A Vague Rule Based Analysis Model for Software Risk and Quality Evaluation

Kavita Dahiya
Research Scholar
Computer Science and Application Department
MDU, Rohtak

Dr. Priti
Assistant Professor
Computer Science and Application Department
MDU, Rohtak

Abstract: A software system is defined large number of modules and integrated components. While developing a software system, it is required to analyze the risk factor or the expected quality at the planning stage. At this stage, the requirement and expectation based mapping can be obtained to identify the risk and the quality. These measures directly represent the chances of system failure, cost of project, delay of project etc. In this research, multiple aspect based software quality estimation is provided. These multiple aspects include the entity specific, procedural, resource specific and the object specific. The paper has defined a vague formulated risk evaluation model with multiple associated stages. At the earlier stage, the risk categorization is provided. Later on, the vague based individual aspect risk and the category specific aggregative risk evaluation is provided. In this paper, the risk estimation measure is applied on Cocomo NASA 2 projects datasets. The result shows that the model provided effective evaluation of software projects.

Keywords: Risk Quality, Vague Rule, Software Metrics, Process Metrics

INTRODUCTION

The quality of software system affects from various errors, faults and associated risks. Various software metrics are available to analyze the software system at each process stage of SDLC (Software Development Life Cycle). The metrics are defined to estimate the software system based on different parameters including the performance, reliability, cost, reusability, etc. These parameters can be adapted individually or in combination form to analyze the effectiveness of the software system. The functionality, resource parameters are available to analyze the software system at each stage. In this study, the software metrics defined to analyze the complexity of the software system. Later on the dynamic parameters are applied to evaluate the quality and reliability of software systems. The run time observation with different test methods can be applied. The analysis includes the deep white box testing methods as well as the run time evaluation using black box testing methods. These dynamic methods can be estimated based on the size level estimation. These metrics also available to generate the evaluation regarding the performance, reliability, efficiency and the cost. The time line of software development, required software efforts or manpower can be estimated based on the size level evaluation. The structure and the component density based observations are also considered to analyze the software system.

In the wider form, the software metrics defined to analyze the software system are divided into three main categories shown in figure 1. These metrics can be applied in composite form at different stages of SDLC to perform the evaluation at each level. Each of the categories also having a number of inclusive parameters, constraints and methods. In composite form, these metrics are e able to characterize the software system quality in the adaptive way. The product level, process level and the required resource level evaluation are required to observe the aspect of software systems.

Figure 1. Methods of Software System Evaluation

1.1 Product type metrics:
These are the static measures used to analyze the software size, complexity, testability measures and the existence of common bugs in software systems. The main effect of these metrics is on the performance of software systems. These metrics a iso able to generate the evaluation regarding the portability, efficiency and the cost. The time line of software development, required software efforts or manpower can be estimated based on the size level evaluation. The structure and the component density based observations are also considered to measure the quality of the software system.

1.2 Process type metrics:
The process metrics analyze the software development activities in the communication flow within the software project. The methods, measures and standards are defined to analyze the control behavior and communication flow within the software project. The methods, measures and standards are defined to analyze the control behavior and communication flow within the software project. The methods, measures and standards are defined to analyze the control behavior and communication flow within the software project.
1.3 Resource metrics:

The resource metrics are able to identify the requirement of software systems at various stages, including the development stage, design stage and the maintenance stage. The resource metrics are used to identify effective contribution and planning of software systems at each design stage. The human and non-human resource estimation, the task assignment can be regulated based on these metrics. The software cost and the schedule can be predicted based on the resource metrics. The metrics are able to generate the quantitative decision on requirement and availability measures. The component specific and the aggregative decision can be taken based on these metrics.

In this paper, an analytical observation on various aspects of software quality evaluation is considered with mathematical formulation. The proposed research methodology with algorithmic method is discussed. The results obtained from the proposed research methodology with algorithmic method is discussed. In section II, the work provided by earlier researchers is discussed. In section I, II, the proposed research methodology with algorithmic method is discussed. In section IV, the results obtained from the algorithmic implementation are discussed. In section V, the impact of different phases of software development was also identified.

