*Journal of Software Engineering and Applications*, 2011, 4, 345-349
doi:10.4236/jsea.2011.46039 Published Online June 2011 (http://www.SciRP.org/journal/jsea)

# Improvement of Software Quality Attributes in Object Oriented Analysis and Design Phase Using Goal-Question-Metric Paradigm

**Hitesh Rajput[1], Lalit Kumar Singh[2]**

[1]Product Engineering Services, Patni Computers, Navi Mumbai, India; [2]Department of Atomic Energy, Nuclear Power Corporation of India Ltd, Bhabha Atomic Research Centre, Mumbai, India.
Email: rajputhitesh0@gmail.com, lalit.rs.cse@itbhu.ac.in, lksingh@npcil.co.in

Received April 19th, 2011, revised May 16th, 2011; accepted May 24th, 2011.

## ABSTRACT

*In a competitive business landscape, large organizations such as insurance companies and banks are under high pressure to innovate, improvise and distinguish their products and services while continuing to reduce the time-to market for new product introductions. Traditional approaches to software reliability modeling for such software are black box-based. Bad structure or model again can lead us to lower down these non functional properties. The basic constructs of the model are objects. We will not deal about the identification of the objects, as may be referred in many books, but how to model those objects. The objective of this paper is to provide an philosophical approach, using Goal-Question-Metric paradigm, to structure or model the identified objects of software system, in better way to improve the quality of the software.*

***Keywords*: *Software Reliability, OOAD, Class, Object, Association, GQM, MFC***

## 1. Introduction

Many companies invest significant resources and effort in software development. Because of today's high market demands for software; its size, complexity, quality needs are increasing rapidly. Due to which the various problems, specific for software development, viz planning difficulties, unknown or bad quality, projects are never ended, milestones that are reached months or years too late, or developers who are working mostly unstructured, under high stress; are also increasing. Those problems are generally known as "software crisis" and are being tackled by 'Software Process Improvement' (SPI) [1]. The object-oriented paradigm gives us tools for fighting the software crisis: Object-oriented techniques make it possible for us to handle large systems, change them, reuse parts of old systems in new systems, ease the communication between customer and developer, and much more. When we are fighting the software crisis we are fighting human nature and human inability to handle complexity. The performance/price ratio has exploded for hardware, whereas the increase in the performance/ price ratio is hardly noticeable for software.

Worse than this quantity problem is the quality problem. The quality problem is due to lack of SPI. One of the ways to handle is problem is through object-oriented modeling. The act of making model is known as modeling. A model is a representation of a real-world process, device or concept [IEEE-729]. When we start building a system, we might find tangible and non tangible items (ideas, etc). All these must be put in our model. So, model is the collection of real objects that exist in the world by applying our ideas. This can be understood with the help of **Figure 1**. Basic concepts used in object-oriented modeling are objects, classes, attributes and messages and important concepts such as information hiding. We thus need a technique that can be used to arrange our model so we can look at it both in detail and, when needed, in larger chunks, hiding its internal details. The mind mapping technique is an excellent example of how good structures can increase our ability to manage a large number of details.

We find lot of things when modeling but what do we find? Purpose is a key. Utility is in the eye of beholder. We cannot tell if we are going in the right direction until you know where we want to go. So we propose to make the use of Goal-Question-Metric approach to model the

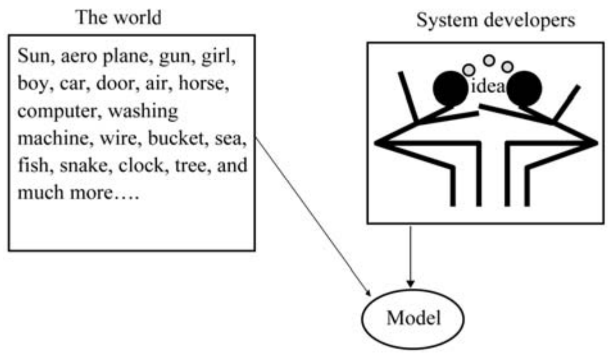


**Figure 1. Real objects and ideas are put in the model.**

objects or our system(in broad sense).

