IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. VIII (May-Jun. 2014), PP 01-15
www.iosrjournals.org

A Quantitative Measurement of Software Requirement Factors using Goal Question Metric (GQM) Approach

S Raju[1], Dr G V Uma[2]
[1]*Associate Professor, Department of Computer Science & Engineering, Sri Venkateswara College of Engineering, Sriperumbudur, India - 602 117 / Research Scholar, Anna University, Chennai.*
[2]*Professor, Department of Information Science & Technology College of Engineering Guindy, Anna University, Chennai, India - 600 025*

Abstract: *Every application software that we propose to develop can be thought of a group or collection of related requirements specified by the user and these requirements are the foundation from which the quality of the software is measured. Until now there is no concrete measurement methodology for measuring the requirements of any application software. A new value driven quantitative measurement approach is presented in this research work. These values associated with each of the requirements are collectively called as Requirement Factor Values (RFV). This new measurement method uses the Goal-Question-Metric (GQM) approach and it is one of the most powerful approaches available for metrics measurement. In this work, seven factors - customer priority of requirements, implementation complexity, changes in requirements, fault impact of requirements, completeness traceability and execution time are considered to be the primary ingredients of every requirement of the software and every requirement is characterized by the numerical values of these factors. A Java based application system has been developed that takes questionnaires, various parameters/values associated with each of the requirements and generates these factor values for each requirement of the software. These requirement factor values can be used in many ways such as measuring the quality of the software, test case design and optimization, test case prioritization.*

Keywords: *GQM. Software Metrics, Requirement Factor values, Software requirements*.

I. INTRODUCTION

1.1 Software Requirements Issues

Accurate and complete requirements are the most important elements that leads to get right development within a project's lifecycle. They ensure products meet customer and business expectations while enabling development teams to work smarter, not harder. Identifying and writing software requirements specification includes the following challenges.

- Inadequate or incomplete requirements that don't clearly meet the needs of the users
- Vague and ambiguous requirements that lead to project rework, scope creep and analysis paralysis
- Underestimating the value of spending time on requirements; providing only a small window to do the work
- Requirements that don't take into account and prioritize business needs and resource constraints
- Struggling with what are requirements and who does them
- Banking on requirements tools that don't help or trying to determine what requirements activities should be automated
- Adopting requirements practices to work effectively with Agile projects

By recognizing the potential impact of risks of these requirements, steps can be taken to turn them into strengths. Instead of requirements being the source of problems, a disciplined software requirements process can help to assure the success of the software projects.

Developing and managing effective requirements practices create significant benefits as listed below.

- Increased speed and efficiency in delivering high value products
- Greater insights into development capacity and capabilities
- Deeper understanding of customer and business needs
- Higher customer satisfaction
- Closer alignment with business goals and expectations
- Stronger team morale and personal satisfaction

Software Requirements Analysis is a primary step in software development life cycle. The process of preparing requirements specification depends on the following issues.

- completeness, consistency and accuracy of information domain analysis

- completeness of problem partitioning
- defining external and internal interfaces properly
- design of proper model that reflect data objects, their attributes and relations
- requirements traceability to system level
- Conducting prototype for the user
- consistency of requirements with schedule, resources and budget
- completeness of validation criteria

1.2. Software Design Issues

Software Design phase includes preliminary design reviews that addresses the following issues.

- reflection of the software requirements in the architecture
- achievement of effective modularity
- definition of interfaces for all necessary modules
- consistency of the data structure with the information domain
- consistency of the data structure with the requirements
- consideration of maintainability
- accomplishment of the desired function by the algorithm
- logical correctness of algorithm
- consistency of the interface with the architectural design
- reasonable logical complexity
- defining local data structures properly
- amenability of the design detail to the implementation language

1.3. Software Implementation and Coding Issues

Software Coding phase is the point where the source code is developed and it is concerned with the following questions.

- proper translation of the design into code
- proper use of language conventions
- compliance with coding standards for the language style
- proper declarations of data and its types

1.4. Software Testing and Maintenance Issues

Software Testing and maintenance phase consisted of the following issues.

- identification of major test phases
- early demonstration of major functions
- consistency of the test plan with the overall project plan
- availability and identification of the test resources
- establishment of traceability to validation criteria conducted during the software requirements analysis
- consideration of the side effects associated with change
- documentation and reporting of the change made

1.5. Importance and Role of Software requirements Measurement

Software requirements express the needs and constraints placed on a software product to be developed. These requirements highly contribute to the development of appropriate solutions of some real-world problems. A software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem. The problem may be to automate part of a task of someone who will use the software, to support the business processes of the organization that has commissioned the software, to correct shortcomings of existing software, to control a device, and many more. An essential property of all software requirements is that they must be verifiable. It may be difficult or costly to verify certain software requirements. Today almost all business involves the development or use of the software. But most of the software application system lack of quality. Also developers are unable to deliver the software system in time and the software development cost always exceeds the planned budget [1].

Every application software that is considered for development can be thought of a group or collection of related requirements specified by the user. These requirements are considered to be the foundation of software-to-be-developed and based on which the quality of the software is measured. Until now there is no concrete measurement methodology for measuring the requirements of any application software. A new value driven quantitative measurement approach is presented in this research work. A set of values associated with

each of the requirements are determined using the GQM Approach and these values are collectively called as *Requirement Factor Values* (RFV).

