CHAPTER 5

Component-Based Software Reliability Models

In this chapter the brief outlay of the component based reliability models has been discussed as the reliability analysis cannot be improve without measuring the models and their techniques. Here we tend to discuss the various forms of component base models and according to Goseva et al., software reliability is one of the most important metrics of Software Quality Assurance. A product unwavering quality model depicts issue presentation what's more evacuation and the operational environment that influences the disappointment process. In request to gauge the unwavering quality of segment based programming structural planning, we need to utilize part based programming unwavering quality models. Part based frameworks contrast from conventional frameworks in the way that the framework is isolated into discrete legitimate units that are incorporated together specifically programming construction modeling. These units may comprise of business off-the-rack programming items or parts of past projects being reused. To gauge the unwavering quality it is important to utilize a procedure that contemplates the distinct parts and the structure of the framework. Segment based programming dependability models have the ability to foresee framework unwavering quality dependent upon the unwavering quality of each singular part and the operational profile of the given requisition. In the following three segments, distinctive classifications of segment based programming unwavering quality models, as recommended by Goseva et al. are talked about.

5.1 Component-Based Software Expansion

New programming advancement orders are rising to address the issues copartnered with part programming frameworks. Specifically, part based advancement (CBD) and part based programming designing (CBSE) have emerged to give a deliberate methodology at the dissection and development of programming frameworks by gathering pre-assembled parts. Around the focal points of CBD is the capability to quickly build and send programming frameworks which have a high level of intricacy. By obtaining and incorporating programming segments from diverse sellers, a product designer can quickly develop a
completely working programming framework. On the off chance that the product parts used to assemble the framework have been checked to be practically right and exact, then the general framework ought to have a comparative level of value, gave that the segments were coordinated effectively. CBD likewise permits programming engineers to substitute new parts into a given structural planning in order to meet different nonfunctional necessities (for instance, memory utilization). Preferably, parts might be substituted for others that backing an indistinguishable interface and perfect connotation. This permits the improvement methodology to rapidly assess the benefits of distinctive parts in the setting of an existing structural planning. There are numerous variables, in any case, that are keeping down segment based software development. For instance, the absence of a reasonable segment business constrains the number of parts freely accessible for reuse. As the segment business sector develops, be that as it may, the amount of parts and in addition the areas over which the segments work will extend, making segment based advancement a more reasonable alternative in the advancement of complex programming frameworks. One of the explanations behind the predetermined number of parts accessible for procurement is the high level of trouble in processing a segment which is both usable and reusable. In place for a part to be usable, the client of the segment must be equipped to coordinate the part effectively into an existing building design. In view of this, a part's interface ought to be moderately straightforward and straightforward. Then again, in place for a part to be reusable, the fashioner of the segment must make the part as nonexclusive and as adaptable as would be prudent in order to permit the segment to work in a wide assortment of situations. Accordingly, a bland segment will normally oblige a more confused interface. A more entangled interface, while pushing reusability, hinders convenience. Characteristically, a harmony between reusability and convenience must be attained. Throughout segment advancement, mind must be taken to guarantee that segments are both dependable and impervious to change. The part engineer should likewise be clear in reporting the part's imperatives and prerequisites. Since a accumulation of parts could be conveyed incrementally, the earth of a part may be always showing signs of change. Accordingly, segments must be intended to be impervious to relevant change. Regarding the incremental segment organization specified in the past segment, segments that are delicate in their arrangement environment are more helpless to unwavering quality issues as nearby components change around them. Part forming and reliance methodologies can help guarantee between part similarity.
5.2 Current Component Models

In industry, parts were initially acquainted with handle the development of usually utilized graphical client interface elements. On the other hand, as the segment scene developed, parts have ended up more adaptable to handle more general issue spaces. Segment shows basically give the establishment whereupon part arrangement and correspondence occur. Segment models give the framework through which segments can distinguish one another and thusly collaborate with each one in turn. This area highlights a percentage of the segment models predominant in the business.

