Abstract—Software Quality means the customer satisfaction according to its requirements and from the site of producer (software developer) it is only dependent upon the profit. Profit can be defined as money and only money. There are so many methods, models and formulas to analyze the quality of software for both producer and consumer and these methods are very good to analyze the products. These methods and models are also very helpful for cost estimation and also to analyze the different quality attributes like reliability, reusability, security etc. Hybrid Re-engineering is one of the technologies to make quality product. This technology works on legacy systems, according to Hybrid Re-engineering a developer can produce software by merging three different techniques. This paper is proposing a new model for quality of Hybrid-re-engineered product by using some matrices for all the different tracks of Hybrid Re-engineering to make effective Hybrid Re-engineering process.

Keywords — commercial –off the self component, software quality, quality attributes, hybrid re-engineering

1. INTRODUCTION

Re-engineering is the examination, analysis and alteration of a legacy software system to reconstitute it in a new form, and the successive implementation of the new form. The process classically encompasses a combination of other processes such as reverse engineering, re-documentation, re-structuring, translation, and forward engineering. The objective is to understand the legacy software (all the phases of software engineering process) and then to re-implement it, to improve system’s functionality and performance. The objective is to maintain the existing functionality and prepare for functionality to be added later to make a new software for the future use. But there is a problem, that is the system currently in use are, “legacy” systems, these have lacking in a good design structure and code organization, making changes to the software is difficult and costly (Schneidewind., 1997). Corporations do not want to “refuse” their systems because they are many built in fine business application processes that have evolved over time that would be lost. Often the developers of the legacy systems are not available to validate or describe this information; the only resource is the current software code (Hamada, 2008). The original cost of developing the logic and components of the software system should not be wasted, so reuse through re-engineering is desired. The challenge in software re-engineering is to take active systems and instill good software development methods and properties, generating a new target system that maintains the required functionality while applying new technologies (Khoshgoftaar., 2006). While explicit objectives of a re-engineering task are determined by the goals of the corporations, there are four general re-engineering objectives and these are:

- Improve Functionality
- Improve Maintainability
- Improve Portability
- Improve Reliability

Legacy systems, throughout the years of modifications due to errors or enhancements, become complicated and expensive to change. The code no longer has an understandable logical structure and documentation may not exist, and if it exists, it is often outdated. Re engineering specifies the characteristics of the legacy system as shown in Figure 1 that
can be compared to the qualifications of the characteristics of the required system (Tahvildari, 2004). But when a developer talk about the quality then there will be some technique is hybrid re-engineering. It is the combination of three different parameters and these are 1) COTS, 2) Translation 3) Modification of code as shown in Figure 2.

II. HYBRID RE-ENGINEERING

In Hybrid Re-engineering, legacy systems are re-engineered using the approach show in Figure 2. And adaptation of the general model for software re-engineering showed in Figure 1. In Figure 2, three development tracks are utilized. The first track is a translation from existing code to a new language, operating system or hardware platform with no abstraction (Karygiannis, 2006). The second track uses the existing code to identify requirements that can be satisfied by the application of COTS packages. Last track is the regular re-engineering process, the development of new code for requirements that cannot be satisfied by either of the other tracks, and to “glue” together the translated and COTS components. Re-engineering as a development methodology has inherent risks, such as schedule, functionality, cost and quality. Hybrid Re-engineering was developed to decrees some of these risks since COTS packages are expected to have high Reliability and require minimal development time (Tahvildari, 2002). No other method of decrease time and cost through Hybrid Re-engineering while maintaining functionality is through a straight translation of part of the current code to the new language of operating system. Hybrid Re-engineering is innovative, combining three distinct re-engineering efforts, hence the risk generally associated with re-engineering can increase by combining the risk inherent o each track.

Since Hybrid Re-engineering is combining products from different development tracks (COTS, Custom Software and Translated Software) one new risk is the interface and interoperability of the products (Ajlouni, 2006). Like-data transfer between products can cause compatibility and timing problems; COTS packages may not work exactly as anticipated. In general metrics can be used by management to improve software efforts and minimize the risk. Metrics can indicate how well a project is meeting its goal (Yongqiang, 2007). In Hybrid Re-engineering, metrics can support the justification for decision on track selection for different software function and components.