In this proposed work, every requirement is characterized based on seven factors - customer priority of requirements, implementation complexity, changes in requirements, fault impact of requirements, completeness traceability and execution time. These factors are considered to be the primary ingredients of every requirement of the software. Development of a Java based application system that takes questionnaires, various parameters/values associated with each of the requirements and generates these factor values is considered as part of this research work. These requirement factor values can be used in many ways such as measuring the quality of the software, test case optimization, test case prioritization.

II. THE GOAL-QUESTION-METRIC APPROACH AND RELATED WORKS

2.1. The GQM Concept

The Goal/Question/Metric approach (GQM) is a method for performing empirical studies on software projects The GQM method was developed for multi-purpose evaluation of software. It was designed by Victor Basili and it is a system of questions and simple answers for evaluation of properties. GQM defines a measurement model on three levels [2]:

- Conceptual level (Goal)
  A goal is defined for an object for a variety of reasons, with respect to various models of quality, from various points of view and relative to a particular environment.
- Operational level (Question)
  A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal.
- Quantitative level (Metric)
  A set of metrics, based on the models, is associated with every question in order to answer it in a measurable way.

GQM [3] is a framework for the definition of metrics. GQM is based on the assumption that in order to measure in a useful way, an organization must:

- specify goals,
- characterize them by means of questions pointing their relevant attributes,
- give measurements that may answer these questions.

They have chosen this framework because it is a top down approach that provide guidelines to define metrics, without a priori knowledge of the specific measures. Following GQM, they stated which dimensions characterize the notion of data quality. Then, a set of questions were asked to characterizing each dimension, without giving a precise (formal) definition -that is sometimes impossible-, only focusing on their relevant characteristics. Finally, they generated metrics (some objective, some others based on people appreciation) to answer these questions, giving a more precise valuation of the quality of data.

A *goal* in GQM is defined in a precise way. A goal is defined for an *object*, with a *purpose*, from a *perspective*, in an *environment*. For example, in a software organization, "To evaluate the maintenance process from the manager point of view in the context of a maintenance staff comprised of new programmers." In this example, the object is the maintenance process, the purpose is to evaluate, the perspective is the manager point of view, and the environment is the composition of the maintenance staff. A goal in GQM is posed at the conceptual level. A *question* in GQM tries to characterize the object of measurement with respect to a selected quality issue, and to determine its quality from the selected viewpoint. For example, in the context of the goal stated above, "What is the current change processing time?" A question in GQM is posed at the operational level. A *metric* in GQM is a set of data associated with every question in order to answer it in a quantitative way. Data can be *objective*, if it depends only on the object being measured and not on the viewpoint, or *subjective*, if it depends on both. For example: "Number of days spent on a user change request" may be a metric for the question presented above. A metric in GQM is posed at the quantitative level. Here, in order to have a concrete way to compute the metrics, they also gave techniques associated with them. Various authors have addressed the usage of the GQM approach and its advantages as explained in [4]-[13].

2.2. Steps of GQM Measurement

Processing GQM consisting of performing six important steps as explained in many articles [4-13]. These steps are as follows.

1. "Characterize the environment". In this step you characterize the context in which your improvement program takes place.
2. "Identify measurement goals and develop measurement plans". Refine the improvement goals to measurement goals regarding the context analyzed in step 1. To do that easily, it's possible to apply GQM

templates to the goals. E.g.: "Analyze the development process for the purpose of change with respect to correctness from the viewpoint of the developer in the context of Project X."
3. "Define data collection procedures". Define data collection procedures for all measures identified during step 2. This includes the "who", "how" and "when".
4. "Collect, analyze and interpret data".
5. "Perform post-mortem analysis and interpret data". Analyze data again from the viewpoint of the organization. Write down the lessons learned in this particular project.
6. "Package experience". Structure all results of the analysis and store it in a reusable way (e.g. in an "experience factory").

2.3. Components of a GQM Plan

A GQM plan contains all the information, which is needed to plan measurement and to perform data analysis. The plan defines precisely why the measures are defined and how they are going to be used. It consists of a goal, questions, measures and models. In addition, there are multiple abstraction sheets which support the communication with different viewpoints. Common components [4],[8],are listed below.
1. **Questions**. They are written in natural language, mainly to make a GQM more understandable for humans. E.g. "How many failures are found by performing the test cases?". Similar to GQM templates, there are predefined categories for questions.
2. **Measures**. Measures provide operational definitions for attributes related to the defined goals (e.g. productivity or complexity). This includes defining its measurement scale and its range. This helps identifying abnormal values.
3. **Models**. Models are necessary to interpret the data given by a measurement.
4. **Abstractions Sheets**. The viewpoint does not need all the details of the GQM plan. Thus, the measurement analyst creates different Abstraction sheets suitable for different roles. In order to capture the experience of the viewpoints, the GQM abstraction sheets are used as knowledge acquisition instrument during interviews.

2.4. Advantages of GQM Approach

The GQM approach has several advantages. It helps to:

- ensure adequacy, consistency, and completeness of the measurement plan and therefore of data collection.
- manage the complexity of the measurement program. Increased complexity occurs when there are too many attributes to measure and too many possible measurement scales for each attribute.
- stimulate a structured discussion and promote consensus about measurement and
- improvement goals, which is a prerequisite for measurement success.

