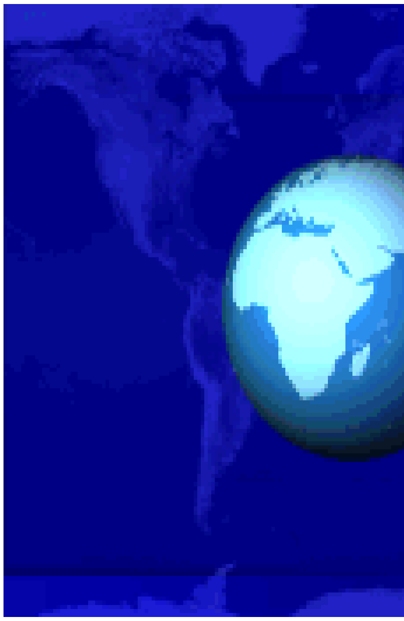


Software Reliability

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Basic Concepts

There are three phases in the life of any hardware con
burn-in, useful life & wear-out.

In **burn-in phase,** failure rate is quite high initially, a
decreasing gradually as the time progresses.

During **useful life period**, failure rate is approximately co

Failure rate increase in **wear-out phase** due to wearing
components. The best period is useful life period. The s
curve is like a “bath tub” and that is why it is known
curve. The “bath tub curve” is given in Fig.7.1.

Software Reliability

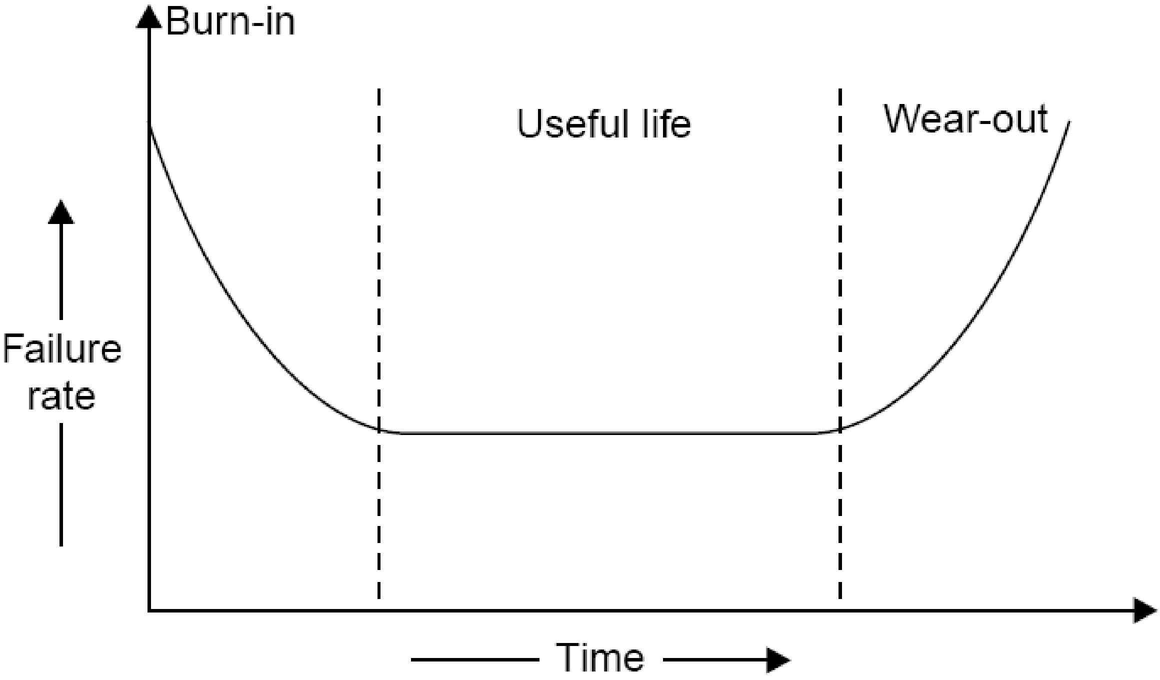


Fig. 7.1: Bath tub curve of hardware reliability.

Software Reliability

We do not have wear out phase in software. The expect
software is given in fig. 7.2.

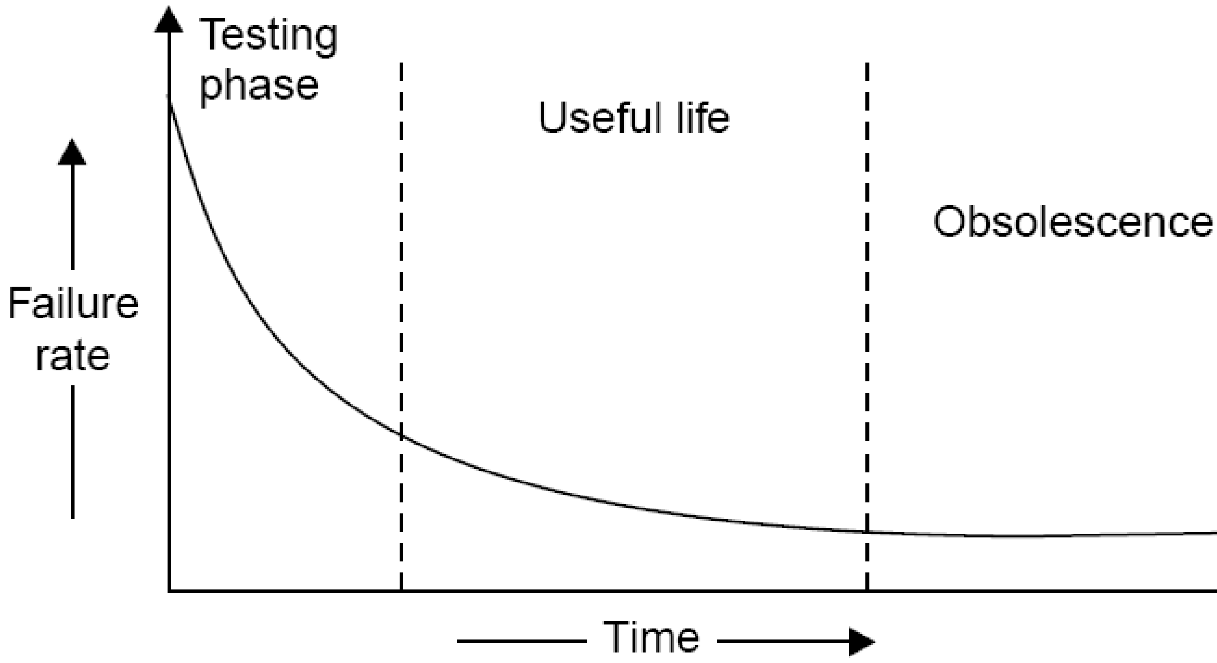


Fig. 7.2: Software reliability curve (failure rate versus

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

Software may be retired only if it becomes obsolet contributing factors are given below:

- ✓ change in environment
- ✓ change in infrastructure/technology
- ✓ major change in requirements
- ✓ increase in complexity
- ✓ extremely difficult to maintain
- ✓ deterioration in structure of the code
- ✓ slow execution speed
- ✓ poor graphical user interfaces

What is Software Reliability?

"Software reliability means operational reliability. Who many bugs are in the program?

As per IEEE standard: "Software reliability is defined as a system or component to perform its required fund stated conditions for a specified period of time".

Software Reliability

Software reliability is also defined as the probability tha system fulfills its assigned task in a given environ predefined number of input cases, assuming that the ha the inputs are free of error.

"It is the probability of a failure free operation of a pr specified time in a specified environment".

Software Reliability

- Failures and Faults

A fault is the defect in the program that, when exec
particular conditions, causes a failure.

The execution time for a program is the time that is actua
a processor in executing the instructions of that pr
second kind of time is calendar time. It is the familiar t
normally experience.

Software Reliability

There are four general ways of characterising failure oc
time:

1. time of failure,
2. time interval between failures,
3. cumulative failure experienced up to a given time,
4. failures experienced in a time interval.

Software Reliability

Failure Number	Failure Time (sec)	Failure interval (se
1	8	8
2	18	10
3	25	7
4	36	11
5	45	9
6	57	12
7	71	14
8	86	15
9	104	18
10	124	20
11	143	19
12	169	26
13	197	28
14	222	25
15	250	28

Table 7.1: Time based failure specification

Software Reliability

Time (sec)	Cumulative Failures	Failure in interval (30
30	3	3
60	6	3
90	8	2
120	9	1
150	11	2
180	12	1
210	13	1
240	14	1

Table 7.2: Failure based failure specification

Software Reliability

Value of random variable (failures in time period)	Probability	
	Elapsed time t_A = 1 hr	Elapsed tim
0	0.10	0.0
1	0.18	0.0
2	0.22	0.0
3	0.16	0.0
4	0.11	0.0
5	0.08	0.0
6	0.05	0.0
7	0.04	0.1
8	0.03	0.1
9	0.02	0.1

Table 7.3: Probability distribution at times t_A and t_B

Software Reliability

Value of random variable (failures in time period)	Probability	
	Elapsed time t_A = 1 hr	Elapsed tim
10	0.01	0.1
11	0	0.0
12	0	0.0
13	0	0.0
14	0	0.0
15	0	0.0
Mean failures	3.04	7.7

Table 7.3: Probability distribution at times t_A and t_B

Software Reliability

A random process whose probability distribution varies time is called non-homogeneous. Most failure processes fit this situation. Fig. 7.3 illustrates the mean value and failure intensity functions at time t_A and t_B. Note tha failures experienced increases from 3.04 to 7.77 betwee points, while the failure intensity decreases.

Failure behavior is affected by two principal factors:

- ✓ the number of faults in the software being execu
- ✓ the execution environment or the operational execution.

