A Methodology for the Abstraction of Software Component from Software Requirement Specification (SRS)

Muzammil H Mohammed
Assistant Professor, Department of Information Technology,
College of Computers and Informathion Technology,
Taif University,
Taif, Saudi Arabia

Syed Naimatullah Hussain
Lecturer,Department of Computer Science,
College of Computers and Informathion Technology
Taif University
Taif,Saudi Arabia

Abstract:**This paper attempts to abstract software components i.e. (object class name and its attributes, Actors and its interface) from the software requirement specification (SRS). These abstractions are further refined using the blend of good database design principles. A sequence of semi methodology is developed to carry out these abstractions. This is a fool proof semi-automated methodology which abstracts the required paradigm from the SRS.**

Keywords: SRS software requirement specificaion, ctu: clients team of users, dfd dataflow diagram

1. INTRODUCTION

The software development project normally starts with customers' requirements. The customers are in general, strategic management people of the organization who work in classical ambience, so the requirements of the expected system reflect their processing mindset. These requirements are influenced by either the data oriented approach or the procedure oriented approach as with the available information of the organization. Presently, since these are not natural ways of processing the information system, these will not serve the development process effectively. Now a day, people feel the object-oriented paradigm is more towards naturalness and will survive for long time. So it is required to transform the requirements into object-oriented paradigm and then proceed for the development. We are intending in our ensued methodology, to develop a system of software tools, which takes the requirements (originally was either procedure oriented or data oriented paradigms) and then transform it into object-oriented paradigm. We are intending to develop an automated sequence of software tools that takes requirements definition as input and produces the developed specifications in object-oriented paradigm. There may also need to limit our ambition, as some of the sub processes may not be automated. In such case, there is a need to provide guidelines for each of these sub processes to minimize the human dependency. We are aiming to develop a sequence of automated software tools with embedded guidelines for inevitable subtasks at some stages, and then the set of guidelines may give the scope to develop software agents to take up the role of semi automated processes.

Few researchers [4, 5] have suggested some techniques for certain stages of the design of object classes. Although, these give good guidelines to the design, the authors could not derive any concrete procedure and/or guidelines to the design in its totality. We have made an attempt develop a methodology that identifies the object-oriented specifications in the form of object structures, object methods and the interrelationships, from the requirements of an information system. This semi automatic methodology comprises of a sequence of steps like feasibility analysis, for object structure identification, resolution of synonyms & homonyms issues, regrouping of attributes of entities & functionalities through the design of data flow diagrams and elimination of imbalance between data & procedure oriented paradigm along with authentication of correctness & completeness of the abstractions at each stage. Manual intervention at few stages is necessitated because of the need for human intelligence in these steps. Even for these manual intervention steps, attempt is made to provide clear-cut guidelines to streamline the design process. This methodology is in perfect tune with the very basic definition of object-oriented paradigm. The paradigm brings the perfect balance between the procedure-oriented and the data-oriented paradigms.

II. THE METHODOLOGY

The algorithm presumes that, based on the customer's requirements, the software requirements specification (SRS) is already available with the developer. He/She can seek needy information from client's team of users (CTU). This paper addresses a sequence of semiautomatic methodology. Each step is discussed with details of either procedure, if it can be automated, or guidelines, if it requires human intervention

A. *Requirement gathering.*

As per the SRS, the detailed requirement is gathered from CTU. This depends on the managerial skill of developer. The developer may interview each member of CTU for his or her work process details. The input, output and how the input is transformed into the expected output, the actor who consumes the result or who supplies input information, the purpose of the work process/es etc. are collected.

- Each individual work process forms one or more functionalities.

- The response to ‘how’ frames the business rules, which identifies the functional dependencies amongst attributes and interrelationships between the entities/actors.
- The designed entities/actors, their characterizing attribute the functional dependencies amongst the attributes of each entity /actor and the interrelationships between the entities are captured in the form of data dictionary.
- These abstractions may partially be of data oriented and partially be of procedure-oriented paradigms. This depends on the nature of existing work processes (if exist) or the response of each CTU member

The requirement of the information system contains the business rule of the information system along with the branches and various applications. For example, the requirements of the college information system may contain some of the business rules as follows.