III. SOFTWARE REQUIREMENT FACTORS

3.1. Software Requirement Factors

In this proposed work, the factors that influence the requirements are identified as (1) Customer priority of requirements (2) Implementation complexity (3) Changes in requirements (4) Fault impact of requirement (5) Completeness, (6) Traceability and (7) Execution time related data. Subsequently the impacts of these factors on the requirements are quantified by the GQM approach, deriving values on a ten point scale.

3.1.1 Customer Priority (CP)

It is a measure of the importance of a requirement to the customer The value for each requirement derived applying the goal-question-metric approach and ranges from 1 to 10 where 10 indicates the highest customer priority.

Reasoning: A focus on customer requirements for development improves the customer satisfaction. So, the requirements that would be of highest importance to the customer should be tested early and thoroughly to improve the customer satisfaction.

3.1.2 Implementation Complexity **(IC)**

It is a subjective measure of how difficult the development team perceives the implementation of requirement to be. Each requirement is analyzed for its implementation complexity and a value ranging from 0 to 10 is derived by the developer. A larger value indicates higher complexity.

Reasoning: Requirements with high implementation complexity is expected to have a higher number of defects.

3.1.3 Requirement changes (RC)

It is based on the number of times a requirement has been altered in the development cycle with respect to its origin date and the nature of changes. The derived value ranges from 1 to 10.

Reasoning: Roughly 50% of all defects identified in the project are the errors introduced in the requirement phase. The significant factor that causes the failure of the project is attributed to changing requirements.

3.1.4 Fault Impact of requirements **(FI)**

It allows the development team to identify the requirement that have customer reported failures. As a system evolves to several versions, the developers can use the prior data collected from versions to identify requirements that are likely to be error prone. FI is based on the number of field failures and in-house failures. FI is considered for those requirements that have already been in a released product. In this research work, we propose to calculate Fault Impact of a requirement applying the goal-question-metric approach based on the set of defects identified in the previous run.

Reasoning: Test efficiency can be improved by focusing on the function that is likely to contain higher number of defects.

3.1.5 Completeness **(CT)**

One element of "requirement completeness" is a test to determine that each requirement, *individually*, is complete for the conditions under which the function is to be performed. Each requirement is analyzed for its completeness and a value is derived, ranging from 0 to 10.

Reasoning: Customer satisfaction such as the quickness of the software response to the user request can be improved by considering the completeness of the requirement.

3.1.6 Traceability **(TR)**

Requirements traceability refers to the "ability to follow the life of a requirement, in both forward and backward direction, i.e., from its origin, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases. Each requirement is analyzed for its traceability and a value is proposed, ranging from 0 to 10.

Reasoning: The quality of the software can be improved by considering the traceability of the requirement.

3.1.7 Execution Time related data **(ET)**

Execution time related data consists of different categories of data related to time constraints, reliability issues, environmental issues and software release constraints. The measurement method using the GQM approach is shown in the table given below. Each requirement is analyzed considering execution related data and a value is proposed, ranging from 1 to 10.

Reason: A focus on execution time related data for project development improves the schedule and reduce the cost of maintenance thereby improving the level of customer satisfaction.

3.2 SUMMARY

Summary of software requirement factors, their value range and significance are listed in the table 3.1.

TABLE 3.1 SOFTWARE REQUIREMENT FACTORS

S. No	Requirement Factors	Value Range	Significance
1	Customer Priority (CP)	1 to 10	Measure of importance of customer's need
2	Implementation Complexity (IC)	0 to 10	Measurement of how complex to implement the customer's need.
3	Requirements Change (RC)	1 to 10	The number of times a requirement is changed from the starting of the project
4	Fault Impact (FI)	1 to 10	Taken from the past projects. Based on the severity of the fault identified in the previous run..
5	Completeness (CT)	0 to 10	The degree of success of execution of a requirement
6	Traceability (TR)	0 to 10	Traceability of requirement ie,. ability of monitoring the life of a requirement.
7	Execution Time Related Data (ET)	1 to 10	Execution time related data.

All the above mentioned factors are considered to be main ingredients of software requirements and hence they are taken for measurement using the goal-question-metric approach explained in the following section.

IV. MEASURING SOFTWARE REQUIREMENT FACTORS

4.1 Measuring Requirement Factors

For every requirement, the above seven factors are measured by farming questionnaire. The following section explains how the seven factor values are calculated using the GQM approach. Each sub-goal is assigned an appropriate weight, arbitrarily by developer/tester depending upon the importance of the sub-goals. All the seven factors (Goals), their sub-goals along with their weights are listed in the table.4.1 below.

TABLE 4.1. SOFTWARE REQUIREMENT FACTORS AND THEIR SUB-GOALS

S. No	Requirement Factors	Sub-Goals
1	Customer Priority (CP)	1. Resources
		2. Budget
		3. Schedule
		4. Quality
2	Implementation Complexity (IC)	1. Plan
		2. Budget
		3. Resources & New Technology
3	Requirements Change (RC)	1. Quality
		2. Productivity
		3. Management Commitment
4	Fault Impact (FI)	1. Sources
		2. Failure Modes
5	Completeness (CT)	1. System Consistency
		2. System Availability
6	Traceability (TR)	1. Source Traceability
		2. Requirements Traceability
		3. Design Traceability
7	Execution Time Related Data (ET)	1. Time Constraints
		2. Reliability Issues
		3. Environmental issues
		4. Software Release constraints

A numerical quantity or value can be considered for the questions in the process of metrics measurement. These numerical quantity may either binary values or non-binary values. These vales can be multiplied by respective factor weights of each sub-goal and then mean value is obtained for each of the requirement factors.