5.2.1 Common Object Request Broker Architecture

CORBA is a standard set forth by the Object Management Group (OMG) Accordingly, the standard is stage and merchant nonpartisan. CORBA essentially considers dispersed articles to find and communicate with each other over an Object Request Specialist (ORB). Technique contentions are marshaled at the customer end and transmitted over the ORB through a decently characterized convention, normally the Internet-Interorb Protocol (IIOP). They are consequently unmarshalled at the server end, the technique is conjured what's more any return qualities are also transmitted once again to the customer. Keeping in mind the end goal to find objects, CORBA characterizes the Naming Service which permits objects to be placed by name. The naming administration is a piece of the Corbaservices bundle which likewise gives backing to framework level administrations, for example, ingenuity, occasions, transactions and database questions. Larger amount reflections and develops are characterized by Corbafacilities system which delivers issues identified with both the flat and vertical requisition markets. One of the qualities of CORBA is the way that it backs various dialects through the utilization of an Interface Definition Language (IDL). This dialect permits the designer to characterize the technique marks and article pecking order of all the appropriated protests in a framework. An interpreter is then used to guide IDL to a routine dialect, commonly C++ or Java. Consequently, libraries of items composed in diverse dialects are fit to connect with each other. Strictly talking, in light of the fact that CORBA just gives an item turned methodology to the accepted Remote Procedure Call (RPC), it could be contended that CORBA does not adjust to the routine meaning of part as displayed previously. Endeavors to amend this have started as of late with the presentation of the CORBA Component Model (CCM) by the OMG.

5.2.2 J2EE / Enterprise JavaBeans
Undertaking Javabeans (EJB) from Sun Microsystems is a later improvement in the part demonstrate industry. This segment model, which is some piece of the J2ee structure, offers a generally develop stage for segment sending and connection. Numerous qualities from CORBA have been obtained and upgraded by EJB including the idea of a naming administration and the correspondence convention utilized by EJB segments to correspond with each other (IIOP). The J2ee schema gives backing to the 3-level structural engineering in which customers (level one) impart in a roundabout way with the Ejbs living on a server (level two). The Ejbs then interface with backend databases (level three) keeping in mind the end goal to fulfill the customers' appeal. Ejbs live in the setting of a compartment on the server, accordingly all correspondence to the EJB must occur through a remote interceptor object which interfaces the customer with the EJB. The holder chooses the life cycle of every last one of Ejbs under its area and can instantiate more EJB segments as needed. This helps the versatility of the J2ee structural engineering. The holder can likewise deal with different obligations for example, steadiness and security, thusly permitting the EJB engineer to think exclusively on the purpose of the segment without being occupied with subordinate assignments. This partition of obligation between the EJB and the holder takes into account the development of a more strong structural engineering. Parameterization of EJB segments is made conceivable through an organization descriptor. This XML record is set on the server as a feature of the organization of the segment and offers an approach to change the conduct of a framework without needing to recompile its constituent parts. Sadly, EJB, by definition, is dialect subordinate. Notwithstanding, in light of the fact that the EJB particular has received the IIOP remote correspondence convention, it is conceivable for EJB to speak with other CORBA questions on a system. The J2ee structural planning additionally has the playing point of being seller nonpartisan as the detail handled by Sun may be actualized by different sellers. For sure, different sellers, for example, IBM, JONA, and BEA Systems have actualized their own particular adaptations of the J2ee building design notwithstanding Sun. Jboss, an unreservedly accessible, open-source usage of the J2ee construction modeling is likewise accessible.

5.2.3 Dot Net

Lately, Microsoft has proposed the purported .Net schema. Thought about to different offerings, it is moderately juvenile and is expected to be stage impartial. Cases of dialect nonpartisanship have been bargained by the advancement of Microsoftcontrolled dialects, for
example, Visual Basic and C# as the support for the .Net schema. Regardless of the level of univendor control, a few endeavors have been made to copy this system outside the limits the Microsoft stage. As of late, a few endeavors have likewise been made to make the stage managable to merchant impartial dialects. Such deliberations may make this structural planning deserving of further study later on

5.3 State-Based Models

Chains, Continuous Time Markov Chains, or Semi-Markov Process to describe the provision building design. These models assume part and interface disappointment rates that can change with time. State-based models might be further ordered as either composite or progressive. Composite state-based models utilize the given design model and disappointment conduct to foresee dependability. The progressive approach first comprehends for the structural model and afterward utilizes the settled compositional model alongside given disappointment conduct to anticipate unwavering quality. A short portrayal of the Littlewood model, and the Laprie model, which are few of the statebased models, is given cry. The Littlewood model is an about general building design based programming dependability model.