The GQM approach to process and metrics, first suggested by Basili and his colleagues, has proven to be a particularly effective approach to selecting and implementing metrics [Basili and Weiss, 1984; Basili and Rombach, 1988].

The GQM paradigm is explained in Section 2.

This paper is organized as follows: in Section 2, GQM paradigm is explained. Section 3 focuses on the way to model the reality, based on GQM. Section 4 concludes the paper.

## 2. The GQM Paradigm

Many metrics programs begin by measuring what is convenient or easy to measure, rather than by measuring what is needed. Such programs often fail because the resulting data are not useful to the developers and maintainers of the software. A measurement program can be more successful if it is designed with the goals of the project in mind. The GQM approach provides a framework involving three steps [2,3]:

1) List the major goals of the development or maintenance project.

2) Derive from each goal the questions that must be answered to determine if the goals are being met.

3) Decide what must be measured in order to be able to answer the questions adequately.

The GQM paradigm provides a method for top-down metric definition and bottom-up data interpretation (**Figure 2**).

## 3. Relationship of GQM and Model of the Identified Objects in Theoretical Perspective

We know the various methods to identify the potential objects like Use case text, CRC models, etc. Once we find the potential objects, to model them, we demonstrate that Goal-Question-Metric will be able to help in this regard. We must follow the following steps:

1) What is our purpose-our intention? Our purpose is obviously to find the potential objects. So the GOAL is to model the potential objects.

2) Now we should ask various set of QUESTIONS like what are the requirements of our system, what are

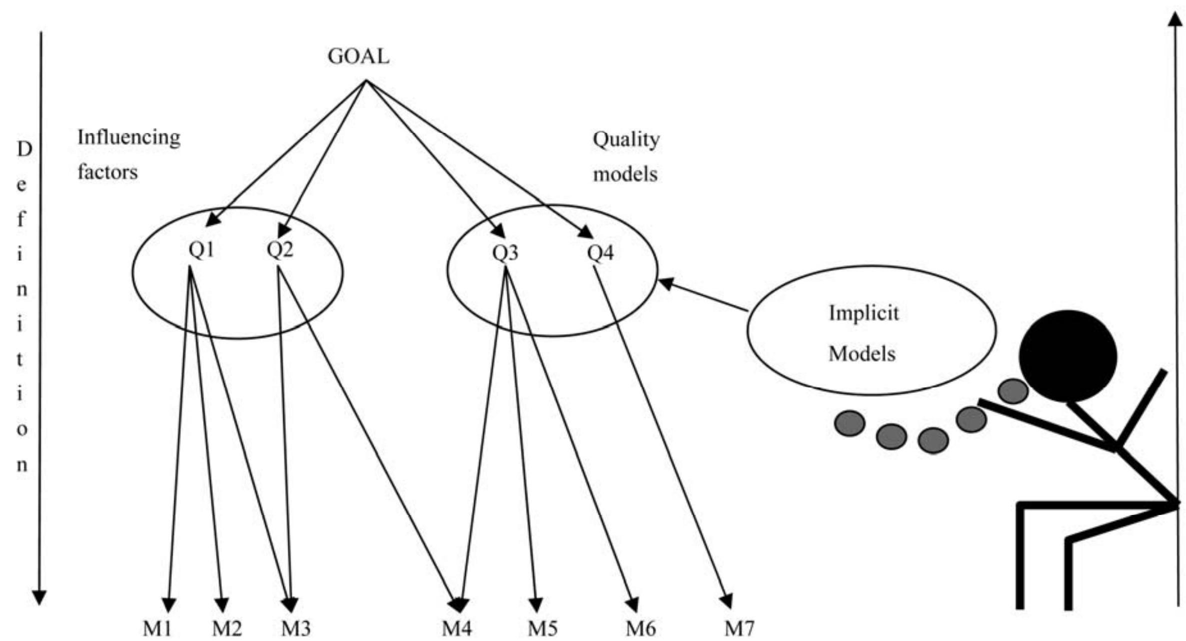


**Figure 2. The Goal-Question-Metric paradigm.**

the possible ways through which objects can be shown or in what possible forms the objects can be shown. These questions can be answered with the help of requirements specification or analysis models.

