Domain Identification and Description Techniques

Naeem Akhter
Assistant Computer Programmer
University of Sargodha

Muhammad Idrees
Lecturer, University of Lahore
Sargodha Campus

Furqan-ur-Rehman
Assistant Network Administrator
UCA, University of Sargodha

ABSTRACT
A system to be developed is expected to provide solution of a problem. After a careful identification of domains, an explicit, precise and well-engineered description is essential to develop an accurate solution providing system. Machine domain is formal while problem domain is informal. Description is like a platform for the developers to start and finish the development activities. Four principles regarding domain descriptions include von Neuman's principle; Principle of reductionism; Montaigne's principle and Shanley law. Graphs and models are also vital tools of domain description.

Keywords
Problem domain, Machine domain, Reductionism, Analytical model, Iconic model, Analogic model

1. INTRODUCTION
A computer is a general-purpose machine which has been built physically by hardware engineers. Building a Computer system and or software is meant to provide solution to a specific problem or purpose-in-hand. Software Engineering as a whole deals with the systematic approach towards building such solutions. The built software is actually a description of the specific machine to solve the in-hand problem, and computer selected for this purpose can be called a machine. On the other hand, there is a world of human beings which will interact with this software or system. This problem world is, in most cases, completely informal. An explicit, precise, formal and well-engineered description of both machine and problem world is direly needed for achievement of optimal end results from the system to be built. Unfortunately, description is somehow a neglected discipline in software engineering. The paper discusses various description techniques in detail. The remaining parts of the paper are: 2.Formal vs Informal 3. Identifying Domains 4. Describing Domains 5. Description Structures 6. 6. Description and Models 7. Conclusion 8. Future Work 9. References.

2. FORMAL VS INFORMAL
The machine is always formal as it is unable to produce any meaningful result if accurate input or instructions are not provided in purely formal way. The problem world, on the other hand, is largely informal because it includes human beings in most of the cases. Human beings can predict, analyze, estimate, judge or take decisions using their own intelligence hence they can afford to be informal in their lives. But to achieve a desired behavior or outcome from a computer system informal problem world is required to be converted into formal one. It is extremely tedious task. In some cases, it is even more difficult than developing the whole software because informality in itself has vast traits depending upon cultural and traditional aspects of the problem world. The job is well done, if the informal problem world has been described formal enough to justify the solution of the problem in hand. [1]

3. IDENTIFYING DOMAINS
Primarily, description refers the exact and precise identification of domains to be described. Broadly, in software engineering, there are two major domains, *problem domain* and *machine domain*. The machine domain consists of the world which will be responsible to provide the solution. The problem domain refers the world which expects the desired solution of the problem from the machine. For example, in a students' management system of a university, machine domain may be a computer having students' database, printer, wires, UPS, generator, storage and backup devices etc. The problem domain may be the students, their parents, staff working on computers etc. In a library management system, the problem domain may contain, periodicals, books, scholarly thesis or papers, students or other members of the library, working personnel, book-shelves, chairs, tables, membership ID-cards etc.

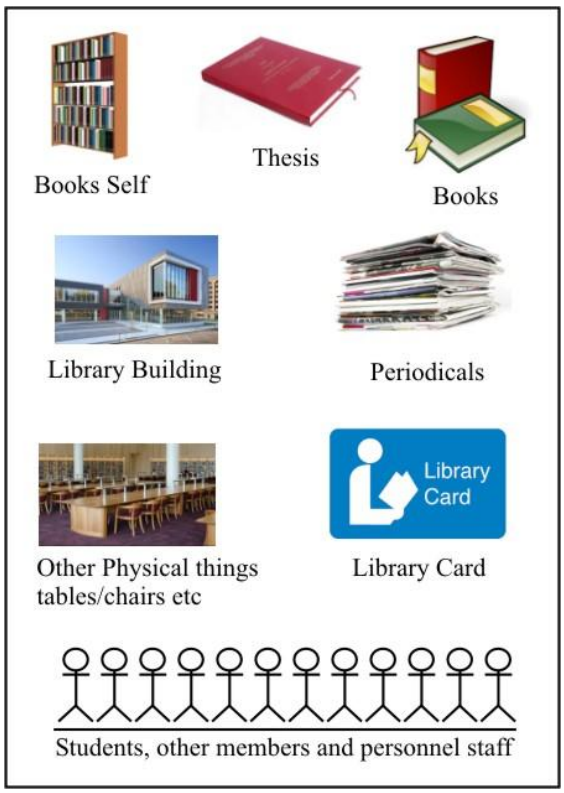

Fig. 2.1 Library Management System - Problem Domain

The machine domain may be computer, software, storage and backup devices, database, cables, printer, UPS etc.