B. *Authentication of correctness and completeness of the process*

Now we have two sets of entities, attributes, interrelationships, business rules, work processes and business process. The developer need to establish the one-to-one and on-to correspondence separately between each pair of items of two sets

C. *Resolve synonyms/homonyms issue.*

In a multi-user system [2], though each user assigns meaningful names to attributes, the semantic flexibility in the use of English words leads to presence of synonymous words for the same attribute. The set of synonymous words of the same meaning forms a synonymy and each such synonymy is replaced by a generic name. Similarly, the use of context specific word leads to the use of same attribute/entity for different meaning in different entities. Each such word is a homonym.. The presence of each such homonym in attributes/entities/actors should be replaced by different names. The establishment of one-one and on-to functionality between the entities and attributes identifies some of the synonymies and homonymous words. handigund et al. [4] have developed semiautomatic techniques to resolve synonyms and homonyms issues. These can be used to resolve their presence. Appropriate modify the data dictionary.

D. *Eliminate redundancy in attributes/ entities presence*

Study each attributes of each entity/actor in isolation with other entities/attributes for absolute necessity of their presence in the information system. This can be identified by the participation of the attribute in any of the functionalities. Discard the attributes that are not participating in the functionalities. If an attribute or group of attributes is present in two or more entities, form a separate entity with each such group. This participation can be tested through the establishment of one-to-one and on-to correspondence between attributes referenced in data flows of logical DFD and data dictionary.

E. *Identify keys and design extended ER-diagram*

Abstract the functional dependencies amongst attributes of each entity from the business rules of the information system. Identify the primary key and foreign keys for each entity/actor. If an attribute or group of attributes of an entity of data store is independent of the primary key, take it out and form separate entity. Design the extended ER diagram with the following component abstractions. The entities and attributes are abstracted from the data dictionary and the interrelationships are abstracted from the business rules of the system.

F. Minimize the imbalance between the procedure and data oriented paradigms.

Regroup the attributes of input data flows, based on their characterization of Person [1], place, thing, event or concept. This can be achieved through the grouping the attributes such that each attribute of a group establishes a functional dependence with one primary key. The input data flows may contain subset of attributes of a group so that each group of attributes may be in input data flows of one or more processes. Each such group forms a first cut object structure

G. *Refine the abstracts by brining the perfect balance between the paradigms.*

Apply good database design principles to each first cut object structure. The good database designed principles are:

- Redundancy should be minimized
- Unrelated attributes should be separated
- The functional dependencies amongst attributes group should be preserved
- Inconsistencies should be eliminated
- Data integrity should be ensured
- Attributes with null values should be minimized
- There should be easy scope for maintenance
- Each attribute should be in one or the other group
- Constraints should be incorporated in the design.

Since the normalization is a way of good database design, the refined group should be normalized at least up to Boyce coded normal form. Indicates the constructs whose balance is to be maintained to bring perfect balance between procedure oriented and data oriented paradigms.

H. *Model the business process through data flow diagram (DFD) to measure the paradigms imbalance.*

Design the logical DFD with the following

- Entities, which are modifiable within the system, form data stores.
- Entities which are non-modifiable within the systems form actors.
- The identified functionalities form processes

The intersection of input/output attributes of functionality with each entity attributes forms input/output dataflow from/to the concerned entity /actor to/from the process (of functionality).

I.Design the context and logical DFD with object structure

Refine the logical data flow diagram with each object structure group as data store, maintaining attributes of total input data flow to each process and redesigning input flows such that each flow emanates from first cut object structure.