3) Now we can select the required form of the objects with the help of domain knowledge or requirements specification.

## 4. Applying GQM to Model the Reality in Object Form

We feel that the GQM paradigm can be adequately used to model (object derivation) the reality because the reality can be modeled in number of ways, depending on the Goal. This can be illustrated with the help of following two examples:

**Example 1** [3]. Nature of fundamental particles likes electrons, protons, neutrons, etc. It became palpable that the particles did not always behave as something that had a well-defined shape and size. Apparently, the fundamental particles in some experiments behaved as particles with characteristics like mass and electric charges, where as in other experiments they behaved as waves producing diffraction patterns (see **Figure 3**). The model needed to explain an experiment depended on the intent or goal of the experiment.

**Example 2**. Classification of Nuclear Power Plant (NPP)-the people who live near to NPP will classify it in one way (probably launch complaint about the radiation exposure or machinery noise). The people who use its electricity will classify it in some different way (probably like a dynamo without their own control). The people who are into maintenance of NPP will classify it in other way.

When we try to describe computer systems, so various purposes also yield various models. We always have to define a clear goal, otherwise modeling situation will soon become very awkward indeed. Let us assume that in Nuclear Power Plant there is a Nitrogen Tank to create some Pressure, may be for process requirement. Being an important object, NPP Engineering group also store many tanks in the store. Now this situation can be modeled by GQM paradigm as in **Figure 4**. If the purposes of modeling nitrogen tank because we need it to pressurize the system, then it is better to model two different simple objects rather than one complex object, else to model only one complex object is sufficient (see **Figure 5**). we shall store a tank in a store is very different from the purpose of modeling a nitrogen tank because we need it to pressurize the system, then it is better to model two different simple objects rather than one