The Laprie show additionally expects that the framework is made up of limited amounts of segments. Exchange of control between the distinct segments is depicted through the utilization of a persistent time Markov chain. What's more, the time used inside each segment is spoken to by \( \lambda_i \). The disappointment conduct in the Laprie model accepts that the parts fall flat with a steady disappointment rate \( \lambda_i \), the disappointment rates are much more modest than the execution rates and that the outcomes in the trade of control happen before the framework disappointment.
The Littlewood model utilizes an irreducible semi-Markov methodology to model the building design of the segment based framework. This model expects that the product framework comprises of a limited amount of elements and the exchange of control between modules could be specified by a likelihood $p_{ij}$, where $p_{ij} = \text{Pr} \{\text{program travel from module } I \text{ to module } j\}$ [8]. In expansion, the time used in a module could be clarified as a general dispersion capacity $F_{ij}(t)$ with a limited mean of $m_{ij}$ [8]. There are two sorts of discontent conduct in the Littlewood model: disappointment throughout execution of the module spoke to as $\lambda_i$, and disappointment throughout exchange of control between two modules spoke to as $v_{ij}$. By joining together this information, the disappointment rate as stated by the Littlewood model is given by the accompanying.

5.3.1 Path-Based Models

The way based models in their dissection have comparable steps as the state-based models watching the different execution ways and their frequencies that specific system can exercise. These replicas may entail an executable program and comprehensive font code for testing to make essential way data. The way data is further joined with the disappointment conduct to foresee unwavering quality. Two illustrations of way based models are the Shooman model [8], and Yacoub et al., [21] model. The Shooman model is one of the first models used to gauge the unwavering quality of segment based frameworks utilizing a way based methodology. This model accepts that the execution ways of the framework are known alongside the frequencies of event for every way $i$, meant by $f_i$. This model describes disappointment conduct by computing the disappointment likelihood on every way $i$, meant by $q_i$. This data, further, is joined together to produce the framework likelihood of disappointment on any test run

$$q_o = \sum_{i=1}^{n} f_i \cdot q_i \text{ where } n \text{ is the number of components in the system [8].}$$

The Yacoub et al model [21] takes into consideration dependability reckoning in the early phases of improvement before an executable of the framework is accessible. It utilizes a dissection strategy that is built strictly in light of execution situations. A situation is a situated of collaborations between parts executed by definite data, and it is identified with the idea of operations and run sorts utilized within operational profiles. To characterize the situations, Yacoub et. al. model utilization arrangement graphs like these utilized within the Unified
Modeling Dialect (UML). These graphs give the essential data to compute the normal execution time of a segment in a situation, the normal execution time of a situation, and conceivable cooperations around parts [21]. Utilizing situations, a probabilistic model called Component Dependency Graph (CDG) must be built. The CDG is an adjusted control stream chart that has been changed to apply to component-based provisions. An examination calculation portrayed in [21] is then connected to the CDG to acquire dependability results.

5.3.2 Additive Models

Added substance models are kept tabs on evaluating the general provision unwavering quality by consolidating the dependability information of the distinctive segments. These sorts of models accept that the parts unwavering quality could be demonstrated by a non-homogeneous Poisson methodology, considering the framework unwavering quality to be communicated as the entirety of its parts unwavering quality [8]. As added substance model samples, there is the Xie-Wohlin, and the Everett models. The Xie and Wohlin model assumes that every segment is a solitary framework joined in arrangement to transform the aggregate framework. Utilizing the theory that disappointment of a segment will absolutely bring about disappointment of the whole framework, the Xie and Wohlin model looks at the singular part disappointment power, indicated by \( \lambda_i(t) \), to acquire the framework disappointment force at time \( t \) as:

\[
\lambda(t) = \sum_{i=1}^{n} \lambda_i(t) \text{ where } n \text{ is the number of components in the system}
\]

The Everett model addresses the issue of assessing singular part's unwavering quality. Dependability of every segment is investigated utilizing the Extended Execution Ti(EET) model. The EET model utilizes two wellsprings of constraints that could be dead set straightforwardly from the properties of the product and from the data on how experiments furthermore operational use stretch every part. The joined together model for the framework overlay segments reliabilities. The point when the underlying EET model for the segments are non-homogenous Poisson procedure shows, the combined number of disappointments for such a model is the whole of the comparing capacities for every part.