Software Reliability

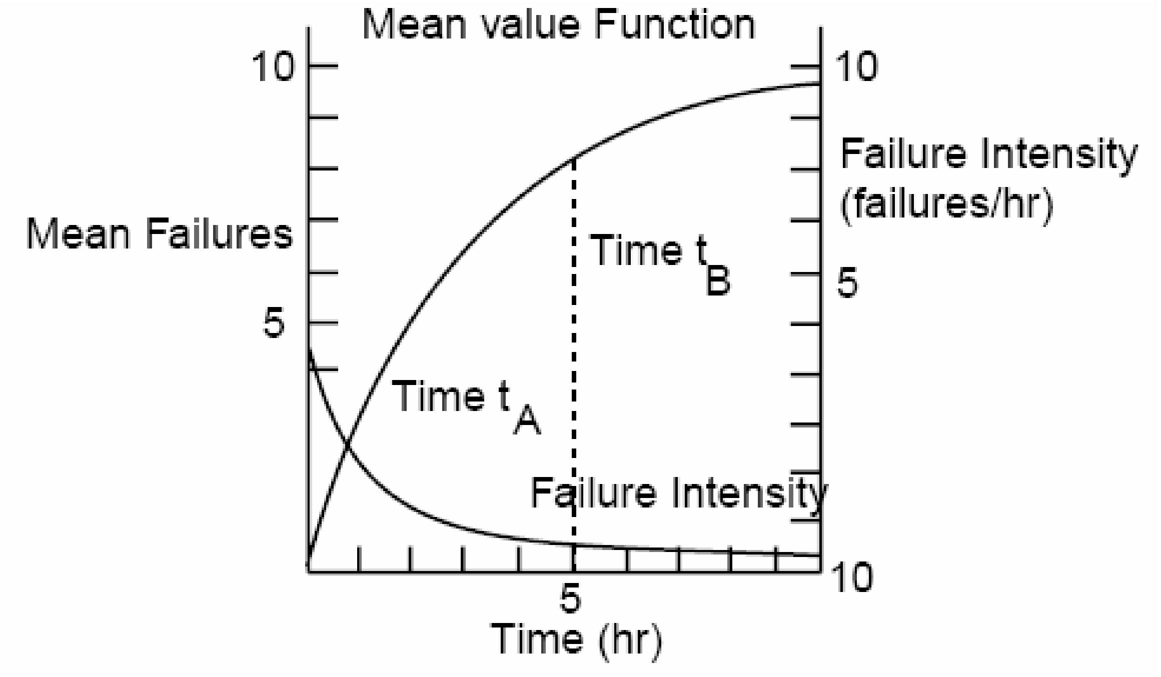


Fig. 7.3: Mean Value & failure intensity functions.

Software Reliability

Environment

The environment is described by the operational
proportion of runs of various types may vary, depen
functional environment. Examples of a run type might be

1. a particular transaction in an airline reservation s
   business data processing system,
2. a specific cycle in a closed loop control s
   example, in a chemical process industry),
3. a particular service performed by an operating s
   user.

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

The run types required of the program by the environn
viewed as being selected randomly. Thus, we define the
profile as the set of run types that the program can ex
with possibilities with which they will occur. In fig. 7.4, w
of many possible input states A and B, with their pro
occurrence.

The part of the operational profile for just these two stat
in fig. 7.5. A realistic operational profile is illustrated in fig

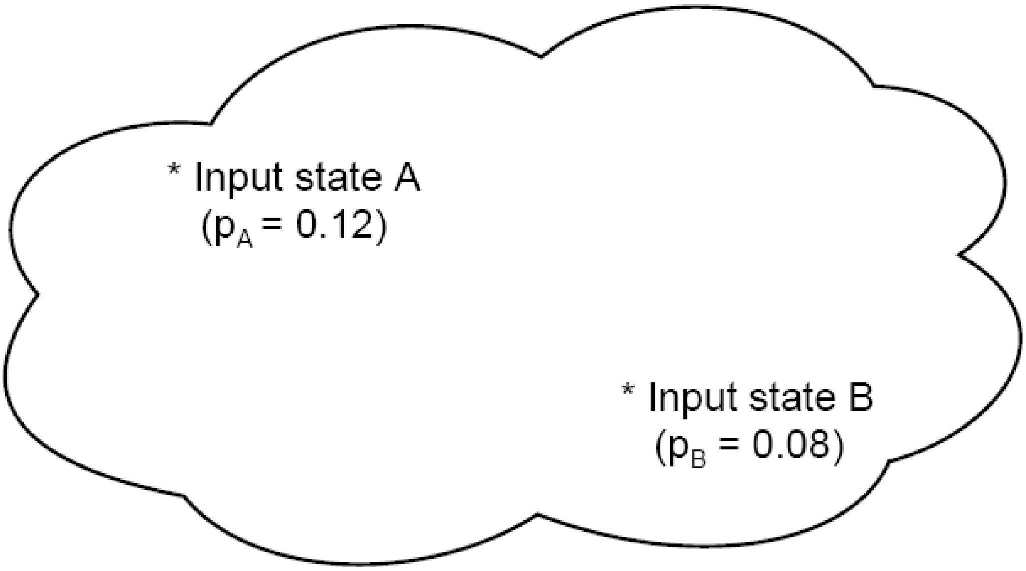


Fig. 7.4: Input Space

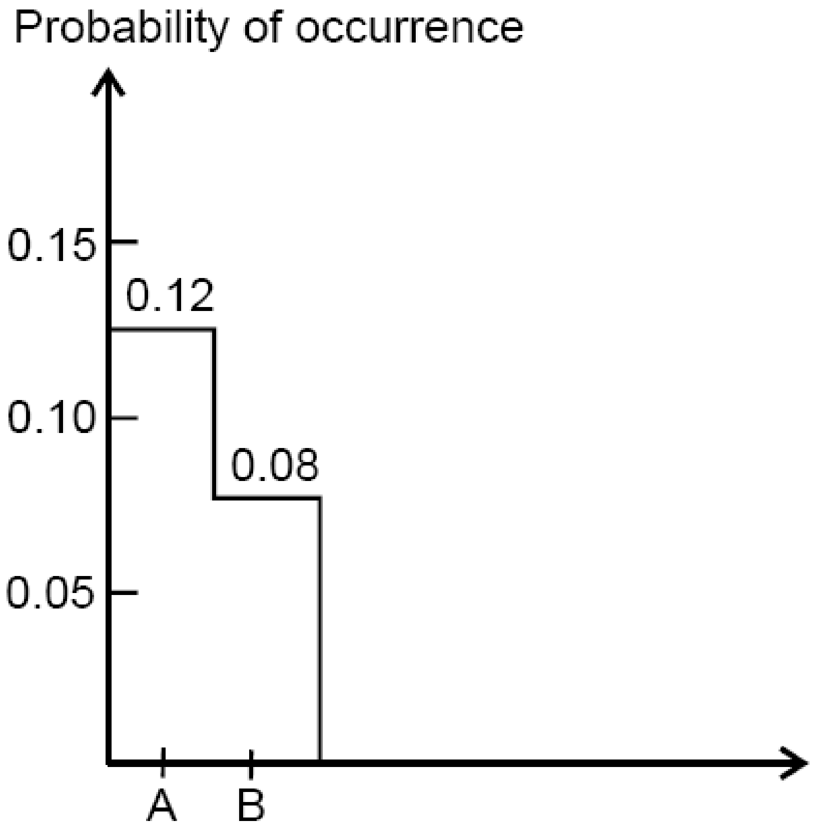


Fig. 7.5: Portion of operational profile

Software Reliability

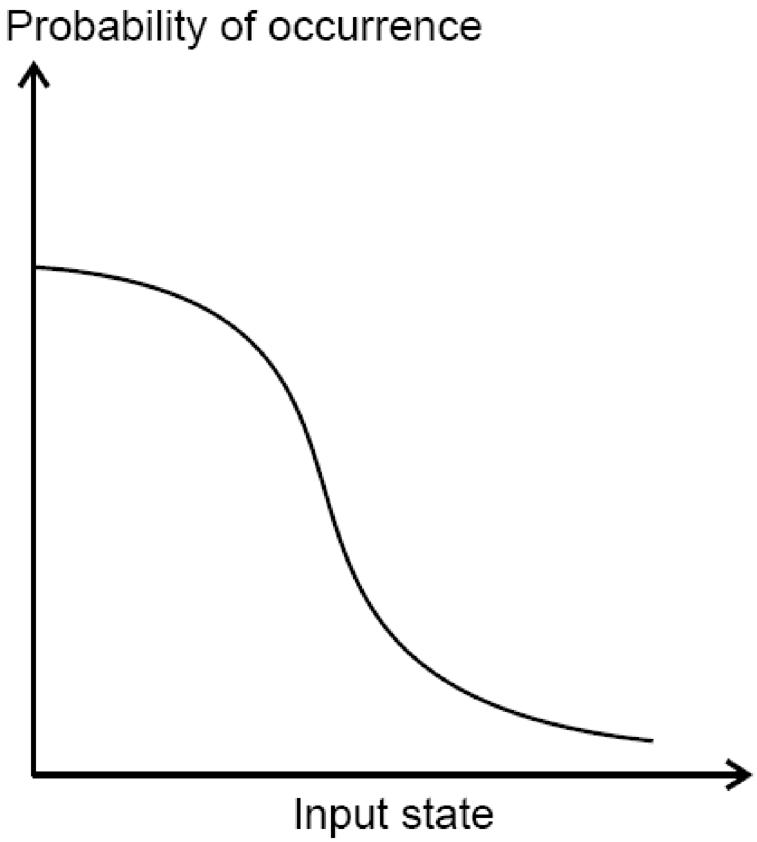


Fig. 7.6: Operational profile

Software Reliability

Fig.7.7 shows how failure intensity and reliability ty during a test period, as faults are removed.