4.1.1 Measuring Customer Priority (CP)

It is a measure of the importance of a requirement to the customer The value for each requirement derived applying the goal-question-metric approach and ranges from 1 to 10 where 10 indicates the highest customer priority. The GQM plan for measurement of customer priority is shown in the table 4.2 below.

TABLE 4.2 GQM PLAN FOR CUSTOMER PRIORIT

GOAL	Customer Priority	Questions
Sub-Goals	1. Resources	1. Are all product licenses current?
		2. Highly skilled people available?
		3. Better build teams available?
		4. Better tools available?
		5. Have all programmers received proper training on the tools?
		6. Do project team members have sufficient workstations?
		7. Does the project have sufficient build & test environments?
	2. Budget	1. Has got properly planned budget?
		2. Is management flexible to relax the budget constraints?
		3. Has requirement a lower priority than the other requirements
	3. Schedule	1. Whether cost over-run fixed?
		2. Has requirement a lower priority than the other requirements
		3. Has project schedule approved by top management
	4. Quality	1. Whether all requirements have been collected from various sources?
		2. Whether the user drives system functionality or the programmer?
		3. High-level languages used?
		4. Are languages better at expressing programming concepts than others?
		5. Are programmers more productive using a familiar language?
		6. Whether requirements Specified adequately?
		7. Whether all requirements have been collected from various sources?

4.1.2 Implementation Complexity **(IC)**

It is a subjective measure of how difficult the development team perceives the implementation of requirement to be. Each requirement is analyzed for its implementation complexity and a value ranging from 0 to 10 is derived by the developer. A larger value indicates higher complexity. The GQM plan for measurement of Implementation Complexity is shown in the table 4.3 below

TABLE 4.3 GQM PLAN FOR IMPLEMENTATION COMPLEXITY

GOAL	Implementation Complexity (IC)	Questions
Sub-Goals	1. Plan	1. Level of adherence to the time plan?
		2. Delivery of the new system and procedures on time?
		3. Improvement of the development process?
		4. Does external deadlines restrict?
	2. Budget	1. Delivery of the new system and procedures within budget.
		2. Flexibility of budget policy, in terms of time and cost restrictions.
		3. Level of adherence to budget.
	3. Resources & New Technology	1. Level of contribution of the program deliverable to the company
		2. Level of use of the new system and procedures?
		3. Reduce tedious and redundant tasks

4.1.3 Requirement changes **(RC)**

It is based on the number of times a requirement has been altered in the development cycle with respect to its origin date and the nature of changes. The derived value ranges from 1 to 10. The GQM plan for measurement of Requirements Change is shown in the table 4.4 below.

TABLE 4.4. GQM PLAN FOR REQUIREMENTS CHANGE

GOAL	Requirements Change	Questions
Sub-Goals	1. Quality Improvement	1. Flexibility to Local responsiveness.
		2. Flexibility to sharing knowledge.
		3. Flexibility to transfer of competence.
		4. High-leverage opportunities for defect prevention.
	2. Productivity Improvement	1. Increasing of productivity over time.
		2. Handling of the project configuration management tool.
		3. Using of Tool to track project change requests.
	3. Management Commitment (towards Technology advancement and adoptability)	1. Level of change occurring due to advances in technology and communication.
		2. How effective have process changes been?
		3. Level of Change occurring due to global competition, acquisitions and alliances, organization restructuring?
		4. Level of change due to Organization restructuring?
		5. Level of change due to fostering horizontal communication?
		6. Level of change due to using cross-border and virtual teams?
		7. Level of change due to using international assignments?
		8. Level of change due to adopting a global “mindset”?

4.1.4 Fault Impact of requirements **(FI)**

It allows the development team to identify the requirement that have customer reported failures. As a system evolves to several versions, the developers can use the prior data collected from versions to identify requirements that are likely to be error prone. FI is based on the number of field failures and in-house failures. FI is considered for those requirements that have already been in a released product. In this research work, we propose to calculate Fault Impact of a requirement applying the goal-question-metric approach based on the set of defects identified in the previous run. The GQM plan for measurement of Fault Impact of requirements is shown in the table 4.5 below.

TABLE 4.5 GQM PLAN FOR FAULT IMPACT OF REQUIREMENTS

GOAL	Fault Impact (FI)	Questions		
Sub-Goals	1. Sources	1. Are specification Inadequate ?		
		2. Whether fault due to processor failure?		
		3. Whether fault due to interference on the communication subsystem ?		
		4. Whether design errors occur in software?		
		5. Whether design errors occur in hardware?		
		6. Whether system's external behaviour- mechanical?		
		7. Whether system's external behaviour- algorithmic?		
		8. Whether systems composed of too many components		
	2. Failure Modes	1. Value domain	1. Whether constraint Error?	
			2. Whether value Error?	
		2. Time domain	1. Occur early?	
			2. Due to Omission?	1. Is fail Silent?
				2. Is fail Stops?
				3. Is fail controlled?
			3. Occur late?	
		3. Arbitrary (uncontrolled)		

4.1.5 Completeness (CT)

One element of requirement completeness is a test to determine that each requirement, *individually*, is complete for the conditions under which the function is to be performed. Each requirement is analyzed for its completeness and a value is derived, ranging from 0 to 10. The GQM plan for measurement of Completeness Factor is shown in the table 4.6 below.