5.4 Component-Based Reliability Applications
There are three instruments as of now being produced that will give backing to displaying the dependability of segment based frameworks. Not at all like the models examined in the past three segments, these devices can address dependability forecast at a prior phase of improvement. These provisions are ECRA, PECT, and SREPT. All of them are the effects of tasks in the scholarly world. Early Component Reliability Assessment device or ECRA is a homegrown apparatus manufactured at the High Assurance Systems Research Center of West Virginia University. The ECRA device varies from most conventional programming unwavering quality models by its capability to be utilized at an early phase of project improvement. Through the utilization of framework structural planning and operational profiles, it can evaluate the dependability of the discharged framework before code advancement starts. This apparatus permits immediate communication with Rational Rose, a case instrument frequently utilized as a part of the early phases of advancement. Expectation Enabled Component Technology or PECT is device created as a feature of the Pacc2 venture at the Software Engineering Institute. The motivation behind PECT is to create an extra for an alternate instrument called Comtek. Comtek is a nature's turf, in which clients can make and execute part based projects. PECT interfaces with Comtek to get the construction modeling of the product framework, and after that take a way based methodology to figure the normal commitment of every part to the framework. Contingent upon the learning about the parts and framework reliabilities accessible, PECT can compute framework dependability or unwavering quality of the distinctive parts. With learning of the framework dependability PECT can ascertain the singular part reliabilities by proportioning the framework unwavering quality with the normal commitment of each segment. With the information of unique parts reliabilities, PECT can gauge the aggregate framework unwavering quality by including the corresponding dependability of each segment concerning the normal commitment of every part to the framework. Programming Reliability Estimation and Prediction Tool or SREPT is a provision created at Duke University. Its essential objective is to "track the nature of a product item throughout the product life-cycle, right from the design stage the distance up to the operational period of the product". As being what is indicated, SREPT is a product dependability development instrument, yet its models are intended to record for part based frameworks. SREPT utilizes a state-based approach and gives numerous strategies to model the programming framework. SREPT executes its model of the framework to ascertain the time used in furthermore the normal number of visits to every part. It then partners the structural planning of the framework with the client characterized disappointment probabilities for every part to prepare the general framework dependability.
5.5 Fault Promulgation in Software

It is realized that a product shortcoming and the ensuing lapse in one segment could be spread to other communicating segments bringing on their disappointments. Part disappointments are, accordingly, rarely free as it is typically accepted by existing part dependability models. Figuring mistake engendering is in this way essential from the designer's point of view, since these assessments might be utilized to discover the most influenced parts, and take activities to enhance the unwavering quality of the framework by applying failure discovery or slip recuperation systems. There have been a few studies on assessing and investigating the slip proliferation between programming parts or modules. Slip proliferation models have been approved through issue infusion examinations, emulated by the examination of the genuine failure engendering effects with the ones anticipated by the models. In this theory, we embrace the failure proliferation model of Nassar et al.. This model rose in an extend that examined extensive variety of compositional characteristics, for example, change engendering likelihood, and necessities spread likelihood. One of the explanations for choosing Nassar's model is its dependence on UML plan antiques, the documentation that this exploration utilizes for the dependability demonstrating as well. It is vital to push that different procedures for ascertaining the mistake engendering likelihood (as will be depicted in Section 3) could be used with this dependability model.

5.5.1 Nassar et al. procedure for scheming error proliferation

In [5.4], slip engendering is ascertained from the data accessible at a young hour in the lifecycle, in the framework outline stage. The data about the structure and connotation of the code is not accessible at this stage, However the data about the stream of control and information inside segments and between parts exists in suitable UML curios. Nassar et al. characterize the slip proliferation likelihood (EP) from part A to segment B as takes after:

\[
EP(A,B) = P([B](x) \neq [b](x')) | x \neq x'
\]

where \([B]\) is the capacity of segment B which catches all the conclusions of implementing B (the state of B and the yields of B), \(x\) is a message example utilized as a part of the correspondence between segments An and B, \(x'\) speaks to defiled message, and \(EP(A,B)\) speaks to the likelihood that an issue and partnered slip state in A will be spread to B. In different words, the conclusion of the execution of B having gained the defiled message \(x'\)
will be not quite the same as on the off chance that it had gained the first message x, as aftereffect of the mistake that happened in A bringing on the message debase ment. Having the construction modeling of N parts, EP is a N x N lattice, where the section in column An and section B is the failure proliferation likelihood from segment A to part B. EP(A, A) is equivalent to 1, implying that failure in the given part will dependable transform its needed result. On the off chance that a segment executes some mistake redressing components, this supposition might need to be changed. EP(A, B) is a restrictive likelihood on the grounds that the likelihood that a slip engenders from A to B is computed under the condition that part A really transmits a message to segment B. The following equation communicates the failure proliferation likelihood between A and B inferred from the probabilities of messages being sent from A to B and their state information:

\[
EP(A \rightarrow B) = \frac{1 - \sum_{x \in A} P_x(x) \sum_{y \in B} [P_{x \rightarrow y}(y)]^2}{1 - \sum_{z \in A \rightarrow B} [F_z^{-1}(y)]^2}
\]