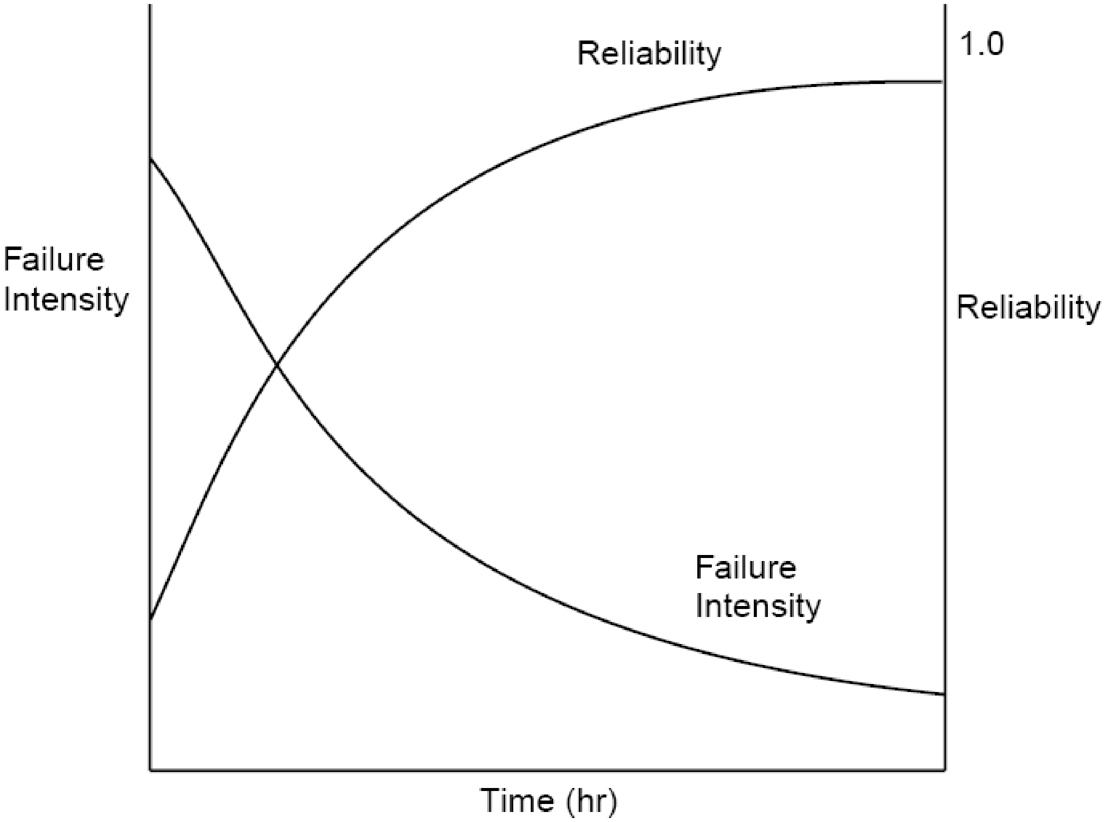


Fig. 7.7: Reliability and failure intensity

Software Reliability

Uses of Reliability Studies

There are at least four other ways in which softwa
measures can be of great value to the software engine
or user.

1. you can use software reliability measures to evalua
engineering technology quantitatively.

2. software reliability measures offer you the po
evaluating development status during the test pl
project.

Software Reliability

3. one can use software reliability measures to operational performance of software and to control r added and design changes made to the software.

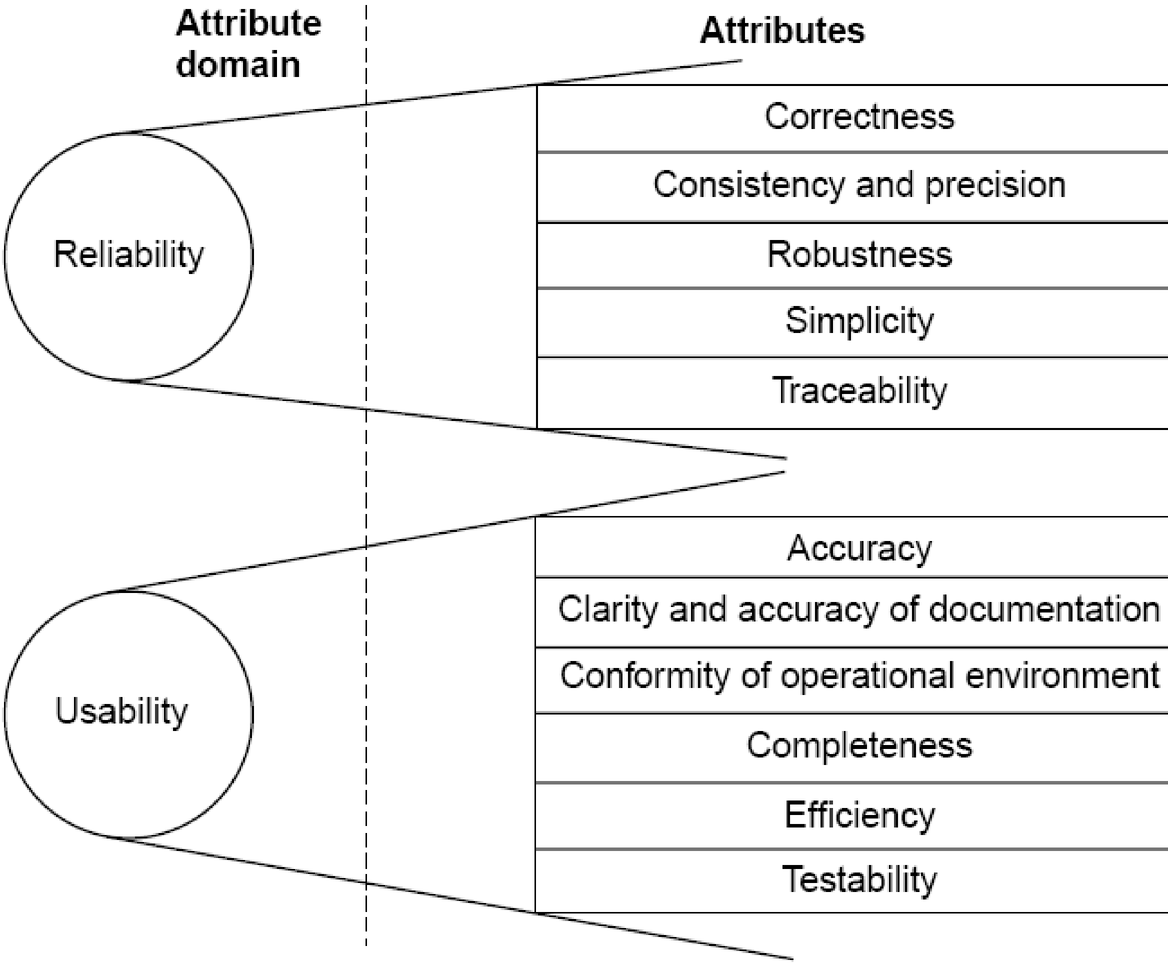
4. a quantitative understanding of software quality and factors influencing it and affected by it enriche software product and the software development proc

Software Quality

Different people understand different meanings of quality

- conformance to requirements
- fitness for the purpose
- level of satisfaction

Software Reliability



Attribute domain

Attributes

Reliability

- Correctness
- Consistency and precision
- Robustness
- Simplicity
- Traceability

Usability

- Accuracy
- Clarity and accuracy of documentation
- Conformity of operational environment
- Completeness
- Efficiency
- Testability

Software Reliability

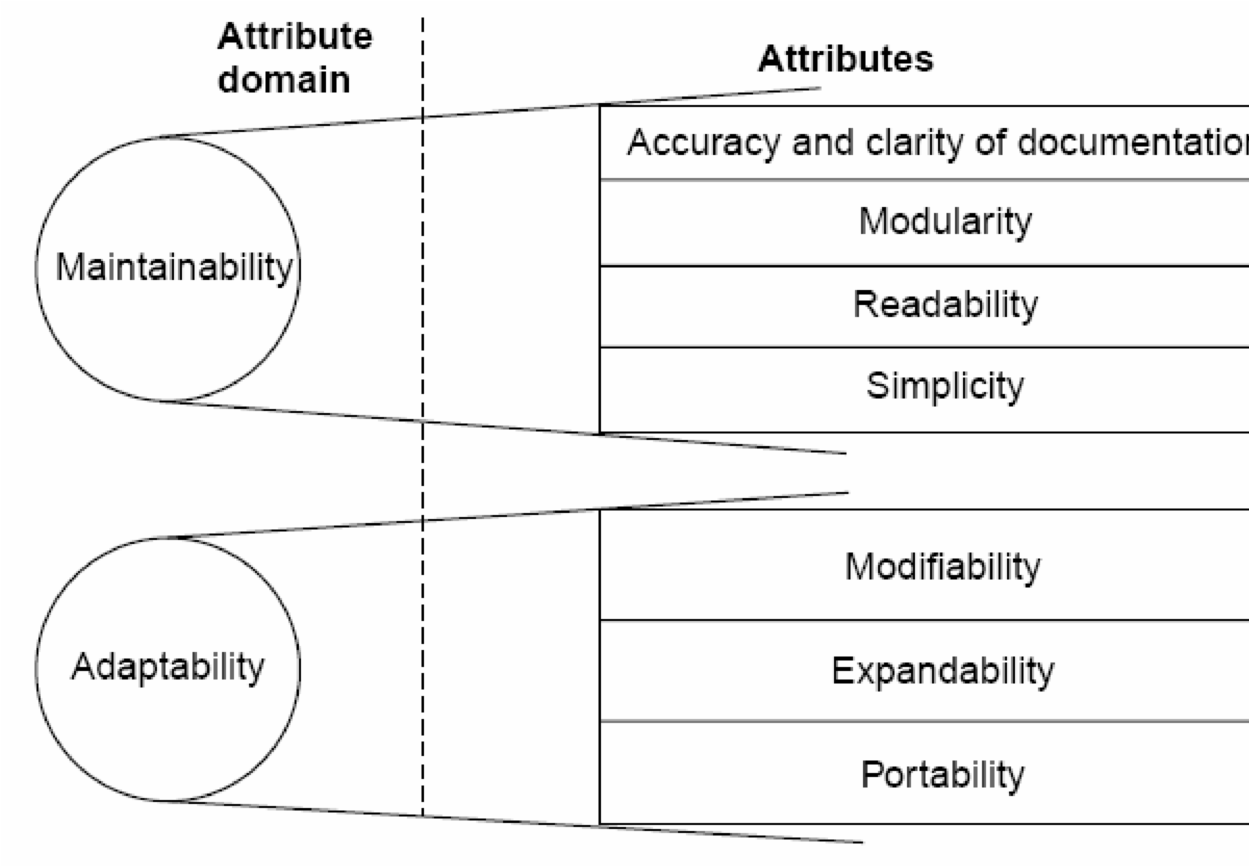


Fig 7.8: Software quality attributes

Software Reliability

1	Reliability	The extent to which a software performs functions without failure.
2	Correctness	The extent to which a software specifications.
3	Consistency & precision	The extent to which a software is consist results with precision.
4	Robustness	The extent to which a software to unexpected problems.
5	Simplicity	The extent to which a software is s operations.
6	Traceability	The extent to which an error is traceabl fix it.
7	Usability	The extent of effort required to learn, understand the functions of the software

(C

Software Reliability

8	Accuracy	Meeting specifications with precision.
9	Clarity & Accuracy of documentation	The extent to which documents are clearly written.
10	Conformity of operational environment	The extent to which a software is in c operational environment.
11	Completeness	The extent to which a software has specifie
12	Efficiency	The amount of computing resources and c by software to perform a function.
13	Testability	The effort required to test a software to e performs its intended functions.
14	Maintainability	The effort required to locate and fix an maintenance phase.