TABLE 4.6 GQM PLAN FOR COMPLETENESS

GOAL	Completeness (CT)	Questions
Sub-Goals	1. System Consistency	1. Level of variation exist in capabilities
		2. Level of variation exist in usage patterns
		3. Isolation of human dependent factors
		4. Isolation of system dependent factors
	2. System Availability	1. Performance
		2. Fault-tolerance
		3. Maintainability
		4. Accuracy

4.1.6 Traceability **(TR)**

Requirements traceability refers to the "ability to follow the life of a requirement, in both forward and backward direction, i.e., from its origin, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases. Each requirement is analyzed for its traceability and a value is proposed, ranging from 0 to 10. The GQM plan for measurement of Traceability Factor is shown in the table 4.7 below.

TABLE 4.7 GQM PLAN FOR TRACEABILITY

GOAL	Traceability (TR)	Questions
Sub-Goals	1. Source Traceability	1. Coverage?
		2. Effectiveness?
		3. Productivity?
		4. Validation?
		5. Maintenance?
	2. Requirement Traceability	1. Coverage?
		2. Effectiveness?
		3. Productivity?
		4. Validation?
		5. Maintenance?
	3. Design Traceability	1. Coverage?
		2. Effectiveness?
		3. Productivity?
		4. Validation?
		5. Maintenance?

***4.1.7 Execution Time related data* (ET)**

Execution time related data consists of sub-goals such as time constraints, reliability issues, environmental issues and software release constraints. The measurement method using the GQM approach is shown in the table given below. The GQM plan for measurement of Execution Time related data is shown in the table 4.8 below.

TABLE 4.8 GQM PLAN FOR EXECUTION TIME RELATED DATA

GOAL	Execution Time related data (ET)	Questions
Sub-Goals	1. Time constraints	1. Whether available execution time is consumed?
		2. Whether maximum level of reliability needed?
		3. Amount of code of the system Maximum?
		4. Whether time span between major changes Minimum.
		5. Whether large time spent for design before starting coding?
		6. Whether hardware planned concurrently?
		7. Whether project time compressed?
		8. Whether project time expanded?
	2. Reliability Factor	1. Projects Similarity
		2. Is data size large compared to the code?
		3. Whether functions of the software are complex?
		4. Whether structure of the software is complex?
		5. Whether components are reusable to maximum extent.
		6. Whether degree of complexity of the user interface is maximum.
		7. Is length of the project large.
	3. Environmental and other Issues	1. Is workplace suitable for creative works.
		2. Whether environmental factor describes the ratio of uninterrupted hours and present hours.
		3. Whether the team members are distributed over the building?
		4. Whether the team members are distributed over multiple sites?
		5. Facilities Support for work at home, virtual teams, video conferencing with clients.
	4. Software Release constraints	1. Whether software release time measured by calendar time?
		2. Whether cumulative test-execution time measured by CPU time?
		3. Whether software-testing cost measured per unit calendar time?
		4. Whether software-testing cost measured per unit execution time?
		5. Whether software fault correction / removal cost measured per unit fault in testing phase?
		6. Whether software fault correction / removal cost measured per unit fault in operational phase?
		7. Whether lifetime of software product measured under stated constant?

4.2. Computing Factor Values

The simple algorithm shown in the figure 1 below, computes factor values using GQM Approach. TW represents sum of weights of sub-goals of seven requirement factors.

```
For each Requirement, Ri
Begin
   fv = 0
   For each Sub-Goal
   Begin
         fv = fv+ (Value of each question X Weight
                        of the respective sub-goal)
   End
   FV = fv / TW
End
```

Figure 4.1 General algorithm for computing Factor Values

V. CASE STUDIES AND APPLICATION OF GQM

Consider the process of developing a safety-critical software project as a case study. The project development team shall ensure that the necessary requirements are implemented in the software. The necessary requirements are listed in the table 5.1 below.

TABLE 5.1 REQUIREMENTS OF A CASE STUDY APPLICATION

Requirements	Purpose
R1	Safety-critical software is initialized, at first start and at restarts, to a known safe state
R2	Safety-critical software safely transitions between all predefined known states
R3	Termination performed by software of safety critical functions is performed to a known safe state
R4	Operator overrides of safety-critical software functions require at least two independent actions by an operator
R5	Safety-critical software rejects commands received out of sequence, if execution of those commands can cause a hazard
R6	Safety-critical software detects inadvertent memory modification and recovers to a known safe state

These requirements are only indicative guidelines and not exhaustive set of all requirements. Each requirement is taken for measurement following the above GQM procedure. Here we will consider some of the requirements only for demonstration purpose. We can use either binary values or numeric values as rating for each of the questions depending upon satisfaction, against each requirements. The binary values represent either presence or absence of the respective attributes of the factors. The non-binary values tells their significance as rating. These values are multiplied by the weights of the sub-goals and then divided by the total weight of the sub-goals to find the requirement factor values as shown in the algorithm. The requirement factor values (RFV) of all the requirements are shown in the table 5.2 below.