III. PROPOSED MODEL

Proposed Model is about the quality with the hybrid re-engineering. In this model there are some metrics for the different tracks of hybrid re-engineering. Metrices are defined according to the tracks through which a developer could make quality software and with satisfying these matrices developer could make quality software by using Hybrid Re-engineering (Alsultanny, 2009). All the matrices are defined according to the tracks of Hybrid Re-engineering as shown in Figure 3.

Metrics for Translation, It is very necessary to find the complexity of translated program and previous program to make a comparison between them. Second metrics will be Maintainability because if we are translating a program from old technology to new one then it would be essential for developer to maintain it. Both the matrices are defined with their different parameters.

Complexity- Complexity of any software is defined as the degree of logic used in it. Software complexity is a combination of two different types of complexities and this are:
- Cyclomatic Complexity
- Essential complexity

We can define Cyclomatic complexity V (G) as it is the measure of the amount of logic in a code module of third and fourth generation language. If V (G) successively high then it leads to impenetrable code i.e., a code which is at higher risk due to difficulty in testing. Essential complexity can be defined as it is a measure of the degree to which a code module contains unstructured constructs.

EDM= Essential complexity/Cyclomatic complexity

We are showing the three kinds of complexity metrics at four different levels of analysis with the help of Table 1. This classification yields different independent metrics.
Table metrics state that-
- The average size of the programs which constitute the system.
- The average size of control-structure complexity which has been normalized for size.
- The average size of the within each program.
- Maintainability, is the ease with which product can be maintained in order to-

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Statement</th>
<th>Sub-Program</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Lines</td>
<td>Effort</td>
<td></td>
</tr>
<tr>
<td>Decision Structure</td>
<td>Density of Decisions</td>
<td>Cyclomatic Complexity</td>
<td>Number of Binary decision</td>
</tr>
<tr>
<td>Control Structure</td>
<td>Density of Branching</td>
<td>Essential Complexity</td>
<td>Number of Branch</td>
</tr>
</tbody>
</table>

### Isolate defects
Product can be maintained in order to correct defects
In order to meet new requirements
- In order to make future maintenance easy
- In order to cope with changed environment.

According to the model developer will use the testing strategy for the maintenance of software. Testing strategy means the V&V techniques. Veriﬁcation and Validation techniques contain the reports, reviews and regression testing through which developer can make effective results to maintain a software with low cost.

**Metrics for COTS**, COTS components are reusable components and these can be reuse again and again for the development of different software’s. Their will some metrics to measure the quality of COTS integration (Interfacing), through which developer can produce a quality software according to the requirements of the customer (Hyatt, 1996). The following metrics should be measure to make it efﬁcient, and these metrics are-

**REUSABILITY: SUB METRICS ARE**-
- LCOM:- Lack of cohesion in method
- CBO:- Coupling between object class
- NOC:- Number of children
- CPD:- Component Packing Density
- IDC:- Interaction Density of Component
- IIDC:- Incoming Interaction Density of a Component
- OIDC:-Outgoing Interaction Density of Component
- AID:-Average Interaction Density

**Complexity: Sub metrics are related to the component complexity**

![Software Quality Model for Hybrid Re-engineering](image)
Method Complexity = Cyclomatic complexity + Number of calls to other methods
\[ MCOM = CC + NCOM \]

Component Complexity = Number of Interfaces of a component + Number of couplings to other components
\[ CCOM = NIMC + NOIC + NCOC \]

Emergent System Complexity = Number of Links between components + Depth of the composition tree
\[ ESCOM = NLBC + DOCT \]

Testability: Sub metrics are-
- NOC (Number of children)
- RCC (Rate of Component’s Customizability)

CRIT-bridge
CRIT-link

Metrics for Customization
Customization of code in Re-engineering means making the code from scratch to get the requirement. There are some quality attributes which would be help full to get a quality software with reverse engineering and forward engineering and these attributes are-
- Reliability
- Maintainability
- Usability
- Efficiency
- Functionality
- Portability

IV. CONCLUSION
Proposed model would be helpful to make the quality hybrid software. This model will help to make quality hybrid re-engineered software according to the customer requirements and also according to the developer requirement (profit), with this model developer could be able to make all the tracks of hybrid re-engineering satisfied according to the requirements of customer and developer will also be able to make more profit.

REFERENCES