(C

Software Reliability

15	Modularity	It is the extent of ease to implement, tes maintain the software.
16	Readability	The extent to which a software is readab understand.
17	Adaptability	The extent to which a software is adap platforms & technologies.
18	Modifiability	The effort required to modify a soft maintenance phase.
19	Expandability	The extent to which a software is expanc undesirable side effects.
20	Portability	The effort required to transfer a progra platform to another platform.

Table 7.4: Software quality attributes

- McCall Software Quality Model

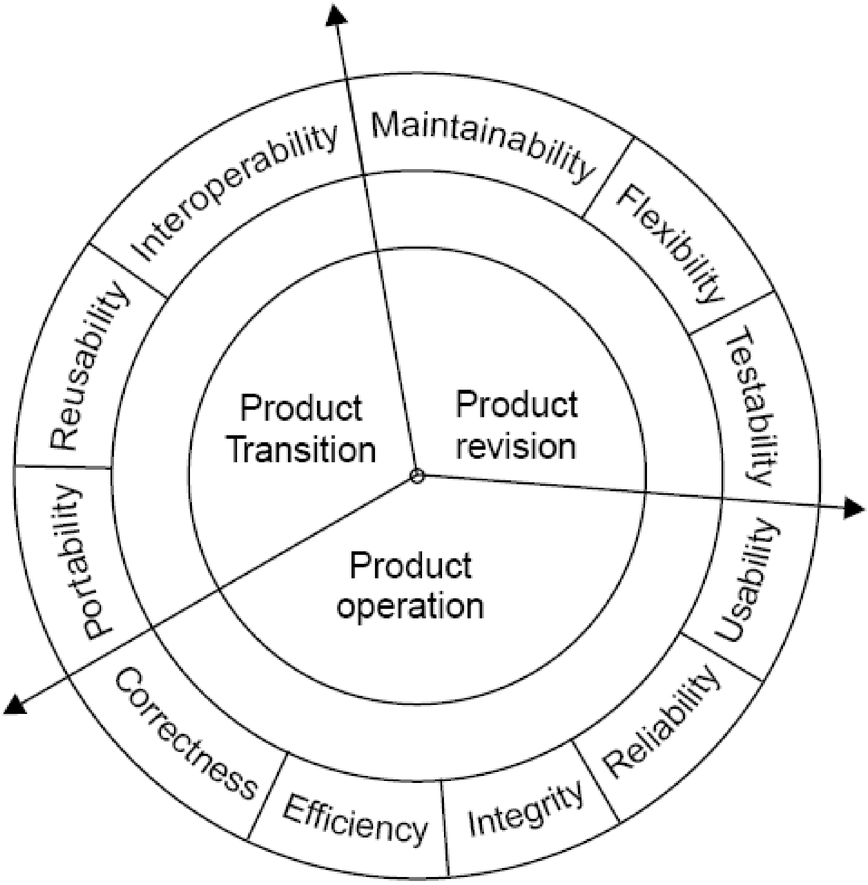


Fig 7.9: Software quality factors

Software Reliability

i. Product Operation

Factors which are related to the operation of a combined. The factors are:

- Correctness
- Efficiency
- Integrity
- Reliability
- Usability

These five factors are related to operational convenience, ease of usage and its correctness. These a very significant role in building customer's satisfaction.

Software Reliability

ii. Product Revision

The factors which are required for testing & maint
combined and are given below:

- Maintainability
- Flexibility
- Testability

These factors pertain to the testing & maintainability
They give us idea about ease of maintenance, flexibility
effort. Hence, they are combined under the umbrella
revision.

iii. Product Transition

We may have to transfer a product from one platform platform or from one technology to another technology. related to such a transfer are combined and given below:

- Portability
- Reusability
- Interoperability

Software Reliability

Most of the quality factors are explained in table 7.4. Th
factors are given in table 7.5.

Sr.No.	Quality Factors	Purpose
1	Integrity	The extent to which access to softwar the unauthorized persons can be contro
2	Flexibility	The effort required to modify an operatic
3	Reusability	The extent to which a program can other applications.
4	Interoperability	The effort required to couple one another.

Table 7.5: Remaining quality factors (other are in tabl

Quality criteria

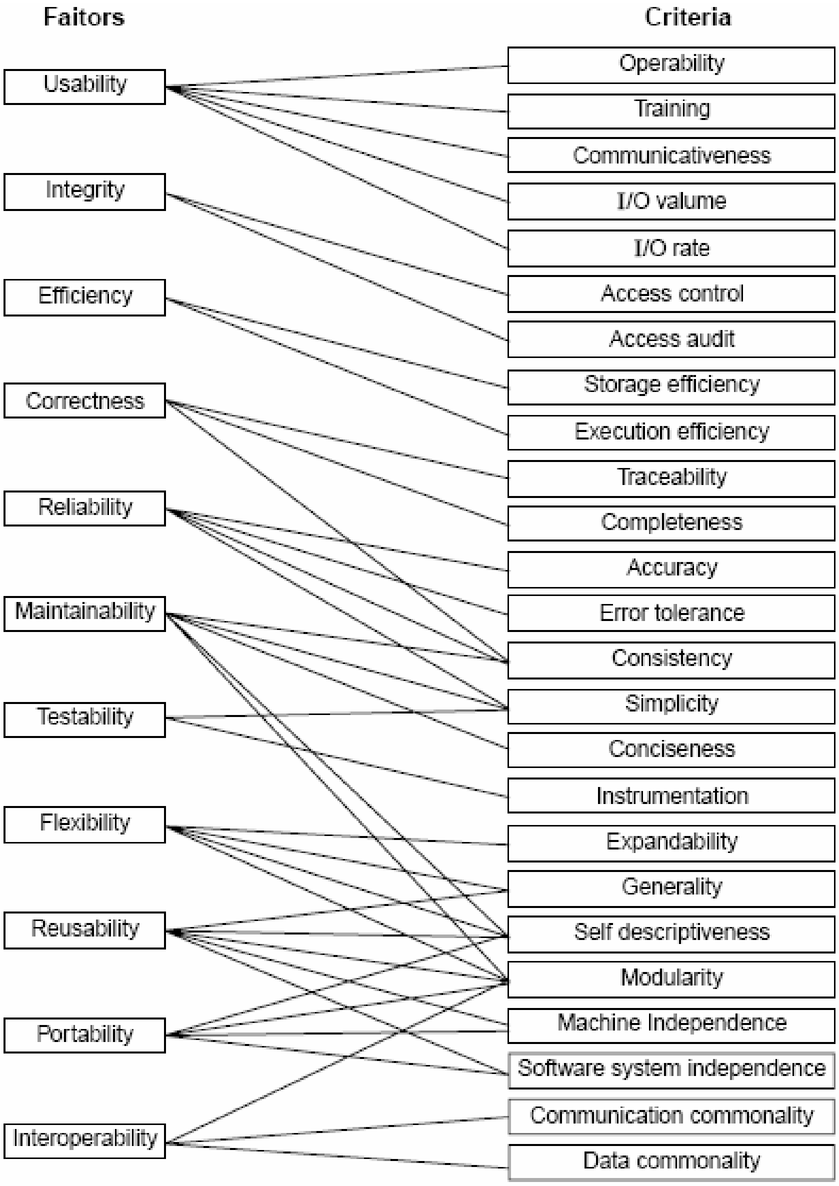
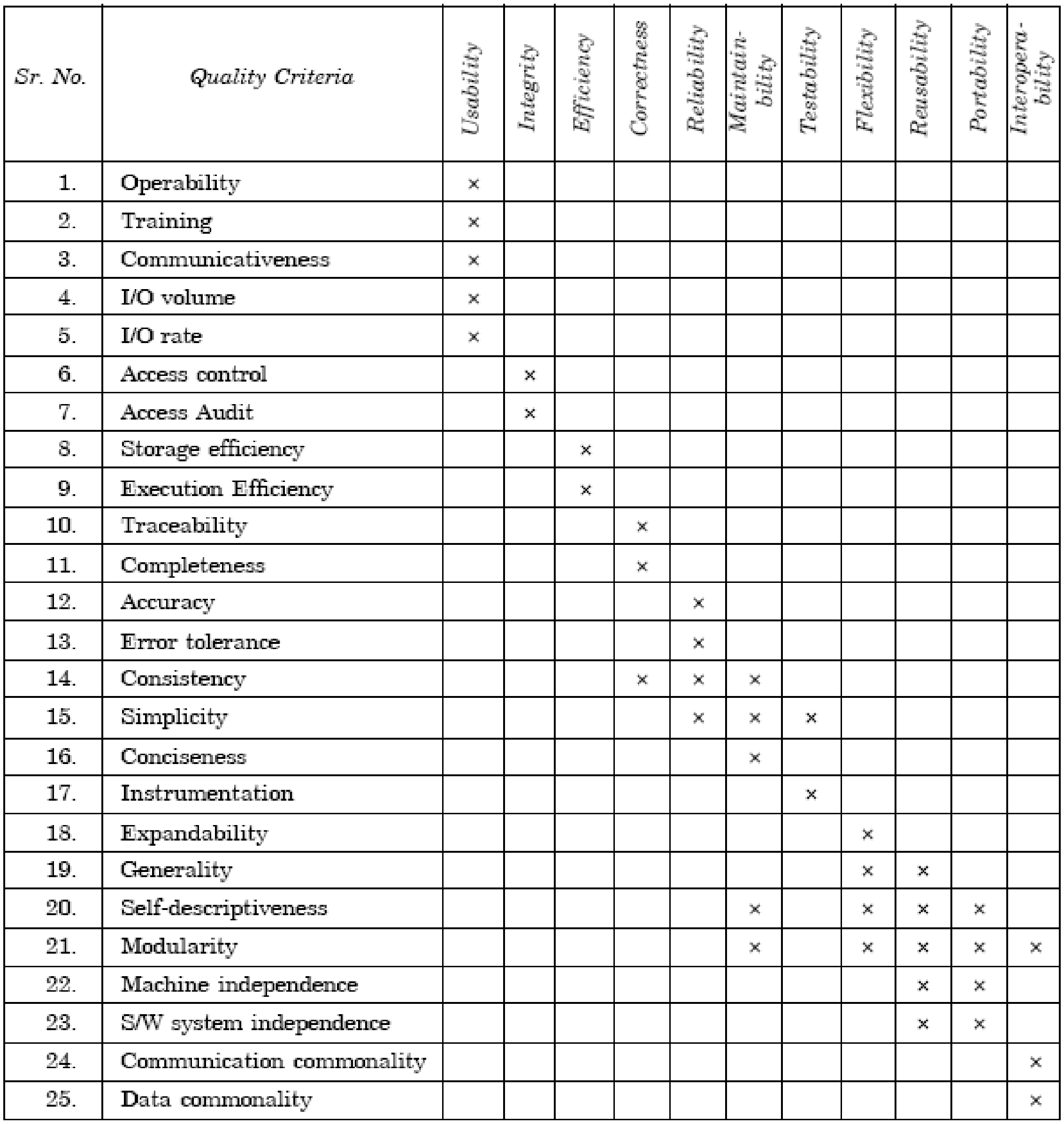