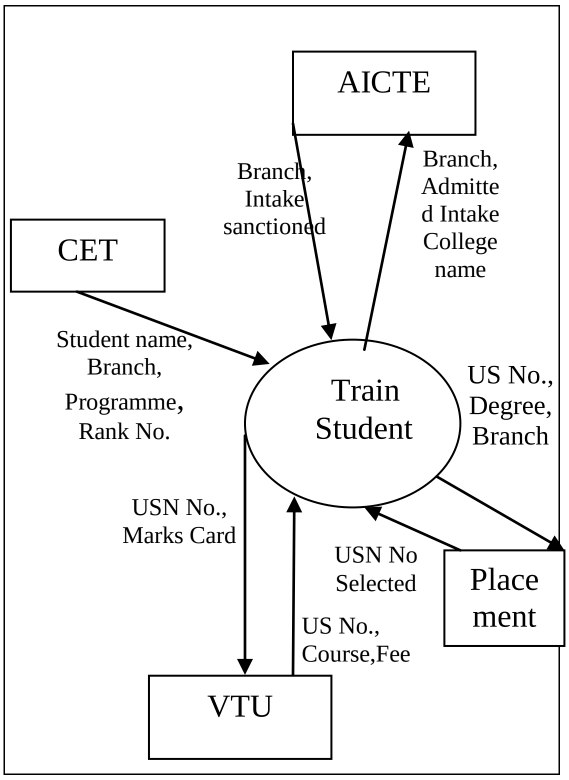

Fig .1 Context Diagram

In the above context diagram, the attributes common Entrance Test(CET), All India Council for Technical Education(AICTE), vishveshraya Technological university(VTU), PLACEMENT are depicted as the actors and TRAIN STUDENT is depicted as the lone process. The data stores, data flows and the sub processes are within this process. Here, a student is admitted to college when he/she qualifies for the CET exam.

J. *Decompose functionality*

If two or more data flows directed towards single
Process; study the possibility of decomposing the process, so that each decomposed process either receives data flow from single entity or additionally through parameter passing from other entities. Refine the logical data flow diagram into physical data flow diagram.

K.Design the class diagram

The actors, data stores form object classes, with the set union of attributes flowing from each of these, for their attributes, the processes that receives data flow from each of these data stores /actors form the object methods and the source of parameter passed to any of these functionalities form the association. The interrelationships identified in the extended ER diagram are to be modified

L. Identification of functional and multivalued dependencies

These entities are now refined with elimination of redundant attributes and entities [3]. These can serve as first cut object structures. Now the functional dependencies and the multi-valued dependencies that may exist amongst the attributes of each entity are to be identified. The undesirable functional dependencies are to be eliminated using normalization process in sequence from the first normal form to the Boyce-Codd normal form (BCNF). The undesirable multi-valued dependencies are identified through the one-to-many relationships between different attributes of each entity. These are eliminated by decomposing each such entity using fourth normal form and project join normal form (PJNF).

Now, we have identified first cut object structures using good database design principles on one hand. On the other hand, we have identified attributes for each dataflow through the design of higher-level data flow diagram. The object-oriented paradigm is the perfect balance [6] of these two paradigms. Thus, the design of object-oriented specifications need to blend the data oriented (object structures) paradigm with procedure oriented (Attributes group each representing a dataflow) paradigm. There needs to be a one-one and onto correspondences [3, 8] between the two sets of structures identified. This also implicitly verifies and validates the selection of object structures.

Now, we study the mapping between two groups, one group comprising attributes groups of data flows, each group representing a dataflow and on the other side, the refined object structures. We identify one-one onto correspondences between these two sets of elements. If an object structure contains one or more dataflow groups then, the corresponding functionalities are assigned to the contained object structure as objects methods. This process continues for all the matching object structures. Now, we take the set union of unmatched object structures and study the possible consideration of one-to-one mapping with left out dataflow groups. Each such matching data flow group forms an object structure with its destined process as object method. The left out data flow groups are manually studied for possible participation. Similarly, the left out object structures are studied for possible formation of abstract classes.

CONCLUSION

The authors have identified the lacunae that present in various methodologies which uses manual process to design object classes. An attempt has been made here to automate the developed process. The Authors have succeeded in making the process semiautomatic, with least human intervention. The intervention is necessitated at critical points where intelligence is necessary

REFERENCES

[1] Muhammed Usman, Stepahane Ducane and Marianne Huchard (IEEE-2008) 15th working conference on reverse engineering “Reconsidering classes in procedural object oriented code” page 257- 266.

[2] Jonathan & charles J Hannon (IEEE-2009) Eighth Mexican International Conference on Current Trends in Computer Science. “An algorithm for identifyingAuthors using synonyms” page 99 – 104.

[3] Alen Lovrencie and Tonirnir Kisasondi (IEEE-2007) 11th International Conference on Intelligent Engineering System “Modelling Functional Dependencies in Databases using mathematical logic” page 307 – 312.

[4] Shivanand M. Handigund, “Reverse Engineering of Legacy COBOL Systems”, Ph. .D. Thesis, 2001, I. IIT. Bambay, Mumbai

[5] Ali Bahrami “Object Oriented System Developments” McGraw-Hill International Editions 1999.

[6] Sinan Si Albir. “UML in nutshell” , Shroff publishers and distributors private Ltd. 1998

[7] Jaco de Bakker and Erik de Vink. Control Flow Semantics. The MIT Press, Cambridge, Masschusetts, 1996.

[8] D B Phatak. Migrating Legacy systems. Pre-coneference tutorial Presented at COMAD95, Pune (India), 1999