G. Antoniol et al.[5] has identified the requirement of software evaluation and research with specification of new trends and needs. The opportunistic challenges and the key factor evaluation for software was also formulated by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. Tomaszewski et al.[6] has applied a comparative evaluation on software fault and change identification for th e metric t echnical development and the risk evaluation was provided by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. Islam et al.[7] has defined a risk management and evaluation model for software development project. A combined quantitative risk model was defined by the author. The benchmark requirement of software development and its integration in real environment was identified by the author.

Some other work on software requirements, including image, textual data and videos was provided by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. The benchmark requirement of software development and its integration in real environment was identified by the author. The benchmark requirement of software development and its integration in real environment was identified by the author.
intelligent aspect modeling and the improved software system development was also provided by the author. Layman et al. [15] has defined a work on component driven analysis of development, scheduling a nd the software delivery. The artifact driven estimation and the trend based measurement were also provided by the author.

From this section, it is identified that to improve the software system development, a prior analysis and evaluation stage is required to identify the future risks and to improve the software quality. In this present work, a vague rule based software risk and quality estimation is provided under different integrated aspects. In next sections, the model and the associated experimentation is also described.

VAGUE RULE ADAPTIVE SOFTWARE QUALITY EVALUATION MODEL

In this present work, a more effective and rule based method is provided to estimate the software quality. The quality of the complete system is not based on one factor but it includes more than 10 different factors. These factors are representing the quality of the software system under different perspectives in cluding the entity specific, object specific, process specific or the resource specific. These all quality vectors are here estimated individually and in combination. In c combination, the quality is defined respective to some quality or the risk category. Such as the programmer ability, OS support, technology is the individual quality or risk factors which are combined to represent the platform risk. In same way, six different categories of software quality measures under different aspects are generated. And each category is defined by multiple individual quality or risk factors. To evaluate these risk factors in composite form, a vague rule based method is applied. Vague defines the mathematical rules that combines the attributes based on union, intersection or some other association rules. Finally, the quality of the individual module or software system is evaluated. The work flow of the proposed mathematical rule based software quality evaluation method is shown in figure 2.

![Flow of Work](image)

Figure 2. Flow of Work

Here figure 2 has described the complete work to analyze the software system under different aspects and by apply the value specific mathematical evaluation. In this method, at first the individual attributes are analyzed under the specific formulation. And later on the aspect inclusive attributes are analyzed collectively. Based on the requirement and the observation, the weights are assigned to each aspect as well as each quality feature.

The presented work begins, as the statistics of software project get a available. This dataset is having the descriptive features as well as the quality specific features. To process the proposed algorithmic approach, the quality specific features are separated and presented as the featured dataset. The raw dataset collected is having the nominal values. To perform the value specific rule formulation, it was required to transform the dataset into numerical form. For this, the mapping of each nominal value to specific numerical value is done. This mapping is based on the weight of the relative nominal values such as vh (very high) is assigned by higher numerical value and vl (very low) is assigned by least numerical value. As the numerical dataset is obtained, the next work done is to apply the rule on each individual aspect to generate the specific weight based on the data values of particular attribute.

In second stage of this model, the static analysis on the quality features of software projects is done to identify the relational aspect. At this stage, the aspect specific features are identified and defined as the composite feature set. In this work, such software quality aspects are identified including the product quality, process quality, platform quality, personal quality, Reuse Quality and Schedule Quality. Each of these aspects is further having multiple inclusive features.
The mathematical rule framed algorithmic method is shown below.

