif University
Taif, Saudi Arabia
s.naimatullah@tu.edu.sa, nisar@computer.org

Sultan Aljahdali and Ashfaq Ahmed K
College of Computers and Information Technology
Taif University
Taif, Saudi Arabia
aljahdali@tu.edu.sa,ashfaqme@gmail.com

Abstract—**The software developer's task begins with the procurement of project charter. This is a legal document containing details regarding the software requirement specification (SRS), cost and the schedule etc., of the project. The SRS of the organization is a text document incorporating the requirements of the organization. The software development of any information system is based on the SRS of the client organization. This paper attempts to abstracts design components (Object class name, Object methods, and its attributes, Actors and interfaces of actors) from software requirement specification.**

The objective of this paper is to develop a single semi automated methodology for the abstraction of different useable components from SRS, so that they can be transformed as model elements. To provide a semiotic environment for the design of model elements to the transformation of useable components.

I. Introduction

UML has become a de facto standard [2]for the design and implementation of object oriented information system. Though the UML is a boom for the design of business processes in an object oriented paradigm, the success rate of projects involving the design using UML is very slow. Apart from other internal lacunae of UML itself, the main reasons for low success rate are due to the non availability of a correct and complete methodology for the design of information system. Though, few of the methodologies viz. Noun phrases approach, common class pattern approach, Use case driven approach and classes Responsibilities and collaborators [3][4] exists only for the abstraction of the object class names and few methods. But they seldom lessen the burden of arbitrary use of human skills. These methodologies contain conjecture based steps and also oxymoron concepts in different steps of same mythology, they can hardly be realized in practice. Thus, the existing methodologies at most serve as patterns or frame works per specific types of business processes. Most of the methods of the methodologies are manual without appropriate guide lines. This absence of concrete guidelines gives scope for arbitrary use of human skills. Thus the methodologies lose their relevance. In their proposed steps, the authors are not explicitly addressed the correctness and completeness of each step (input & output of the step).

978-1-4673-2008-5/12/$31.00 ©2012 IEEE

The SRS is the abstraction of number of end-user's requirements. Because of the environment in which the individual is working, the end user may use different words for the same meaning. Thus, the SRS contains synonymous words. In addition, the use of contexts specific words compels the end user to use the same word for different meanings. Thus, SRS also contains homonymous. The non-resolution of synonymous issue results in the redundancy model elements whereas; non-resolution homonym issue results in violation of uniqueness of the model elements. Thus any information system design needs to resolve the synonym and homonym issues. In general for other paradigms, the synonyms & homonyms issue is resolved by the different researchers. However, their resolution is limited to resolving the issue only at the system level and not at the application level. The end-user need to have a mechanism for mapping the system level synonymies to his corresponding synonymous words on separate name, specific to the application into an original homonym name. Alternatively, the end-user may be provided with list of synonymy issues as it may give scope for the intrusion of erroneous data. These shortfalls compel the design components abstraction for an information system to be human dependent and the success depends on individual skills. Thus, the success rate of the information system developed using such error prone approaches swings. There is an urgent need to isolate the dependency of software development success rate on human skills and to elevate it to the height of the automatic process.

This paper attempts the automatic abstraction of design components (design components: Object class name, Actors and interfaces of actors, and its attributes) this methodology that overcomes all the identified shortfalls like presence of synonymies & homonyms, the conjectures based steps, existence of oxymoron concepts between steps, absence of correctness & completeness authentication and limitation in the number of design components abstracted through existing methodologies. And groups the attributes of a class using the good data base design principles and strong cohesion & weak coupling concepts. Concurrently, the methodology also abstracts the actor interfaces & their attributes that are involved in the interaction with the information system.

The proposed methodology also abstracts the groups from set of identified attributes and also the subset of attributes for which there exist a method in the SRS. The attributes subset may contain parameters of the object class participating either as a referenced or as defined attributes in the methods and may optionally contain attributes of other object class or other actors interface as parameters referenced or defined in those methods.