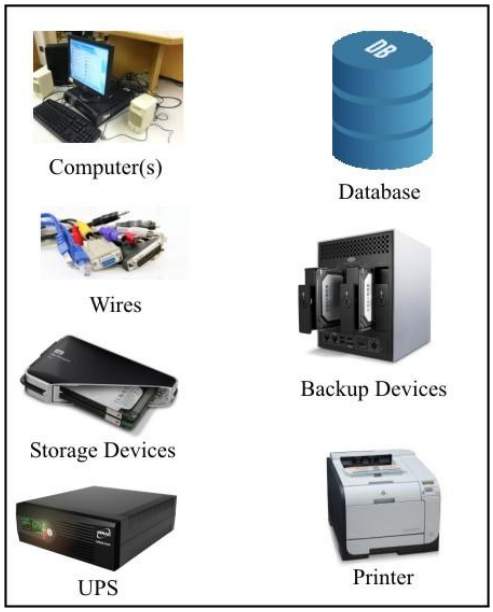

Fig. 2.2 Library Management System - Machine Domain

4. DESCRIBING DOMAINS

Describing the machine domain is relatively easy because it is completely formal. Describing the informal problem domain is very complex. For clear, precise, careful and well-engineered description of problem domain, following four principles are of key importance:

- von Neuman's principle;
- Principle of reductionism;
- Montaigne's principle
- Shanley law

4.1 John von Neuman's principle

While discussing the "Theory of Games" Neuman [2] states:

"There is no clarity in the concepts and issues to which they are to be applied."

For description of informal reality, first step is to clearly identify and recognize the ground terms; second step is to develop a generic vocabulary of these terms. Each and every recognized phenomenon will be assigned a *formal term*. These terms must be helpful in description of both machine and problem worlds because these are representative of specific phenomena of both worlds which are of key interest. For recognition of these *formal terms* a *recognition rule* is taken into practice which provides the phenomenon of informal world. John von Newman's Principle says that combination of *recognition rule* and assigned *formal term* provides us *designation*. For example:

member(M, L) ⇄ M is the member of L

In the above example, *formal term* is "member(M, L)" and the *recognition rule* is "M is the member of L (library)". If the terminology of the domain is poor, von Neuman's Principle can be used to recognize the phenomena of interest and designate it accordingly.

4.2 The Principle of Reductionism

Reductionism is an approach in which complex things can be decomposed into parts or simpler things. von Neuman discussed about the recognition of phenomena of interest and designate it through the use of recognition rule and formal terms. The Principle of Reductionism expresses the ways through which a recognized phenomenon can further be decomposed into simpler units. The purpose behind this principle is to assign exact, precise and clear ground terms to the decomposed units of phenomenon. For example, in a library management system an eye-catching ground term may a '*member*'. But after application of Reductionism the decomposed units of phenomenon may have more clear, accurate and precise ground terms as "*enrolled*" in the library, "*resigned*" from the library, "*lapsed*" or "*expelled*" from the library.

4.3 Montaigne's Principle

Montaigne's Principle discusses the need to carefully identify the *suggestive* and *desirable*. Montaigne was a French essayist of 16th century who presented his views about *indicative mood* and *optative mood*. His thoughts are very much relevant to description of problem world.

Indicative mood expresses underlying facts of the phenomenon and explicitly unfolds the reality. Optative mood, on the other hand, expresses a trivial wish or hope. It does not express anything meaningful about the problem world instead tries to express the world in different perspective. For example:

- Worker xyz works 8 hours daily. (Indicative mood)
- Wish you work 24 hours daily. (Optative mood)

Wish or hope may be trivial or positive, but its clear identification is essential because it leads towards ambiguity in domain description. Wishes and hopes must be eliminated as a whole and only factuality must be taken into consideration while precisely describing the domains.

4.4 Shanley's law

Knuth [3] quotes Arnoul [4] while discussing the "go to statements":

"…..If you make a cross-section of, for instance, the German V-2 [rockets], you find external skin, structural rods, tank wall, etc. If you cut across the Saturn-B moon rocket, you find only an external skin which is at the same time a structural component and the tank wall. Rocketry engineers have used the 'Shanley Principle' thoroughly when they use the fuel pressure inside the tank to improve the rigidity of the external skin!"