Fig 7.10: McCall's quality model

Software Engineering (3^{rd} ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability



Sr. No.	Quality Criteria	Usability	Integrity	Efficiency	Correctness	Reliability	Maintain-bility	Testability	Flexibility	Reusability	Portability	Interopera-bility
1.	Operability	×										
2.	Training	×										
3.	Communicativeness	×										
4.	I/O volume	×										
5.	I/O rate	×										
6.	Access control		×									
7.	Access Audit		×									
8.	Storage efficiency			×								
9.	Execution Efficiency			×								
10.	Traceability				×							
11.	Completeness				×							
12.	Accuracy					×						
13.	Error tolerance					×						
14.	Consistency				×	×	×					
15.	Simplicity					×	×	×				
16.	Conciseness						×					
17.	Instrumentation							×				
18.	Expandability								×			
19.	Generality								×	×		
20.	Self-descriptiveness						×		×	×	×	
21.	Modularity						×		×	×	×	×
22.	Machine independence									×	×	
23.	S/W system independence									×	×	
24.	Communication commonality											×
25.	Data commonality											×

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Ta

betw

fa

qu

Software Reliability

1	Operability	The ease of operation of the software.
2	Training	The ease with which new users c system.
3	Communicativeness	The ease with which inputs and out assimilated.
4	I/O volume	It is related to the I/O volume.
5	I/O rate	It is the indication of I/O rate.
6	Access control	The provisions for control and prote software and data.
7	Access audit	The ease with which software and c checked for compliance with standa requirements.
8	Storage efficiency	The run time storage requirements of t
9	Execution efficiency	The run-time efficiency of the software

(C

Software Reliability

10	Traceability	The ability to link software comp requirements.
11	Completeness	The degree to which a full implementa required functionality has been achieved.
12	Accuracy	The precision of computations and output.
13	Error tolerance	The degree to which continuity of operatio under adverse conditions.
14	Consistency	The use of uniform design and imp techniques and notations throughout a proj
15	Simplicity	The ease with which the software can be u
16	Conciseness	The compactness of the source code, in te of code.
17	Instrumentation	The degree to which the software p measurements of its use or identification of

(C

Software Reliability

18	Expandability	The degree to which storage requ software functions can be expanded.
19	Generability	The breadth of the potential application components.
20	Self-descriptiveness	The degree to which the documen explanatory.
21	Modularity	The provision of highly independent mod
22	Machine independence	The degree to which software is depe associated hardware.
23	Software system independence	The degree to which software is indepe environment.
24	Communication commonality	The degree to which standard pro interfaces are used.
25	Data commonality	The use of standard data representations

Table 7.5 (b): Software quality criteria

Software Reliability

- Boehm Software Quality Model

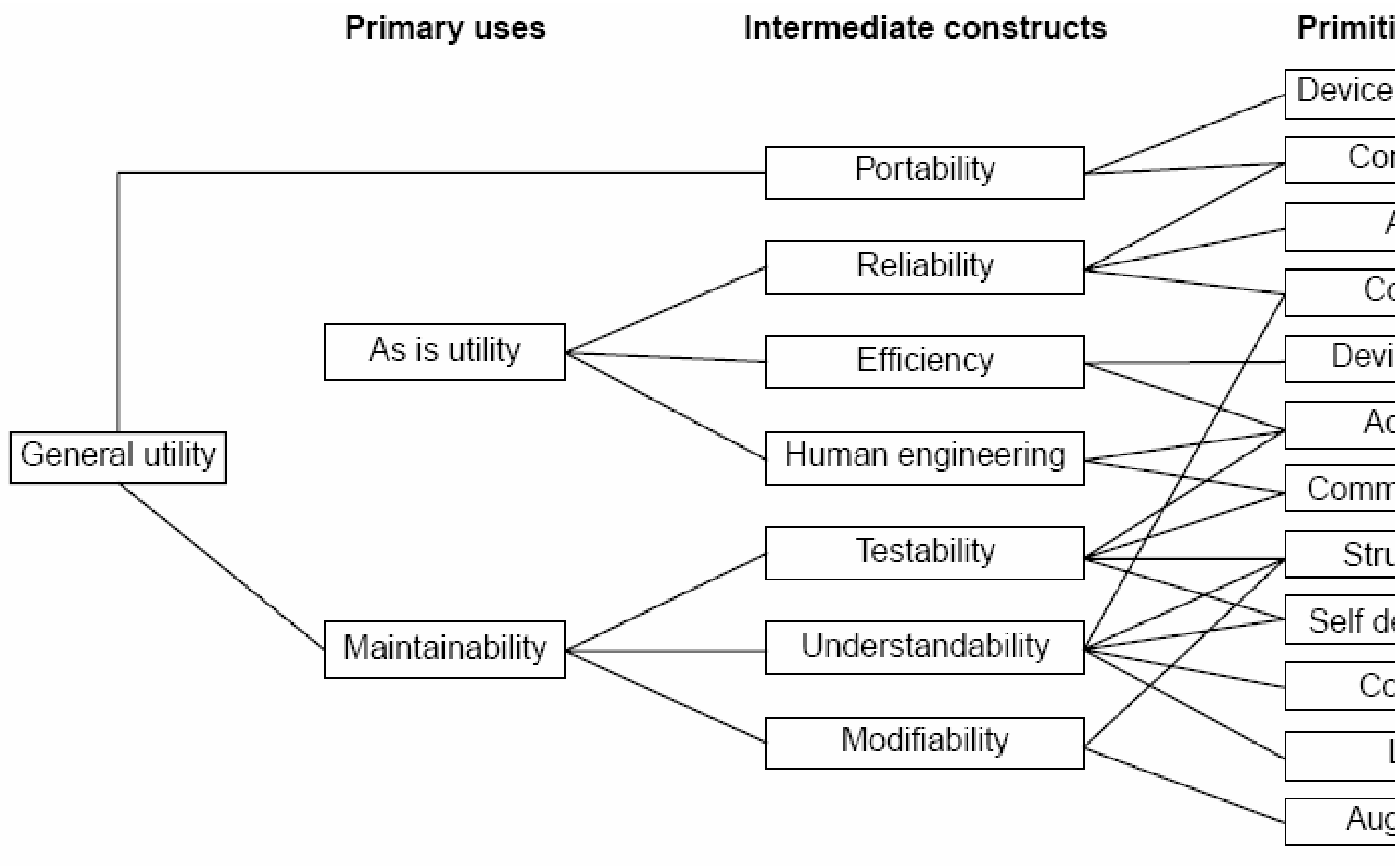


Fig.7.11: The Boehm software quality model

ISO 9126

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Software Reliability

Characteristic/ Attribute	Short Description of the Characteristic concerns Addressed by Attribut
Functionality	Characteristics relating to achievement c purpose for which the software is being engin
• Suitability	The presence and appropriateness of a set o specified tasks
• Accuracy	The provision of right or agreed results or effe
• Interoperability	Software’s ability to interact with specified sys
• Security	Ability to prevent unauthorized access, wheth or deliberate, to program and data.
Reliability	Characteristics relating to capability of maintain its level of performance under stat for a stated period of time
• Maturity	Attributes of software that bear on the freque by faults in the software

(C

Software Reliability

• Fault tolerance	Ability to maintain a specified level of performa of software faults or unexpected inputs
• Recoverability	Capability and effort needed to reestabl performance and recover affected data a failure.
Usability	Characteristics relating to the effort needed fo the individual assessment of such use, by a s set of users.
• Understandability	The effort required for a user to recogniz concept and its applicability.
• Learnability	The effort required for a user to learn its operation, input and output.
• Operability	The ease of operation and control by users.
Efficiency	Characteristic related to the relationship betw of performance of the software and the resources used, under stated conditions.

(C

Software Reliability

• Time behavior	The speed of response and processing throughout rates in performing its function.
• Resource behavior	The amount of resources used and the dur use in performing its function.
Maintainability	Characteristics related to the effort need modifications, including corrections, impr adaptation of software to changes in requirements and functions specifications.
• Analyzability	The effort needed for diagnosis of deficienci of failures, or for identification of parts to be r
• Changeability	The effort needed for modification, fault re environmental change.
• Stability	The risk of unexpected effect of modifications
• Testability	The effort needed for validating the modified

(C

Software Reliability

Portability	Characteristics related to the ability to software from one organization or hardware environment to another.
• Adaptability	The opportunity for its adaptation to differe environments.
• Installability	The effort needed to install the software in environment.
• Conformance	The extent to which it adheres to st conventions relating to portability.
• Replaceability	The opportunity and effort of using it in the p software in a particular environment.