The steps of this methodology will be with sound reasoning & mathematical rigors. Most of the steps may be automated and remaining few are semi automated in the sense that a lot of guidelines steps are proposed to narrow down the difference between the human skill & the actual process and their scope is limited to the authentication of correctness & completeness of each step.

II. LITERATURE REVIEW

We have studied available literature in the area: the existing methodology and available tools and techniques have the following shortfalls.

Rebecca Wirfs Brock et al [3][4]. have developed twelve steps methodologies to abstract object class name only. The process of identification is iterative process from the software requirements specification as input.

Step1: Read through the requirements specification for nouns/nouns phrases and identify them. Convert plurals into singulars and make a preliminary list.

Shortfall: In the above step attributes are discarded in subsequent steps and the intransitive verb has not been represented in the abstracted noun/noun phrases because it contains incomplete object class name.

Step2: categorize noun phrases into relevant, irrelevant and fuzzy classes.

Shortfall: This step just act as suggestion, as their does not exist any categorization

Step3: Discard irrelevant classes from noun phrases.

Shortfall: This elimination depends on individuals human skills

Step4: eliminate redundant noun/noun phrases

Shortfall: Identify synonyms and homonyms issues before elimination

Step5: considered the retained nouns/noun phrases as first cut object class names

Shortfall: This step does provide any guidelines to filter the attribute names from object class name

Step6: analyze noun/noun phrases containing adjectives means different object, different use of same object or different states of the object class

Shortfall: No guidelines have been provided for analyzing the noun/noun phrases, according our view noun phrases containing adjective may be an object class or state of the object class or attribute.

Step7: Review nouns/noun phrases for possible attributes, noun phrases presented only as are attributes

Shortfall: Non availability of guidelines, and also this step will not filter the attribute group names from the object class names

Step8: convert passive voice sentences into active voice

Shortfall: There is no sound reasoning behind this step. This step abstracts only the object class names.

Step9: For each identified noun/noun phrases there should bea statement of purpose.

Shortfall: There are no proper guidelines.

Step10: Model categories of classes i.e. super class, subclass hierarchy

Shortfall: This steps lacks proper guidelines

Step11: model interfaces to outside world i.e. interface to other program which helps in identifying complex object classes. Shortfall: The above step helps in identifying the interface but it is not clear how to identify it.

Step12: Identify missing classes from the irrelevant and fuzzy class

Shortfall: This itself show the incompleteness of the methodology, as in step3 the authors have suggested to discard the irrelevant classes

A. Common Class Pattern Approach (CCP) (developed by shlaer et. Al.) [3][4]

Shortfall: CCP considers the entity as the object class structures. We know that the object class, the structure, the entity, the states and methods are encapsulated together. But in the entity concept functionality may utilize part of an entity attributes or it may contain number of attributes across different entities. Thus it fails to distinguish between entity and the object class structure. Authors have not considered the good database design principles. This method does not provide any idea for assigning the attributes to different object classes.

B. Use case driven approach [3][4]

Shortfall: In this approach the authors have abstracted the scenarios of different activities from the SRS and then decomposed the scenarios based on actors to form use-cases. Then, for each of these use-cases, a sequence/collaboration diagram is designed to abstract object class names.
The authors in the above approach have stated that, the object classes are abstracted from the scenarios of the SRS. That indicates the object classes are guessed during sequence diagram design. This methodology may be use for some specific application

C. Deva Kumar et. al [5]

This paper uses a tool (UMGAR) which is nothing but automation of noun phrase approach and short falls of noun phrase approach is already addressed in the earlier part of literature survey.

D. G.S. Anandha et al

The authors in this paper have addressed the abstraction of all the useable components but failed to authenticate the correctness & completeness of this methodology.

E. M.G. Ilieva et al

This approach consists of three main processing parts: i) the Linguistic Component, in which the sentences in the text are analyzed; ii) the Semantic Network, built by the formal NL presentation; and iii) OO modeling, the final phase of the formal presentation of the specification, through which the knowledge and information included in the semantic network are transmitted to the OO analysis model's elements. This paper fails in Actors identification and also authentication of correctness and completeness.