Shanley's law suggests design efficiency of the world. Profound architecture of the world is of much concern. The world is full of elementary individuals and everyone plays multiple roles in performance of various functions. Distinct domains can be found in individuals same as one node in graphical representation may be a part of more than one sub graphs. While describing these multiple structures and many-sidedness each and every complexity of the phenomenon must be understood. A book [5] was written on object-orientation which ends with the following golden words from Christopher Alexander [6]:

"It is possible to make buildings by stringing together patterns, in a rather loose way. A building made like this is an assembly patterns. It is not dense. It is not profound. But it is also possible to put patterns together in such a way that many patterns overlap in the same physical space: the building is very dense; it has same physical space: the building is very

dense; it has many meanings captured in a small space; and through this density it becomes profound."

The patterns in the world are required to be recognized and provision of precise recognition accordingly to support strong design of description.

5. DESCRIPTION STRUCTURES

The general description of a problem was presented by Jackson [7].

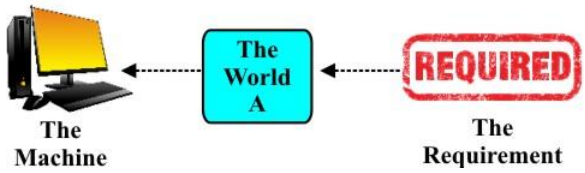

Fig. 5.1: Description of a general problem.

But in most of the cases, the problems are very complex. As a whole, the formal description of these problems is extremely difficult and confusing. Such problems can be decomposed into sub-domains. See figure below:

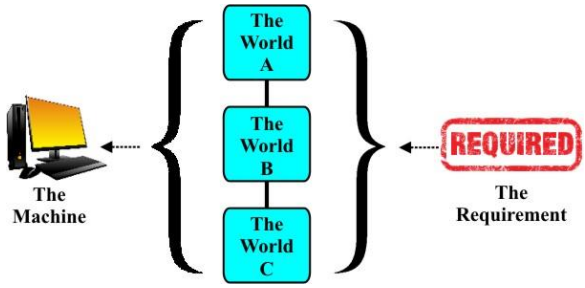

Fig. 5.2: Description of decomposed world (3-domain).

Decomposing the problem world into sub-problems or domains is not enough. The requirement is also decomposed accordingly because one part of a requirement may relate to Domain-A, and the other may be linked to Domain-B. See figure below:

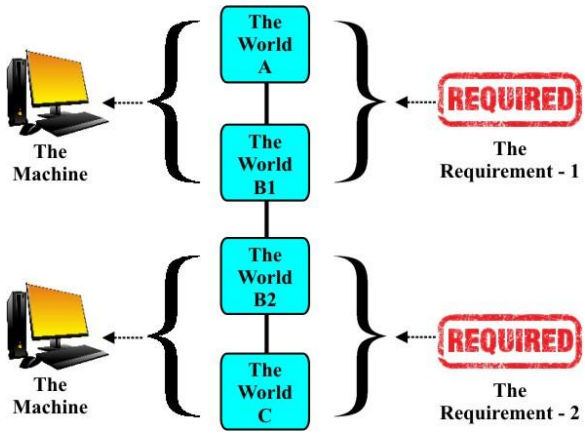

Fig. 5.3: Description of decomposed requirement

The requirement has been decomposed and corresponds to its relative domain. The world-B1 and B2 have different views which are appropriate to each sub-problem [8].

6. DESCRIPTION AND MODELS

Describing domains with models is vital to explore underlying patterns and relationships among entities of the problem domain. The models are helpful in unveiling the relationships among these entities. There are multiple outcomes of using models for domain description. Domain models enable stakeholders especially the non-technical stakeholders to develop a deep understanding of the system to be built. The designers are envisioned to control the complexity of the system. They become proactive in terms of system maintainability, reliability, testability and future enhancement.

According to Ackoff [10] there are three types of models of a domain. These are analytical, Iconic and Analogic models.

Analytical model may be a description of the vending machine behavior shown through labeled transition diagram. This model refers to the formal description which leads towards further formulation of properties.

Fig. 6.1: Description through Analytical Model.

Iconic model represents the physical structure of a domain. In Iconic Model, domain objects are modeled in their exact shape or extremely similar in all respects. A drawing or sketch of a building, cars, airplanes etc may be an example of iconic model.

Fig. 6.2: Description through Iconic Model

Analogic model represents the phenomenon of the problem world in a uniquely analyzable and understandable way. It is a representation of information in a domain in terms of a map of a country or traffic or railway schedules etc. Analogic models are also called "target systems" or dynamical analogies.