TABLE 5.2. REQUIREMENT FACTOR VALUES

Requirements (R_i)	Requirement Factor Values						
	CP	IC	RC	FI	CT	TR	ET
R1	2.6	6.3	6.0	2.0	5.0	4.8	3.5
R2	3.4	7.3	7.0	2.5	5.7	5.0	3.9
R3	2.1	5.3	5.3	3.7	5.7	4.8	3.1
R4	2.8	3.3	6.3	4.0	5.0	5.5	3.7
R5	2.4	5.3	4.7	0.0	5.7	5.3	4.2
R6	2.8	4.0	5.7	5.0	4.3	5.3	4.4

The complete procedure for arriving the above values along with explanation is shown in the Tables A-1.1 to A1.8 of appendix A.

V. CONCLUSION AND FUTURE WORKS

Functions and/or modules of every applications software implements the user requirements. These requirements can be measured in many ways. A new value driven approach using the powerful GQM approach is presented in this research work. It uses seven factors to that influence the requirements of software applications. This novel approach can be applied to any type of software. These requirement factor values can be beneficial in many ways. These values and their Mean RFV can be used in Software Quality Assurance activities such as Test case prioritization and optimization.

References

[1] Gibbs, W W., Software Chronics Crises, Scientific American, 1994.
[2] Basili, V., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach, Encyclopedia of Software Engineering, John Wiley & Sons, New York,1994, ISBN 0-471-54004-8
[3] Basili, V.R., Rombach, H.D.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, vol. 14, no. 6, June 1988.
[4] Dominik Richter, Michael Hohenstein, Stefan Templin and Gero Zimmer, "GQM Best Practice, Technical report"
[5] Peter Hantos, Xerox Corporation ," Strengths and Weaknesses of the GQM Approach in Developing Software Size Metrics", 13th International Forum On COCOMO And Software Cost Modeling.
[6] Aziz Deraman, Jamaiah H. Yahaya, Zaiha Nadiah Zainal Abidin and Noorazean Mohd Ali (2014), "Software Ageing Measurement Framework Based on GQM Structure", Journal of Software and Systems

[7] Abdul Azim Abdul Ghani Koh Tieng Wei Geoffrey Muchiri Muketha Wong Pei Wen "Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM)", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.5, May 2008 219

[8] Salvatore Alessandro Sarcia' Italian MoD, "Is GQM+Strategies Really applicable as is to non-Software Development Domains?", ESEM'10, September 16-17, 2010, Bolzano-Bozen, Italy., Copyright 2010 ACM 978-1-4503-0039-01/10/09

[9] Norazlina Khamis, Sufian Idris, and Rodina Ahmad, "Applying GQM Approach towards Development of Criterion-Referenced Assessment Model for OO Programming Courses", International Journal of Computer, Information Science and Engineering Vol:1 No:8, 2007, World Academy of Science, Engineering and Technology

[10] Tong Chen, Behrouz Homayoun Far, Yingxu Wang, "Development of an Intelligent Agent-Based GQM Software Measurement System", Proceedings of ATS 2003 188

[11] Victor BasiliB,C, Jens HeidrichA, Mikael LindvallB, Jürgen MünchA, Myrna RegardieB, Adam TrendowiczA "GQM+Strategies – Aligning Business Strategies with Software Measurement", IEEE, First International Symposium on Empirical Software Engineering and Measurement, 0-7695-2886-4/07 © 2007 IEEE, DOI 10.1109/ESEM.2007.66

[12] Jurgen Munch, Fabian Fagerholm, Petri Kettunen, Max Pagels, Jari Partanen, "Experiences and Insights from Applying GQM+Strategies in a Systems Product Development Organisation", IEEE, Proceedings of the 39th Int'l Conference on Software Engineering and Advanced Applications (SEAA 2013).

[13] Akito Monden, Tomoko Matsumura, Mike Barker, Koji Torii, Victor R. Basili , "Customizing GQM Models for Software Project Monitoring", IEICE Transactions on Information and Systems, Vol.E95-D, No.9, pp.2169-2182, September 2012.

APPENDIX A

Table A-1.1 - Weight Assignment for the sub-goals

S. No	Requirement Factors	Sub-Goals	Weight
1	Customer Priority (CP)	1. Resources	1
		2. Budget	2
		3. Schedule	3
		4. Quality	4
2	Implementation Complexity (IC)	1. Plan	1
		2. Budget	1
		3. Resources & New Technology	1
3	Requirements Change (RC)	1. Quality	1
		2. Productivity	1
		3. Management Commitment	1
4	Fault Impact (FI)	1. Sources	1
		2. Failure Modes	2
5	Completeness (CT)	1. System Consistency	2
		2. System Availability	1
6	Traceability (TR)	1. Source Traceability	1
		2. Requirements Traceability	2
		3. Design Traceability	1
7	Execution Time Related Data (ET)	1. Time Constraints	2
		2. Reliability Issues	3
		3. Environmental issues	1
		4. Software Release constraints	4

Table A-1.2. Measuring Customer Priority (CP)