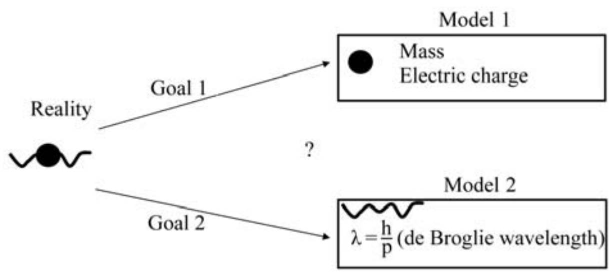


**Figure 3. An electron can behave like a wave or a particle depending on goal.**

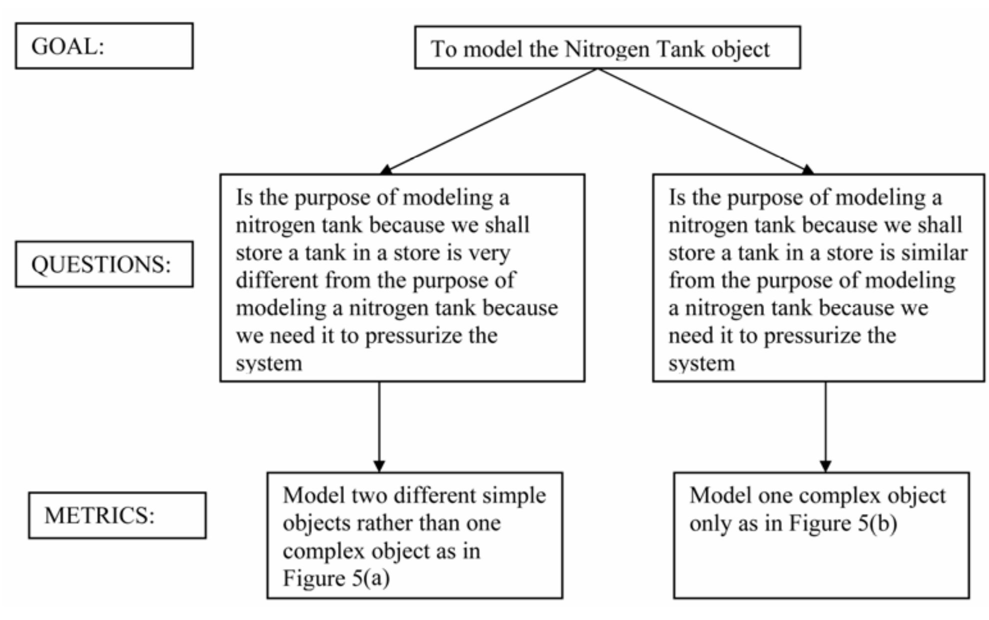


**Figure 4. Use of GQM to model the object (Nitrogen Tank) of NPP.**

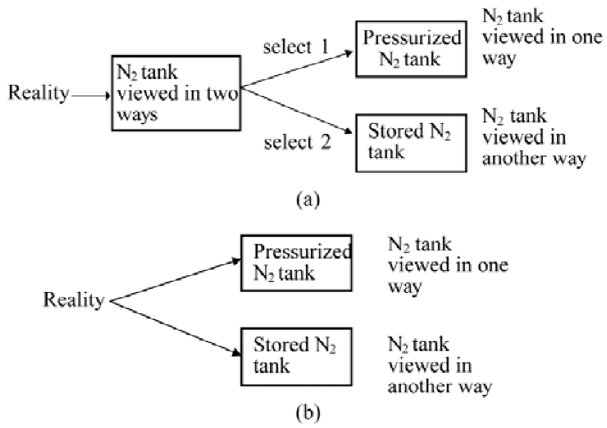


**Figure 5. (a) One way to model nitrogen tank, (b) Other way to model nitrogen tank.**

complex object, else to model only one complex object is sufficient. The above concept has also been implemented in Document/View Architecture [4] in (Microsoft Foundation Classes) MFC. The MFC multiple-document interface (MDI) architecture is focused on opening multiple documents of the same class. The document class contains the internal representation of the application data. For each new or opened document the framework creates an instance of Cdocument. The frame class describes the user interface of document windows of the application, typically MDI frames. The view class shows a [5] graphical representation of the document type. All classes are as CObject subclasses. In MFC the CDocument based classes act as observers for the views attached to them. Whenever user modifies document data via one of the views the view notifies the associated document by calling its UpdateAllViews() member. The document then notifies all views that are attached to it by calling their OnUpdate() member. The views can then redraw their data based on the hints passed to them as parameters (see **Figure 6**).

The above method is just like a brainstorming in which our set of questions is a tool to make a discussion take a new and different direction.

## 5. A Comparative Study in between the Classical Approach and the Proposed Philosophical Approach

The classical approach to model the objects is very nicely given in the text book by Pressman [6], in which a data object can be treated an external entity, anything that produces or consumes information, a thing like report, an occurrence like telephone call, event, a role, a place or a structure. It has been stated that according to the requirements the data object should be modeled but no technique or model is given for that. In this case a modeler can model the object which may lead to many changes in the future, may be during development phase.