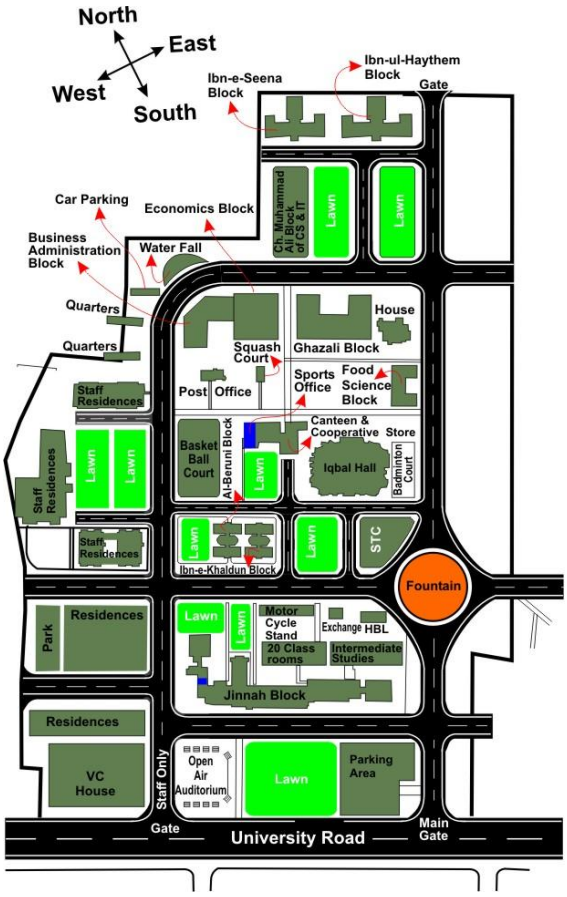

Fig. 6.3: Description through Analogic Model

A particular domain can be modeled in different ways. Model describing a scenario for machine domain will be different from the model describing the same corresponding scenario of problem world.

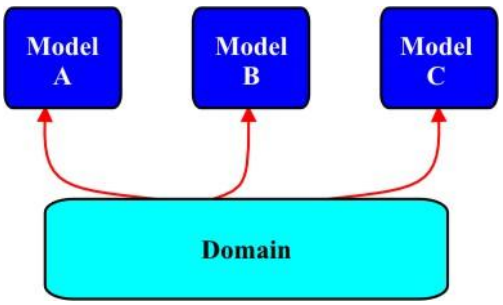

Fig. 6.4: Domain description through models.

Keeping in mind, these three types of models, domains can be described by choosing the model which suits a particular scenario.

7. CONCLUSION

Broadly, there are two domains, machine domain and problem domain of a system to be built. Identification of domain properties is the key to initiate description in black & white. Informal world must be described carefully for formal machine. For this purpose, four principles i.e. von Neuman's principle; Principle of reductionism; Montaigne's principle and Shanley law can be taken into practice. A complex problem containing sub-domains needs to be decomposed accordingly. Explicit description of decomposed units must be carried out carefully. Graphs can also be used as vital tool for description because a picture is worth thousand words. Models have close link with description. But a model representing a machine scenario will be different from the model representing the same scenario of problem domain. Description seems to be a neglected discipline in software engineering. In order to develop an accurate, efficient, reliable and dependable solution-providing system, it is the duty of the domain expert to bring into practice a suitable set of description techniques so that there may remain no ambiguous or unexplained part of both problem and machine domains.

8. FUTURE WORK

Informality of the problem domain is under the shadow of traditional, cultural, religious and regional traits. The ground terms used in different parts of the globe are varying in their literal meanings. An effort can be made to develop a generic set of ground terms, designations, definitions which may represent the problem world all over the globe.

9. REFERENCES

[1] Michael Jackson; Requirements and Specifications: a Lexicon of Practice, Principles and Prejudices; Addison-Wesley, 1995

[2] John von Neumann and Oskar Morgenstern; Theory of Games and Economic Behaviour; Princeton University Press, 1944.

[3] Donald E Knuth; Structured Programming with **go to** Statements; ACM Computing Surveys Volume 6 Number 4 pages 261-301, December 1974.

[4] Pierre-Arnoul de Marneffe; Holon programming:A survey; Université de Liège, Service Informatique, 1973. Quoted in [Knuth74].

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley 1994.

[6] Christopher Alexander; The Timeless Way of Building; Oxford University Press, 1979.

[7] Michael Jackson; Software Requirements & Specifications: a lexicon of practice, principles and prejudices; Addison-Wesley and ACM Press 1995.

[8] Daniel Jackson and Michael Jackson; Problem Decomposition for Reuse; Software Engineering Journal 11,1 pages 19-30, January 1996.

[9] Jim Woodcock and Martin Loomes; Software Engineering Mathematics: Formal Methods Demystified; Pitman, 1988.

[10] R L Ackoff, Scientific Method: Optimizing Applied Research Decisions, Chichester, England, Wiley, 1962.