Table 7.6: Software quality characteristics and attributes – The view

Software Reliability

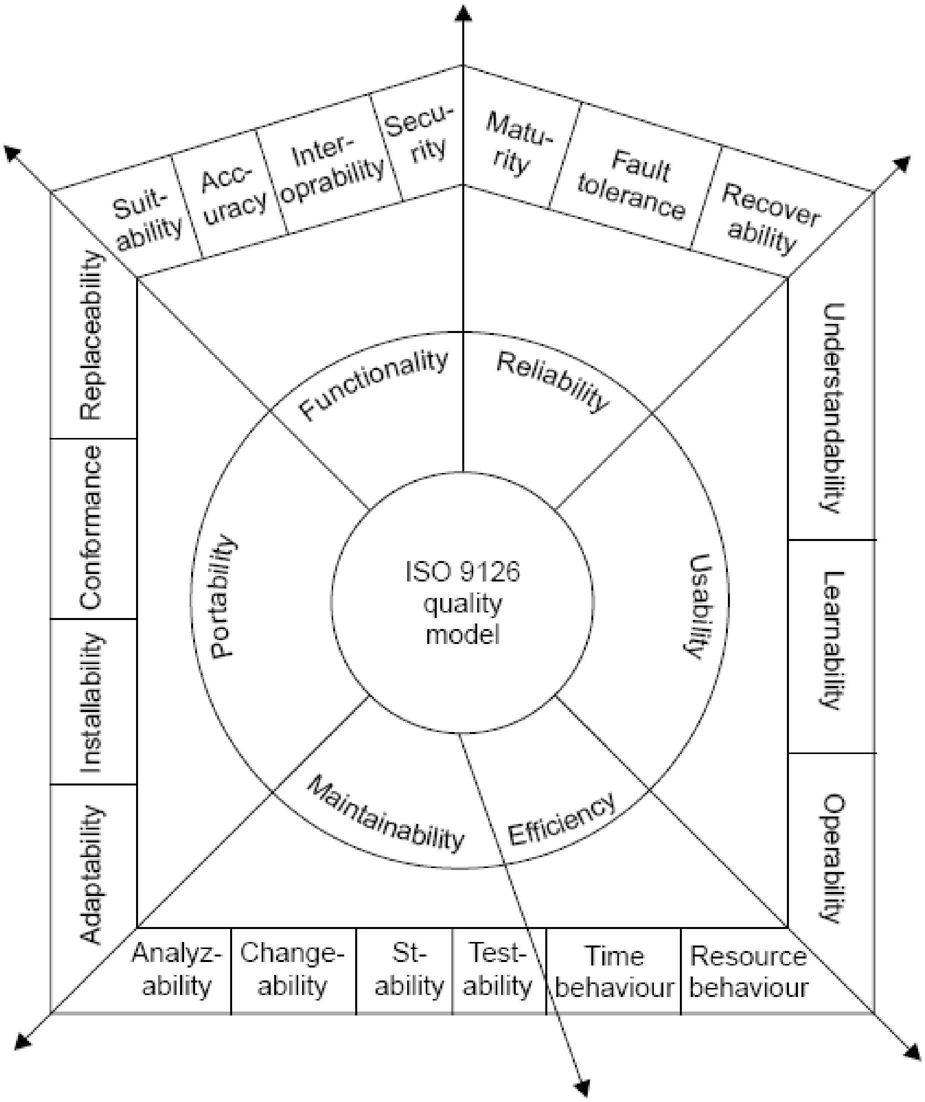
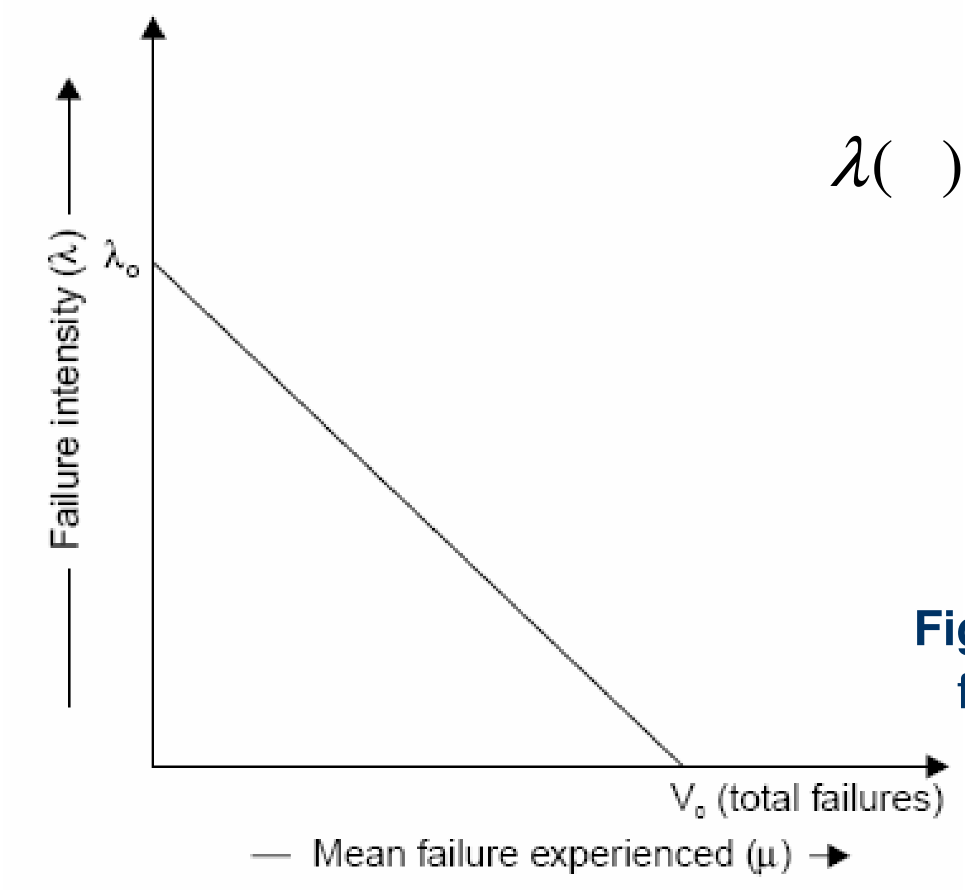


Fig.7.12: ISO 9126 quality model

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability Models

- Basic Execution Time Model



$$\lambda(\ ) = \lambda_0\left(1 - \frac{}{V_0}\right)$$

Fig.7.13: Failure in
function of μ for b

Software Reliability

$$\frac{d\lambda}{d} = \frac{-\lambda_0}{V_0} \qquad (2)$$

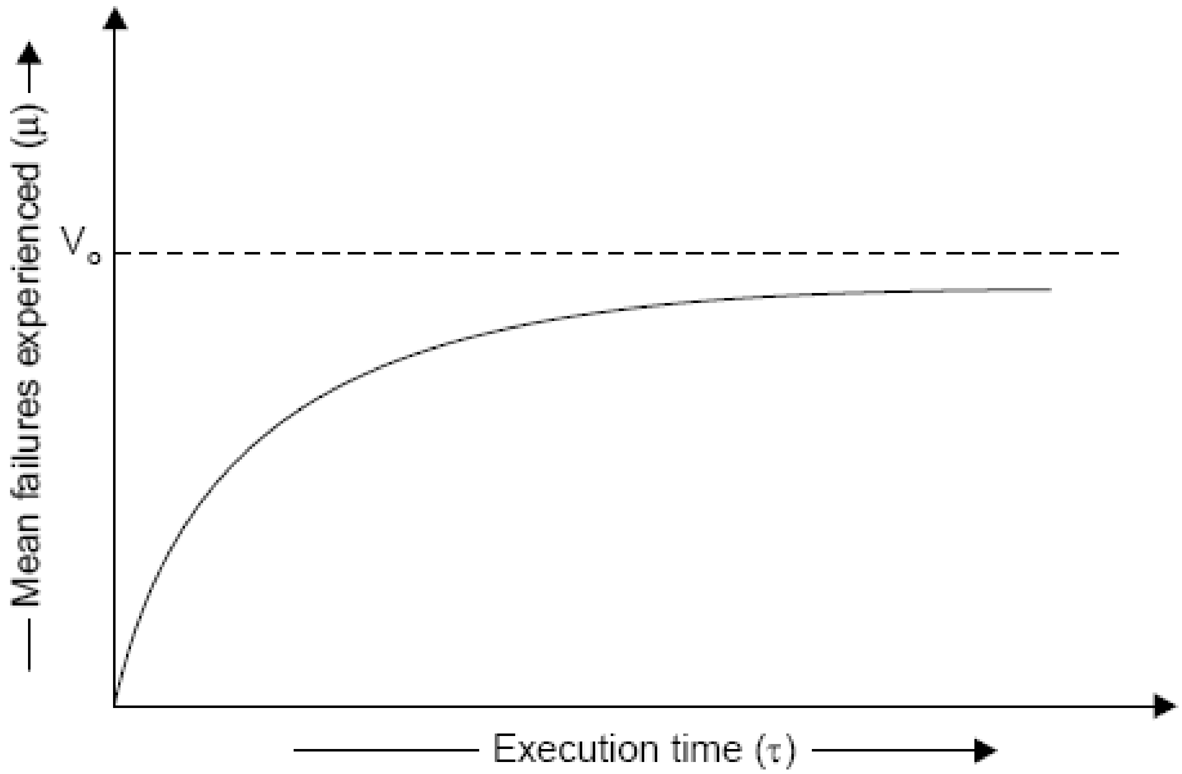


Fig.7.14: Relationship between τ & μ for basic model

Software Reliability

For a derivation of this relationship, equation 1 can be wr

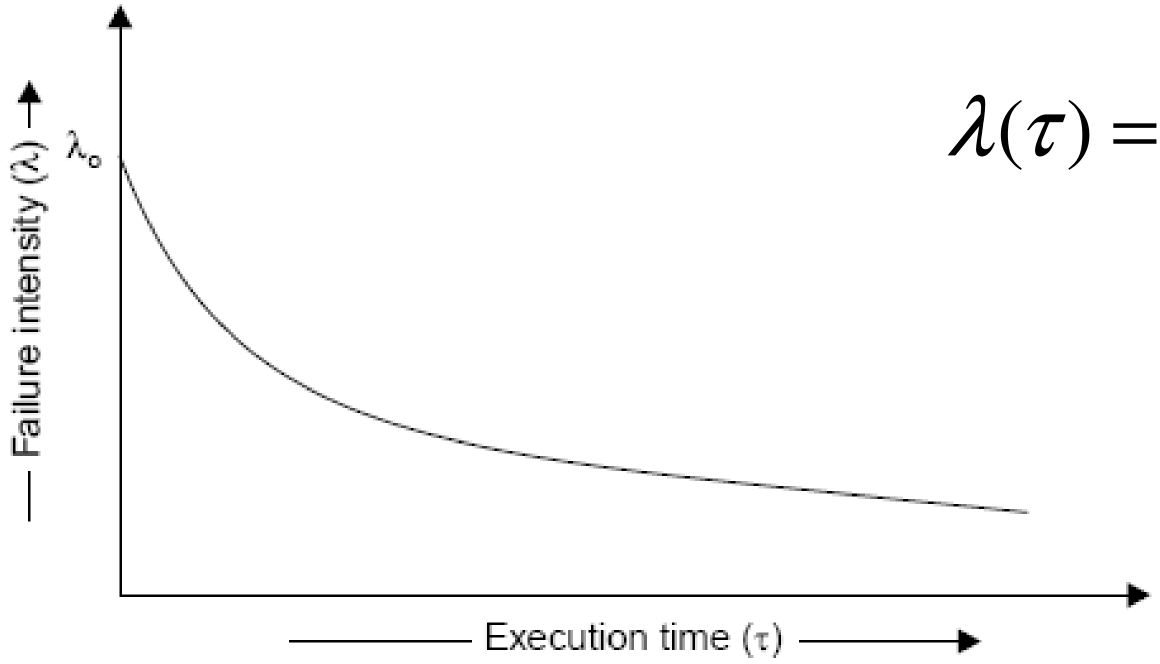
$$\frac{d\quad(\tau)}{d\tau} = \lambda_0\left(1 - \frac{\quad(\tau)}{V_0}\right)$$