F. Sukhamay Kundu et al

The authors are identifying the functions which should be factored into sub functions, including their desired signatures and a reduced use-complexity, in order to simplify the class subclass structure. This paper helps in the designing only the class and subclass structures but fails to identify the object class structure i.e object class name, attributes, actors and its interfaces.

G. H.J Klien et al

The authors discuss the relational concept of a functional dependency can be adopted to object databases in order to get more convenient ways of fetching objects. This paper helps in formation of functional dependencies for object database which is part of our object structure identification.

H. Sagar Pidaparthi et al

The authors have given more weight age to the procedural oriented paradigm hence they have not use the naturalness of Object – Oriented paradigm which gives the perfect balance between data and procedural oriented paradigm, and also which more towards naturalness

I. D. L. Carver et. Al

In their approach stated that system is decomposed from an object rather than from a functional view point. Among the characteristics exhibited by an object model are abstraction, encapsulation, hierarchy and typing. Abstraction is achieved by the conceptual boundary that makes object distinguishable, encapsulation is present by the hiding of unneeded details about the object modularity is achieved as a result of the high cohesive and low coupling exhibited by the objects. The authors have used only Good Software Engineering Design Principles for object class identification. We are going to use a perfect combination of Good Software Engineering design Principles as well as Good Database Design Principles.

J. Enrico Maim et.al

This paper explores the functionality of recognition in a constraint- based framework. In such a frame work since an attribute is only a special case of a constraint. The idea of recognition can be a generalized rather than just reasoning from a set of attributes to the type of an object recognition can reason from any set of constraints such as arithmetic constraints, inequalities and other relations to the type of an object. The authors in this paper are abstracting only the object class name and its attributes but not the Actors and its interfaces.

III. PROPOSED METHODOLOGY

We are proposing a fourteen steps methodology for the abstraction of design components that are required to design the system in an object oriented environment. The proposed methodology overcomes all the shortfalls like synonym and homonyms issue, authentication correctness and completeness. The proposed methodology identifies object class name from modified noun phrase approach and groups of attributes of object class using good design principles. Actors and its interfaces by use of context diagram and use of questionnaire. Further refinement of methodology is achieved through the use of completeness and correctness.

1. Identify the nouns-noun phrases (N), Adjectives (Ad), Transitive Verbs (Vt), Intransitive Verbs (Vi), Auxiliary Verbs (Vaux), Adverbs (Av). and Adverbial Phrases [3][4]

   Guideline**:**

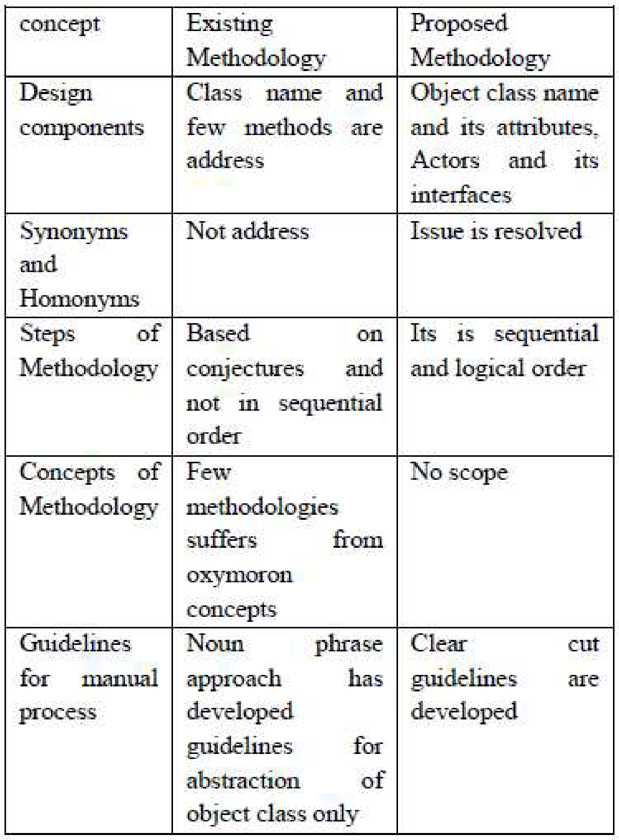
   - Manually Read through the SRS convert passive voice to active voice so as to transform auxiliary & impersonal verbs into corresponding intransitive or transitive verbs.
   - Assign a logical sequence for each SRS statement.
   - Assign each statement with appropriate digit in their logical order.
   - Design a Context flow graph of the SRS statement and represent it in the form of Table
   - Represent the context flow graph in the form of four columns table.
   - Represent the data flow graph with six columns table.