The creators of [3] further characterize Unconditional Error Proliferation, which we discover especially convenient for unwavering quality displaying. Meant by E(A, B), it is characterized as the likelihood that a mistake proliferates from A to B, without being molded by a genuine event of a message from A to B. E(A, B) is computed from the Transmission Likelihood Matrix, T(A, B). Every section in T(A, B) shows the likelihood of the connector from A to B being initiated throughout an authoritative execution. The object for network T is "to reflect the change in recurrence of actuations of diverse connectors throughout a common execution". The Unconditional Error Proliferation is acquired from

\[
E(A, B) = EP(A \rightarrow B) \times T(A, B).
\]

T(A, B) is computed from the amount of messages between segments An and B in the given UML framework model partitioned by the amount of all watched messages in the framework. As such, T(A, B) speaks to an assessment of the likelihood that a message is sent from A to B.

5.5.2 Hiller et al. approach for scrutinising the Proliferation of DataErrors in software
Hiller et al. present an alternate methodology for breaking down the proliferation of information slips in programming. They present the idea of blunder porousness as a fundamental measure upon which they characterize a set of related measures. These measures are utilized to break down the powerlessness of programming and to discover the modules that are undoubtedly laid open to engendering mistakes. Taking into account the performed slip porousness investigation and the related measures, Hiller et al. depict how to select suitable areas for mistake location instrument (EDM) and mistake recuperation system (ERM). In their study, they think about particular programming which has summed up modules as secret elements having different inputs what's more yields. Case in point, for a specific module M with m inputs, numbered from 1 to m, and n yields, numbered from 1 to n, they characterize mistake penetrability as the restrictivelikelihood of a mistake happening on the yield given that there is a blunder on the input. Accordingly, for input I and yield K of a module M, they characterize mistake porousness, M i k P, as follows:

\[ 0 \leq P_{M, i, k} = Pr\{\text{error in output error in input i}\} \leq 1 \]

This measure shows how penetrable an information i/output k pair of a product module is to lapses happening in the inputs. In the event that the slip porousness of an info i/output k pair is zero, this does not imply that an approaching failure completed not result in any harm in light of the fact that the same failure may have created some concealed blunder in the inward state of the module and for some reason is not obvious on the yields. The failure porousness is the fundamental measure whereupon it is conceivable to characterize a set of related measures. Correspondingly, Hiller et al. characterize the relative penetrability, PM, for input i and yield k of a module M as:

\[ 0 \leq P_M = \frac{1}{M} \cdot \frac{1}{n} \sum_i \sum_k P_{M, i, k} \leq 1 \]

where m is number of inputs, and n is number of yields. This articulation does not naturally reflect the general likelihood that a slip is pervaded from the input of the module to the yield. Rather, it is a conceptual measure that could be utilized to get a relative requesting over the modules. To recognize modules with a vast number of information and yield signs from those with a little number of data and yield indicators, the creators evacuated the weighting component in the past mathematical statement, and characterize the non-weighted relative porousness, PM, for information i and yield k for module M as follows.
These measures, characterized in the last two comparisons, are utilized by the creators to break down the modules of a given framework. When the qualities for the lapse penetrability for each one input/output pair are acquired, it is conceivable to develop a penetrability diagram. Every hub in the diagram compares to a specific module and has various approaching and friendly curves. Each one curve has a weight connected with it which is the blunder penetrability esteem. With blunder penetrability charts it is conceivable to perform two separate sorts of spread dissection: · Backtrack from framework yield to discover ways with most elevated likelihood of blunder spread (Output Error Tracing). · Trace failures from framework data to discover ways these blunders will probably spread along (Input Error Tracing). Yield failure following is refined by building a set of backtrack trees, one for each framework yield. The root in backtrack tree speaks to the framework yield, the halfway hubs speak to interior yields, and the leaves speak to framework inputs. All vertices in backtrack tree have a weight comparing to a blunder penetrability esteem. Once the backtrack tree is acquired, discovering the proliferation ways with the most elevated engendering likelihood is a matter of discovering which ways from the root to the leaves have the most noteworthy weight. Info Error Tracing is performed in comparative way. We need to build a set of follow tree for every framework information. In a follow tree, the root speaks to a framework enter, the leaves speak to framework yields, and the middle of the road hubs speak to inner yields. From the follow trees it is conceivable to discover the engendering pathways that failures on framework inputs might in all likelihood take by discovering the ways from the root to the leaves having themost noteworthy weights. Utilization of these trees empowers deciding two viewpoints: 1) the ways in the framework along which lapses will no doubt proliferate to certain yield indicators, and 2) which yield indicators are in all likelihood influenced with blunders happening in the information signs. So as to establish the modules that are destined to be presented to engendered lapses along the got ways with most possible spread, the writers characterize the mistake introduction measure, \( X^M \), of a specific module \( M \) as