The above equation can be solved for $\quad(\tau)$ and result i

$$\quad(\tau) = V_0\left(1 - \exp\left(\frac{-\lambda_0\tau}{V_0}\right)\right)$$

Software Reliability

The failure intensity as a function of execution time i
figure given below



$$\lambda(\tau) = \lambda_0 \exp\Bigg($$

Fig.7.15: Failure intensity versus execution time for basic model

Software Reliability

- Derived quantities

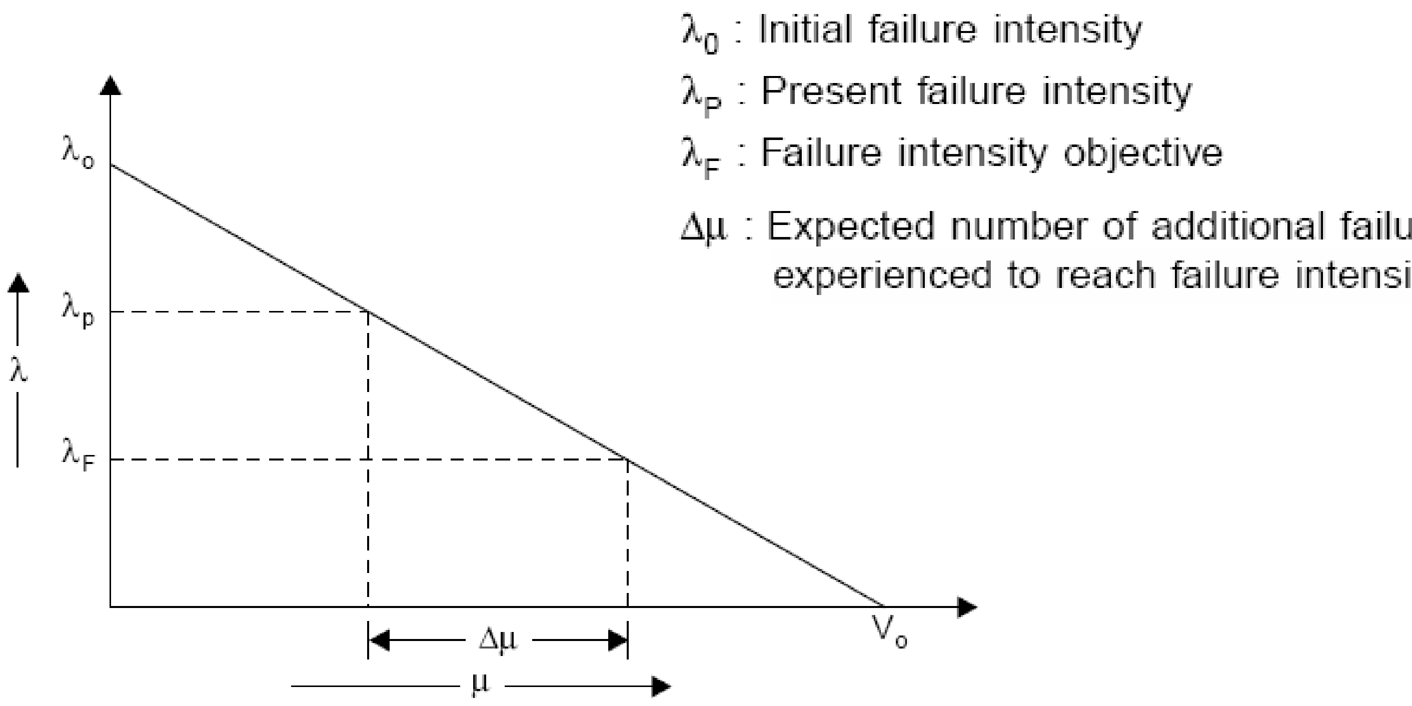


Fig.7.16: Additional failures required to be experienced to reac objective

Software Reliability

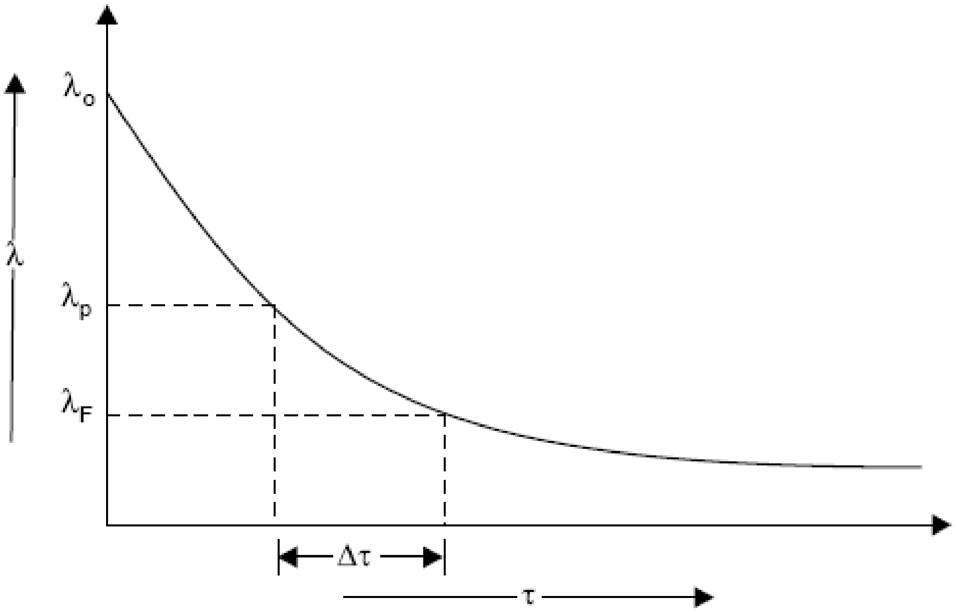


Fig.7.17: Additional time required to reach th
objective

This can be derived in mathematical form as:

$$\Delta\tau = \frac{V_0}{\lambda_0} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

Example- 7.1

Assume that a program will experience 200 failures in infinite
now experienced 100. The initial failure intensity was 20 failure

(i) Determine the current failure intensity.

(ii) Find the decrement of failure intensity per failure.

(iii) Calculate the failures experienced and failure intensity afte CPU hrs. of execution.

(iv) Compute addition failures and additional execution time reach the failure intensity objective of 5 failures/CPU hr.

Use the basic execution time model for the above mentioned ca

Solution

Here V_o=200 failures

$$=100 \text{ failures}$$

$$\lambda_0 = 20 \text{ failures/CPU hr.}$$

(i) Current failure intensity:

$$\lambda(\quad) = \lambda_0\left(1-\frac{}{V_0}\right)$$

$$= 20\left(1-\frac{100}{200}\right) = 20(1-0.5) = 10 \text{ failures/CPU hr}$$

Software Reliability

(ii) Decrement of failure intensity per failure can be calcul

$$\frac{d\lambda}{d} = \frac{-\lambda_0}{V_0} = -\frac{20}{200} = -0.1/\text{CPU hr.}$$

(iii) (a) Failures experienced & failure intensity after 20 C

$$(\tau) = V_0\left(1 - \exp\left(\frac{-\lambda_0 \tau}{V_0}\right)\right)$$

$$= 200\left(1 - \exp\left(\frac{-20 \quad 20}{200}\right)\right) = 200(1 - \exp(1 - 2))$$

$$= 200(1 - 0.1353) \approx 173 \text{failures}$$

Software Reliability

$$\lambda(\tau) = \lambda_0 \exp\left(\frac{-\lambda_0 \tau}{V_0}\right)$$

$$= 20 \exp\left(\frac{-20 \quad 20}{200}\right) = 20 \exp(-2) = 2.71 \, failures / C$$

(b) Failures experienced & failure intensity after 100 CPU

$$(\tau) = V_0\left(1 - \exp\left(\frac{-\lambda_0 \tau}{V_0}\right)\right)$$

$$= 200\left(1 - \exp\left(\frac{-20 \quad 100}{200}\right)\right) = 200 \text{ failures(almost}$$

$$\lambda(\tau) = \lambda_0 \exp\left(\frac{-\lambda_0 \tau}{V_0}\right)$$

Software Reliability

$$= 20\exp\left(\frac{-20 \quad 100}{200}\right) = 0.000908\ failures / CPU\ hr$$

(iv) Additional failures $(\Delta \quad)$ required to reach the failure i objective of 5 failures/CPU hr.

$$\Delta \quad = \left(\frac{V_0}{\lambda_0}\right)(\lambda_P - \lambda_F) = \left(\frac{200}{20}\right)(10-5) = 50 \text{ failures}$$

Software Reliability

Additional execution time required to reach failure intensi
of 5 failures/CPU hr.