GOAL	Customer Priority	Questions (Binary values: 1-Y, 0-N)	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. Resources (Weight=1)	1. Are all product licenses current?	Y	Y	Y	Y	Y	Y
		2. Highly skilled people available?	N	Y	Y	Y	Y	Y
		3. Better build teams available?	Y	Y	Y	Y	Y	Y
		4. Better tools available?	Y	Y	Y	Y	Y	Y
		5. Have all programmers received proper training on the tools?	Y	Y	Y	Y	Y	Y
		6. Do project team members have sufficient workstations?	N	Y	Y	Y	Y	Y
		7. Does the project have sufficient	Y	Y	Y	Y	Y	Y

		build & test environments?						
	2. Budget (Weight =2)	1. Has got properly planned budget?	Y	Y	Y	Y	Y	Y
		2. Is management flexible to relax the budget constraints?	N	N	N	N	N	N
		3. Has requirement a lower priority than the other requirements	N	Y	N	N	N	N
	3. Schedule (Weight=3)	1. Whether cost over-run fixed?	N	N	N	N	N	N
		2. Has requirement a lower priority than the other requirements	N	N	N	N	N	N
		3. Has project schedule approved by top management	Y	Y	Y	Y	Y	Y
	4. Quality (Weight=4)	1. Whether all requirements have been collected from various sources?	Y	Y	N	N	Y	Y
		2. Whether the user drives system functionality or the programmer?	N	N	N	N	N	N
		3. High-level languages used?	Y	Y	Y	Y	Y	Y
		4. Are languages better at expressing programming concepts than others?	Y	Y	Y	Y	Y	Y
		5. Are programmers more productive using a familiar language?	N	Y	N	N	N	N
		6. Whether requirements Specified adequately?	N	Y	N	Y	N	N
		7. Whether all requirements have been collected from various sources?	Y	N	N	Y	N	Y
	Measure		2.6	3.4	2.1	2.8	2.4	2.8

Steps for Measuring Customer Priority: Number of sub-goals = 4, Total Weight, TW = 5, Y=1, N=0
For the above requirements, measuring Customer Priority is shown below.
For R1: ((5x1)+(1x2)+(1x3)+(4x4))/TW =26/10 =2.6 For R2: ((7x1)+(2x2)+(1x3)+(5x4))/TW =34/10 =3.4
For R3: ((7x1)+(1x2)+(1x3)+(2x4))/TW = 21/10 = 2.1
The remaining values are shown in the table itself. The same procedure is used for measuring other goals.

Table A-1.3. Measuring Implementation Complexity (IC)

GOAL	Implementation Complexity (IC)	Questions (Rate: 1-Poor, 2-Good, 3-Best)	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. Plan (W=1)	1. Level of adherence to the time plan?	2	1	1	1	2	1
		2. Delivery of the new system and procedures on time?	2	2	1	1	1	1
		3. Improvement of the development process?	2	2	1	1	2	2
		4. Does external deadlines restrict?	1	3	2	1	1	1
	2. Budget (W=1)	1. Delivery of the new system and procedures within budget.	2	2	2	1	2	1
		2. Flexibility of budget policy, in terms of time and cost restrictions.	1	1	2	1	2	1
		3. Level of adherence to budget.	3	2	1	1	2	1
	3. Resources & New Technology (W=1)	1. Level of contribution of the program deliverable to the company	2	3	2	1	2	2
		2. Level of use of the new system and procedures?	2	3	2	1	1	1

		3. Reduce tedious and redundant tasks	2	3	2	1	1	1
	Measure	Total Weight = 3	6.3	7.3	5.3	3.3	5.3	4.0

Table A-1.4. Measurement of Requirements Change (RC)

GOAL	Requirements Change (RC)	Questions (Responses;0-Low, 1-Medium, 2=high)	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. Quality Improvement (W=1)	1. Flexibility to Local responsiveness.	1	2	1	1	0	1
		2. Flexibility to sharing knowledge.	2	2	1	2	1	1
		3. Flexibility to transfer of competence.	0	2	1	1	0	2
		4. High-leverage opportunities for defect prevention.	2	1	1	1	1	2
	2. Productivity Improvement (W=1)	1. Increasing of productivity over time.	1	1	1	2	2	1
		2. Handling of the project configuration management tool.	1	2	1	2	1	2
		3. Using of Tool to track project change requests.	1	2	1	2	1	2
	3. Management Commitment (towards Technology advancement and adoptability) (W=1)	1. Level of change occurring due to advances in technology and communication.	2	1	2	1	1	1
		2. How effective have process changes been?	2	1	2	1	1	2
		3. Level of Change occurring due to global competition, acquisitions and alliances, organization restructuring?	1	2	2	1	2	0
		4. Level of change due to Organization restructuring?	2	2	1	0	0	1
		5. Level of change due to fostering horizontal communication?	1	2	1	2	1	1
		6. Level of change due to using cross-border and virtual teams?	1	0	0	0	0	0
		7. Level of change due to using international assignments?	0	0	0	1	1	0
		8. Level of change due to adopting a global mindset?	1	1	1	1	2	1
	Measure		6.0	7.0	5.3	6.3	4.7	5.7

Table A-1.5. Measuring Fault Impact (FI)

GOAL	Fault Impact (FI)	Questions (Responses: Y (1) / N (0)		Measures					
				R1	R2	R3	R4	R5	R6
Sub-Goals	1. Sources (W=1)	1. Are specification Inadequate ?		Y	Y	Y	N	N	Y
		2. Whether fault due to processor failure?		N	N	N	N	N	N
		3. Whether fault due to interference on the communication subsystem ?		Y	Y	Y	Y	N	Y
		4. Whether design errors occur in software?		N	Y	Y	Y	N	Y
		5. Whether design errors occur in hardware?		N	N	N	N	N	N
		6. Whether system's external behaviour-mechanical?		N	N	N	N	N	N
		7. Whether system's external behaviour-algorithmic?		Y	Y	Y	Y	N	Y
		8. Whether systems composed of too many components		Y	Y	Y	Y	N	Y
	2. Failure	1. Value	1. Whether constraint Error?	N	N	Y	N	N	Y