2. Resolve Synonym and Homonym issues in each category of words [13].
   - For each noun or ad –noun in the defined columns of table.
   - Search the entries for a set in table.
   - The homonym issue may be at the same or different levels i.e. a noun or ad-noun may be used both at attributes level.
   - The homonyms are identified by use of nouns/ad-nouns in different functionality. If both uses are at the same level, assign generic name.
   - Identify functional dependencies from the table

3. Use modified Wirfs-Brock approach, Abstract the structural requirements [3][4].
   - Create a class name directory from the table.

- The table contains group of nouns/ad-nouns, this are class name.
- The nouns/ad-nouns are studied for the presence of the single domain. Such items form the attributes. The remaining items are manually checked for their characterized as attributes. Store them in class attributes dictionary.
- The ad-nouns are categories as: attributes value and object/subclass name.

4. Identify the main functionalities of the system, manually from Transitive, Intransitive verbs and the associated object classes based on statement of purpose.
    - From the table identify transitive and intransitive verbs.
5. Identify the actors of the system through the use of questionnaire and delete their names from the noun/noun phrases set[15].
    - Design the context diagram from the SRS
    - Search the answer for the following questionnaires.
        - Who will use the main functionality of the system?
        - Who needs support from the system?
        - Who will maintain the system i.e identify the secondary actors?
        - Which hardware the system needs to handle?
        - With which other systems, the system needs interface?
        - Who/what has an interest on the product, service or the results, the system produces?
6. Identify the functionality of the system through the use of questionnaire and response from the use-case, attributes [1][15].
    - Intersection of word phrases from the context diagram and questionnaire are the list of actors.
7. Ensure the utilization of all transitive verbs and intransitive verbs should be included in the set of nouns/noun-phrases
8. For each functionality, identify the attributes set and first cut object class name from the noun/noun-phrases. Remaining nouns/noun phrases. Remaining nouns/noun phrases may be studied for their being possible synonyms
    - Sieve Actor name, related attribute, class name, class attribute, subclass and generic from each of the noun and ad-nouns from the table
9. Design logical dataflow diagram (DFD) [14].
    - A DFD is a graphical representation of flow of data through an information system.
    - A logical DFD it depicts the graphical representation of overview of SRS it is used here for better communication with users, more stable system and better understanding of the entire business of an information system.
10. Perform two levels of the system data integration with reference to data and functionalities.
    - System data integration involves combining data residing in different sources and providing users with a unified view of these data.
        - Level 1: A data flow may contain part of an entity
        - Level 2: A data flow attributes of the entity may be distributes in the number of data flow.
11. Normalize the data flows attributes groups at least to Boyce Codd Normal form [14].
    - Normalize the data using axioms to achieve good database design principles.
    - By normalizing
        - Redundancy is minimized.
        - Unrelated attributes will be separated.
        - Functional dependencies amongst attributes group should be preserved.
        - Inconsistencies will be eliminated.
        - Data integrity will be ensured.
        - Attributes will null value is minimized.
        - Easy scope of maintenance.
        - Each attribute will be one or the other group.
12. Ensure each normalized attributes set participates as a dataflow in the logical DFD. If an attributes or group of attributes set has one or more attributes common, then apply the appropriate axioms on Functional Dependencies to takeout the common attributes to form
a separate object.
13. Design the physical data flow diagram.
    - Describing processes in more detail than logical DFD. It clarifies which processes are manual and which are automated
    - It is used to visualized data processing(structured design)
    - A DFD shows what kind of data will be input to and output from the system, where data is stored.
14. Identify the first cut object structures and first cut use-cases respectively from the input data flows, and the processes to which they have incident in the physical DFD.