**Algorithm 1**

VagueRuledAnalysis(projects)
/*projects is the list of software projects defined with relative characteristics*/
{
  1. for i=1 to projects.length
      [Process the projects]
      {
        2. Pfeatures=projects(i).GetFeatures()
           [Obtain all the project features]
        3. For j=1 to Pfeatures.length
           [Process all the features with nominal values]
           {
             4. tfeatures(j)=ApplyRule(Pfeatures(j))
                [Apply first level rule to transform the nominal values to numerical features]
           }
        5. ScheduleRisk=GetVagueScheduleRisk(tfeatures)
           [Identify the schedule risk from the generated features]
           [Obtain the Vague adaptive product risk]
           [Identify the Vague based Platform Risk]
           [Identify the vague based Personal Risk]
           [Identify the Vague based Process Risk]
        10. ScheduleRisk=VagueAgg(ScheduleRisk)
            [Apply Vague Aggregative for Schedule Risk]
            [Apply Vague Aggregative for Product Risk]
            [Apply Vague aggregative for Platform Risk]
            [Apply Vague Aggregative for Personal Risk]
            [Apply Vague Aggregative for Process Risk]
      }
}

The Algorithm 1 has formulated the method to extract the project features and trained them under various mathematical rules to analyze the quality of a software project. At the first level, the feature driven analysis on each individual rule is applied. Later on, composite evaluation of the software project is done by combining these high level rules. The algorithmic process has provided the product level, process level, personal level, platform level quality of software systems. The implementation of the proposed work model is done on Cocomo NASA 2 projects dataset. The implementation results are discussed in the next section.

**RESULTS AND ANALYSIS**

In this paper, a vague inspired software quality estimator is defined based on various integrated features. The proposed model is implemented in Matlab environment on PROMISE Software engineering repository. It is the publicly available repository which consists of 93 NASA projects designed between 1971 and 1987. The description of this collected dataset is shown in table 1.

<table>
<thead>
<tr>
<th>Table 1: Dataset Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>Dataset Repository</td>
</tr>
<tr>
<td>Source of Data</td>
</tr>
<tr>
<td>Number of Projects</td>
</tr>
<tr>
<td>Year of Development</td>
</tr>
<tr>
<td>Number of Features</td>
</tr>
</tbody>
</table>

Here figure 1 has shown the basic characterization of the dataset considered in this work for the analysis. Each of the projects is defined here with 24 different features. These features include the descriptive features such as the name of the project, project category, development environment, development year, etc. as well as the quality specific features including the storage support, platform support, tool support, etc. The evaluation results based on various quality aspects are presented in this section. Figure 3 is showing the quality estimation based on the process level complexity of software projects.

![Figure 3: Process Complexity Analysis](image_url)
experience of programmer, tool usage experience of programmer, turn a round time on software project and the time constraint associated to the particular project. The white lines are showing the projects with lesser process level quality.

Figure 4.

Figure 4 is shows the individual project evaluation based on personal evaluation. Here x axis represents the software projects and y axis shows the personal associated complexity to represent the criticality in software projects. The personal capability represents the individual capability in terms of language experience, machine experience, modified program practice and etc. Higher the personal quality of software individuals, more reliable the software project can be developed. The white lines are showing the projects with lesser personal level quality.

Figure 5.

Figure 5 is shows the individual project evaluation based on platform capability evaluation on which the software will be developed. Here x axis represents the software projects and y axis shows the platform associated complexity concerns. The platform quality aspect depends on the dead line specification, user system memory, DB size and etc. The higher the developing environment and the user environment are the same, the higher risk in software delivery and more chances of its software maintenance. Such system can be considered with lesser deliverable quality features.

CONCLUSION

In this paper, a vague inspired method is defined for software risk evaluation. The proposed work model first divided the available software features in relative categories. The method also evaluated the individual feature under mathematical rules as well as aggregative category inclusive features. The work model is applied on Nasa 2 projects dataset. The results are obtained in terms of different risk categorization for all available projects.

REFERENCES

[1] Reham Ejaz, "A Quality Assurance Model for Analysis Phase", NSEC’10, 04-OCT-2010, Rawalpindi, Pakistan ACM 978-1-4503-0026-1/10/10