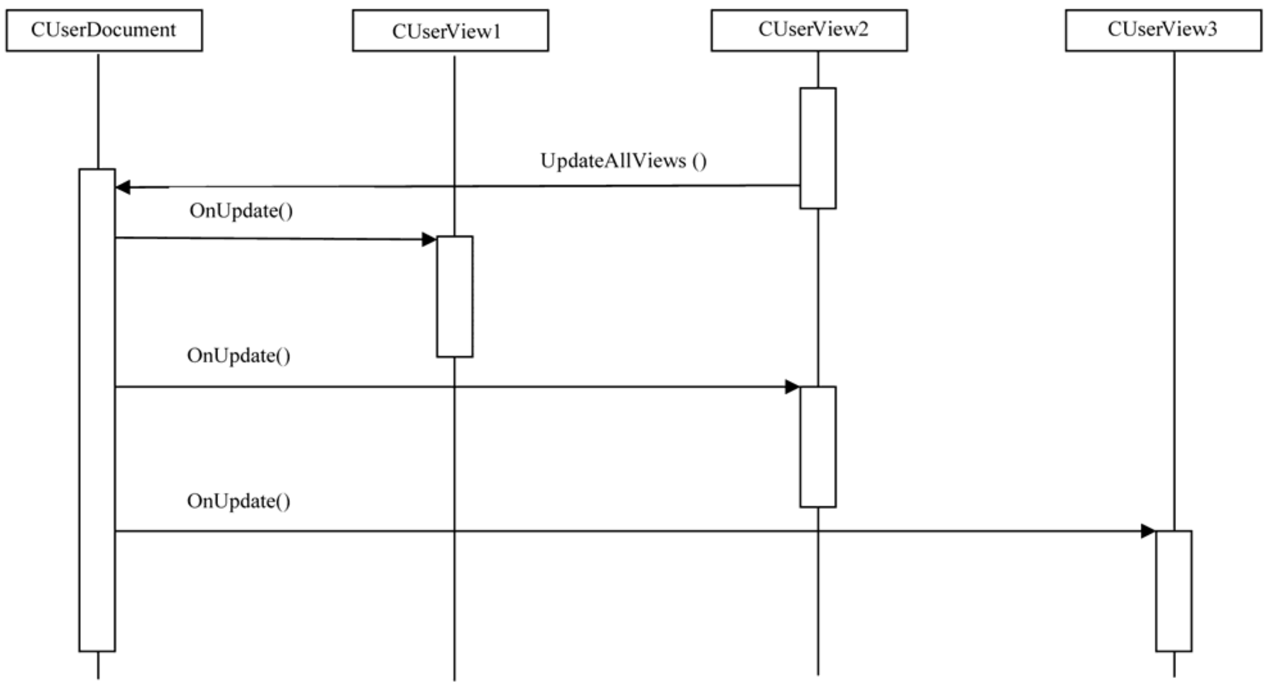


**Figure 6. Observer in document/view model.**

So modeler may have to re-model the objects which could be very costly and time consuming, which can slip off the deadline. As we have seen above, our method provide a clear cut method to model the object during the analysis or design phase to overcome the flaws, using the traditional method.

## 6. Conclusions

The concept of Goal-Question-Metric paradigm [7-8] has been proposed to model the identified objects, which are the basic constructs of model, of the software system, in better way for the improvement of non functional properties, which are the quality attributes of software. Two examples have been used for illustration.

## REFERENCES

[1] S. Sigfried, "Understanding Object-Oriented Software Engineering," *IEEE Press Understanding Science & Technology Series*, Piscataway, New Jersey, 1995.

[2] N. E. Fenton and S. L. Pfleeger, "Software Metrics, A Rigorous and Practical Approach," 2nd Edition, International Thomson Computer Press, Boston, 1996.

[3] R. V. Solingen, *et al.*, "Improvement by Goal-Oriented Measurement," *Proceedings of European Software Engineering Process Group Conference*, 16-20 June 1997, Amsterdam, The Netherlands, p. 11.

[4] SDI (Single Document Interface) and MDI (Multiple Document Interface) Documents, "Views and Framework (MFC)," 2005. http://www.msdn.microsoft.com/en-us/library/b2kye6c4(v=**vs**.80).aspx

[5] Multiple Document Interface, "Multiple Document Inference," August 2007. http://www.en.wikipedia.org/wiki/Mul-tiple_document_interface

[6] R. S. Pressman, "Software Engineering: A Practitioner's Approach," McGraw-Hill Science Engineering Publication, New York, 2004.

[7] L. Cyra, *et al.*, "Extending GQM by Argument Structures," *International Federation for Information Processing* (*IFIP*), Vol. 5082, 2008, pp. 26-39.

[8] F. V. Latum, R. V. Solingen, M. Oivo, B. Hoisl, D. Rombach and G. Ruhe, "Adopting GQM-Based Measurement in an Industrial Environment," *Software IEEE*, Schlumberger RPS, Vol. 15, No. 1, 1998, pp. 78-86.

 