concept	Existing Methodology	Proposed Methodology
Design components	Class name and few methods are address	Object class name and its attributes, Actors and its interfaces
Synonyms and Homonyms	Not address	Issue is resolved
Steps of Methodology	Based on conjectures and not in sequential order	Its is sequential and logical order
Concepts of Methodology	Few methodologies suffers from oxymoron concepts	No scope
Guidelines for manual process	Noun phrase approach has developed guidelines for abstraction of object class only	Clear cut guidelines are developed

TABLE I. COMPARISION

IV. CONCLUSION

The proposed fourteen stage methodology attempt to abstract different useable components from the SRS. There exists number of approaches for this abstraction. The Noun Phrase approach abstract class name only which is based on conjectures and oxymoron concepts. This step in noun phrase approach will not abstract completely the object class names. Some of the object class name is implicit with the presence of intransitive verb of the same name and attributes. The abstracted nouns/noun phrases may contain incomplete object class. The common class approach uses the definition of entity. Entity cannot be represented as object structures. The correctness and completeness of the methodology is addressed. In use-case driven approach the author have abstracted the scenarios of the different activities from the SRS and then decomposed the scenarios based on actors to form use-cases. In this the author have not addressed any concrete guidelines to abstract scenarios, nor attempted to authenticate the correctness and completeness. No where the authors have stated that the object classes are abstracted from the scenarios. In this paper we have proposed a fourteen stage methodology for successful abstraction of different useable components from the SRS.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the contributions of Dr. Shivanand M Handigund, Professor and HOD CSE Department, BIT, Bangalore, for his constant support and encouragement for successful completion of this research paper.

REFERENCES

[1] Pankaj Jalote (2005). An Integrated approach to Software engineering, 3rd edition springers publications, India.

[2] Qasim Siddique (August 2010). Unified Modeling Language to Object Oriented software development International Journal of Innovation, management and Technology, Vol. 1, No. 3, ISSN: 2010-0248.

[3] Ali Bahrami (2008) object oriented system development using unified modeling Language, Tata McGraw-Hill Edition.

[4] Rebecca Wirfs (1990). Designing Object Oriented Software, Prentice-Hall, India.

[5] Deva Kumar Deeptimahanti, Muhammad Ali Babar (2009) 24th IEEE/ACM International Conference on Automated Software Engineering, ASE '09.

[6] G.S. Anandha Mala and Dr.G.V.Uma "Object Oriented Visualization of Natural Language Requirement Specification and NFR Preference Elicitation. IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.8, August 2006.

[7]. M.G. Ilieva and Olga Ormandjieva A. Montoyo et al. (Eds.): NLDB 2005, LNCS 3513, pp. 392–397, 2005 Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation. © Springer-Verlag Berlin Heidelberg 2005.

[8] Sukhamay Kundu & Migel (IEEE-2005) 29th International Conference Software and Application Conference "A Formal approach to Designing a class subclass structure using a partial order on the functions" page 1 -8.

[9] H.J Klien and J.Rasch (IEEE-1997) in their case study "Value based identification and functional dependencies for object databases" page 22 - 32.

[10] A case study in Migration to Object – Oriented system structure using Design Transformation methods by Sagar Pidaparthi and Grzegorz Cysewski (1997) IEEE page 128 – 135.

[11] D. L. Carver (IEEE-1992) in his case study "Promoting the use of an object-oriented software development methodology by merging structured and object oriented analysis methods" page 593 – 599.

[12] Enrico Maim (IEEE-1992) in his case study "Recognizing objects from constraints" page 47 – 54.

[13] Shivanand M. Handigund (2001)."Reverse Engineering of Legacy COBOL Systems", Ph.D. Thesis, 2001, IIT Bambay, Mumbai.

[14] S.B. Navathe et al. (1986)."A comparative analysis of methodologies for database schema integration." ACM Computing Surveys, 18(4), 323-364.

[15] William R. Duncan, A guide to project Management Body of

[16] Knowledge, PMI Standard Committee, Newtown Square, USA, 2004, Page 50-51.