\[
X^M = \frac{1}{N} \sum \text{weight of all incomings paths of } M
\]
where \( N \) is the amount of approaching ways and \( M \) is the hub in the porousness diagram, speaking to programming module \( M \). The weight for every way is the result of the mistake porousness values along the way. The mistake presentation is the mean of the weights of all approaching circular segments of a hub and is limited by \( N \). Practically equivalent to the non-weight relative porousness, the creators characterize the non-weighted mistake presentation as

\[
X^M = \sum \text{weight of all incomings paths of } M
\]

The two introduction measures are utilized for recognizing hubs with a little number of approaching ways and those with a substantial number of approaching ways. At long last, ith these two sets of measures; the two introduction measures, and additionally the two long ago characterized penetrability measures, the creators were ready to give the accompanying principles for picking Edms and Erds areas: · The higher the mistake presentation qualities of a module, the higher the likelihood that it will be subjected to mistakes proliferating through the framework if blunders are available. They presumed that it may be more powerful to place Edms in the modules with higher lapse presentation than in those with more level blunder introduction. · The higher the lapse porousness qualities of a module, the higher the likelihood of subsequent modules being subjected to spreading blunders if failures ought to pass through the module. Thusly, the creators recommended it is practical to plaErds in the modules with higher failure penetrability values than in those with easier blunder penetrability values.

### 5.6 Primary Reliability Estimate for Component-Based Systems

As officially said, a few procedures exist for unwavering quality investigation of part based programming frameworks. Be that as it may, just a couple of might be connected in promptly phases of framework improvement, i.e., before an executable form of the whole framework is accessible. The model of Sing et. al. has the accompanying attributes:

1. Relevance promptly in the product advancement lifecycle.

2. Consistent mix with UML outlines.
3. Dependability forecast in the framework configuration stage and unwavering quality evaluation dependent upon the watched disappointment conduc. Singh's dependability model is dependent upon a few suppositions:

- Existence of data about disappointment rates for parts and connectors in an building design.
- Independence of the disappointments around distinctive segments.
- Component disappointments take after the standard of consistency, i.e., a part is relied upon to display the same discontent rate at whatever point it is summon.

5.7 Conclusion

As the above dialog of segment models exhibits, there is no predictable methodology to demonstrating segments and their cooperations. There are a few contending approaches each with their own particular favorable circumstances and hindrances. CORBA offers the most develop innovation; however J2ee has had the capacity to receive a large portion of the more fruitful ideas began by CORBA to make a suitable server-side segment model. Of specific imperativeness is the way that none of the down to earth segment advancement models depicted in this section gives a feasible method for deciding similarity between programming segments. Separated from the exact constrained static checking of parameter and return sorts, none of the models makes any endeavor to secure the element consistency between segments that must be available in place for segments to effectively associate with each other.

In this Chapter we intend to analyse the component based reliability models which further shows the degree of reliability on the basis of different models and pattern. It is further analysed that the component reliability has a great impact on the overall system of software reliability and improving the reliability efforts. We also make sure for the some efforts to make the prediction of the reliability efforts for the improving the reliability of the component based. As when we reduce the error proliferation from the component then we can also realize the extension of the software. Early models have some limitations which motivate us to design and modularize the extension of the some previous models. In this regard we extended the Nisaar’s Model for the reliability effort for the component based.