	Modes (W=2)	domain	2. Whether value Error?		N	N	N	Y	N	N
		2. Time domain	1. Occurs early?		N	N	Y	Y	N	Y
			2. Due to Omission?	1. Is fail Silent?	N	N	N	Y	N	Y
				2. Is fail Stops?	N	N	Y	N	N	Y
				3. Is fail controlled?	Y	N	N	N	N	N
			3. Occurs late?		N	Y	N	N	N	N
		3. Arbitrary (uncontrolled)			N	N	N	Y	N	Y
	Measure				2.0	2.5	3.7	4.0	0.0	5.0

Table A-1.6. Measuring Completeness (CT)

GOAL	Completeness (CT)	Questions (Rating:0-poor, 1-good, 2-best)	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. System Consistency (W=2)	1. Level of variation exist in capabilities.	0	1	1	0	1	0
		2. Level of variation exist in usage patterns.	1	1	1	1	1	1
		3. Isolating human dependent factors.	2	2	1	2	2	1
		4. Isolation of system dependent factors.	1	1	2	1	1	1
	2. System Availability (W=1)	1. Performance	2	2	2	2	2	2
		2. Fault-tolerance	2	2	2	2	2	2
		3. Maintainability	2	2	2	2	2	2
		4. Accuracy	1	1	1	1	1	1
	Measure		5.0	5.7	5.7	5.0	5.7	4.3

Table A-1.7. Measuring Traceability (TR)

GOAL	Traceability (TR)	Questions (0-Poor, 1-Good, 2-Best)	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. Source Traceability (W=1)	1. Coverage?	0	0	1	1	0	1
		2. Effectiveness?	1	1	1	1	1	1
		3. Productivity?	1	1	1	1	1	1
		4. Validation?	1	1	0	0	0	0
		5. Maintenance?	1	1	1	1	1	1
	2. Requirement Traceability (W=2)	1. Coverage?	1	1	1	1	1	1
		2. Effectiveness?	1	1	0	1	1	1
		3. Productivity?	1	1	2	1	1	1
		4. Validation?	1	1	1	1	1	1
		5. Maintenance?	1	1	1	2	2	2
	3. Design Traceability (W=1)	1. Coverage?	1	1	1	1	1	1
		2. Effectiveness?	1	1	1	1	1	1
		3. Productivity?	1	1	1	1	1	1
		4. Validation?	1	2	1	2	2	1
		5. Maintenance?	1	1	1	1	1	1
	Measure		4.8	5.0	4.8	5.5	5.3	5.3

Table A-1.8. Measuring Execution Time related data (ET)

GOAL	Execution Time Relate Data (ET)	Questions (Response: Y(1) / N (0))	Measures					
			R1	R2	R3	R4	R5	R6
Sub-Goals	1. Time constraints (W=2)	1. Whether available execution time is consumed?	N	N	N	N	Y	Y
		2. Whether maximum level of reliability needed?	Y	Y	Y	Y	Y	Y
		3. Amount of code of the system Maximum?	N	N	N	N	Y	N
		4. Whether time span between major changes Minimum.	N	N	N	Y	N	Y
		5. Whether large time spent for design before starting coding?	Y	Y	Y	Y	Y	Y
		6. Whether hardware planned concurrently?	N	N	N	N	N	N
		7. Whether project time compressed?	N	N	N	N	N	N
		8. Whether project time expanded?	Y	Y	Y	Y	Y	Y
	2. Reliability Factor (w=3)	1. Projects Similarity	Y	Y	Y	Y	Y	Y
		2. Is data size large compared to the code?	N	N	N	N	Y	Y
		3. Whether functions of the software are complex?	Y	Y	Y	Y	Y	Y
		4. Whether structure of the software is complex?	N	N	N	N	N	N
		5. Whether components are reusable to maximum extent.	N	N	N	N	N	Y
		6. Whether degree of complexity of the user interface is maximum.	N	N	N	N	N	N
		7. Is length of the project large.	Y	Y	Y	Y	Y	Y
	3. Environmental and other Issues (w=1)	1. Is workplace suitable for creative works.	Y	Y	Y	Y	Y	Y
		2. Whether environmental factor describes the ratio of uninterrupted hours and present hours.	N	N	N	N	N	N
		3. Whether the team members are distributed over the building?	Y	Y	Y	Y	Y	Y
		4. Whether the team members are distributed over multiple sites?	Y	Y	Y	Y	Y	Y
		5. Facilities Support for work at home, virtual teams, video conferencing with clients.	Y	Y	Y	Y	Y	Y
	4. Software Release constraints (w=4)	1. Whether software release time measured by calendar time?	Y	Y	Y	Y	Y	Y
		2. Whether cumulative test-execution time measured by CPU time?	Y	Y	Y	Y	Y	Y
		3. Whether software-testing cost measured per unit calendar time?	Y	Y	Y	Y	Y	Y
		4. Whether software-testing cost measured per unit execution time?	N	N	N	N	N	N
		5. Whether software fault correction / removal cost measured per unit fault in testing phase?	Y	Y	N	Y	Y	Y
		6. Whether software fault correction / removal cost measured per unit fault in operational phase?	N	N	N	N	N	N
		7. Whether lifetime of software product measured under stated constant?	N	Y	N	N	N	N
	Measure		3.5	3.9	3.1	3.7	4.2	4.4