$$\Delta\tau = \left(\frac{V_0}{\lambda_0}\right) Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

$$= \frac{200}{20} Ln\left(\frac{10}{5}\right) = 6.93 \text{ CPU hr.}$$

Software Reliability

- Logarithmic Poisson Execution Time Model

Failure Intensity

$$\lambda(\quad) = \lambda_0 \exp(-\theta\quad)$$

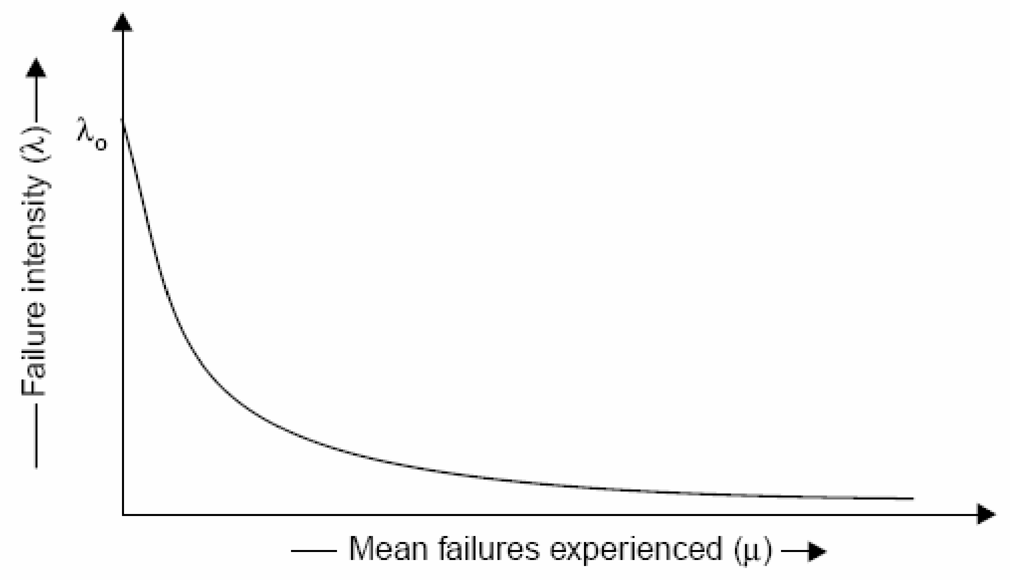


Fig.7.18: Relationship between μ & λ

Software Reliability

$$\frac{d\lambda}{d} = -\lambda_0 \theta \exp(- \quad \theta)$$

$$\frac{d\lambda}{d} = -$$

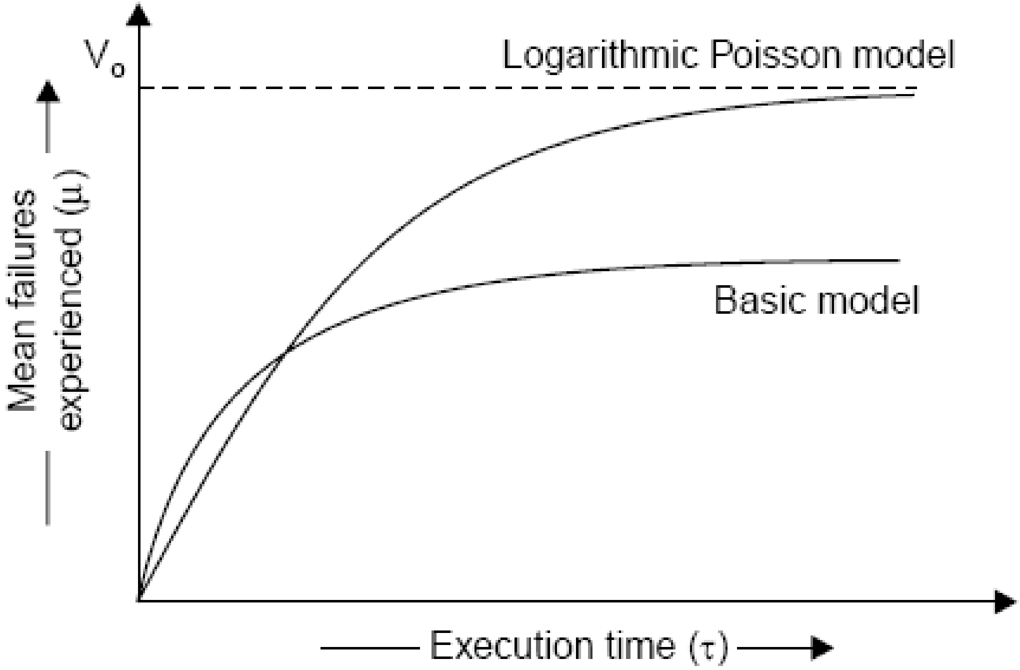


Fig.7.19: Relationship between

Software Reliability

$$(\tau) = \frac{1}{\theta} Ln(\lambda_0 \theta \tau + 1)$$

$$\lambda(\tau) = \lambda_0 / (\lambda_0 \theta \tau + 1)$$

$$\Delta \quad = \frac{1}{\theta} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

$$\Delta\tau = \frac{1}{\theta}\left[\frac{1}{\lambda_F} - \frac{1}{\lambda_P}\right]$$

λ_P = Present failure intensi

λ_F = Failure intensity objec

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Example- 7.2

Assume that the initial failure intensity is 20 failures/CPU hr intensity decay parameter is 0.02/failures. We have expe failures up to this time.

(i) Determine the current failure intensity.

(ii) Calculate the decrement of failure intensity per failure.

(iii) Find the failures experienced and failure intensity after 20 hrs. of execution.

(iv) Compute the additional failures and additional execution ti reach the failure intensity objective of 2 failures/CPU hr.

Use Logarithmic Poisson execution time model for the abo calculations.

Software Reliability

Solution

$$\lambda_0 = 20 \text{ failures/CPU hr.}$$

$$= 100 \text{ failures}$$

$$\theta = 0.02 / \text{failures}$$

(i) Current failure intensity:

$$\lambda(\quad) = \lambda_0 \exp(-\theta\quad)$$

= 20 exp (-0.02 x 100)

= 2.7 failures/CPU hr.

Software Reliability

(ii) Decrement of failure intensity per failure can be calcul

$$\frac{d\lambda}{d} = -\theta\lambda$$

= -.02 x 2.7 = -.054/CPU hr.

(iii) (a) Failures experienced & failure intensity after 20 C

$$(\tau) = \frac{1}{\theta} Ln\left(\lambda_0 \theta \tau + 1\right)$$

$$= \frac{1}{0.02} Ln(20 \quad 0.02 \quad 20 + 1) = 109 \ failures$$

Software Reliability

$$\lambda(\tau) = \lambda_0 / \left(\lambda_0 \theta \tau + 1\right)$$

$$= (20)/(20 \quad .02 \quad 20+1) = 2.22 \ failures/CPU \ hr.$$

(b) Failures experienced & failure intensity after 100 CPU

$$(\tau) = \frac{1}{\theta} Ln\left(\lambda_0 \theta \tau + 1\right)$$

$$= \frac{1}{0.02} Ln(20 \quad 0.02 \quad 100+1) = 186 \ failures$$

$$\lambda(\tau) = \lambda_0 / \left(\lambda_0 \theta \tau + 1\right)$$

$$= (20)/(20 \quad .02 \quad 100+1) = 0.4878 \ failures/CPU$$

Software Reliability

(iv) Additional failures $(\Delta \quad)$ required to reach the failure i
objective of 2 failures/CPU hr.

$$\Delta \quad = \frac{1}{\theta} Ln \frac{\lambda_P}{\lambda_F} = \frac{1}{0.02} Ln \left(\frac{2.7}{2} \right) = 15 \text{ failures}$$

$$\Delta\tau = \frac{1}{\theta} \left[\frac{1}{\lambda_F} - \frac{1}{\lambda_P} \right] = \frac{1}{0.02} \left[\frac{1}{2} - \frac{1}{2.7} \right] = 6.5 \text{ CP}$$

Example- 7.3

The following parameters for basic and logarithmic Poisson

given:

(a) Determine the addition failures and additional execution ti
reach the failure intensity objective of 5 failures/CPU hr. for

(b) Repeat this for an objective function of 0.5 failure/CPU hr
we start with the initial failure intensity only.

Basic execution time model	Logarithmic Poisson execution time model
$\lambda_o = 10$ failures/CPU hr	$\lambda_o = 30$ failures/CPU hr
$V_o = 100$ failures	$\theta = 0.25/$ failure

Solution

(a) (i) Basic execution time model

$$\Delta \quad = \frac{V_0}{\lambda_0}(\lambda_P - \lambda_F)$$

$$= \frac{100}{10}(10-5) = 50 \text{ failures}$$

λ_P (Present failure intensity) in this case is same as failure intensity).

Now,
$$\Delta\tau = \frac{V_0}{\lambda_0} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

Software Reliability

$$= \frac{100}{10} Ln\left(\frac{10}{5}\right) = 6.93 \text{ CPU hr.}$$

(ii) Logarithmic execution time model

$$\Delta \quad = \frac{1}{\theta} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

$$= \frac{1}{0.025} Ln\left(\frac{30}{5}\right) = 71.67 \text{ Failures}$$

$$\Delta\tau = \frac{1}{\theta}\left(\frac{1}{\lambda_F} - \frac{1}{\lambda_P}\right)$$

$$= \frac{1}{0.025} Ln\left(\frac{1}{5} - \frac{1}{30}\right) = 6.66 \text{ CPU hr.}$$

Software Reliability

Logarithmic model has calculated more failures in almost som
execution time initially.

(b) Failure intensity objective (λ_F) = 0.5 failures/CPU hr.

(i) Basic execution time model

$$\Delta \quad = \frac{V_0}{\lambda_0}(\lambda_P - \lambda_F)$$

$$= \frac{100}{10}(10 - 0.5) = 95 \ failures$$

$$\Delta\tau = \frac{V_0}{\lambda_0} Ln\left(\frac{\lambda_P}{\lambda_F}\right.$$

$$= \frac{100}{10} Ln\left(\frac{10}{0.05}\right) = 3$$

(ii) Logarithmic execution time model

$$\Delta \quad = \frac{1}{\theta} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

$$= \frac{1}{0.025} Ln\left(\frac{30}{0.5}\right) = 164 \ failures$$

$$\Delta\tau = \frac{1}{\theta}\left(\frac{1}{\lambda_F} - \frac{1}{\lambda_P}\right)$$

$$= \frac{1}{0.025}\left(\frac{1}{0.5} - \frac{1}{30}\right) = 78.66 \, CPU/hr$$

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

- Calendar Time Component

The calendar time component is based on a debugg model. This model takes into account:

1. resources used in operating the program fo execution time and processing an associated failure.

2. resources quantities available, and

3. the degree to which a resource can be utiliz bottlenecks) during the period in which it is limiting

Table 7.7 will help in visualizing these different asp resources, and the parameters that result.

Resource usage

	Usage parameters requirements per		Planned para	
Resource	**CPU hr**	**Failure**	**Quantities available**	**Ut**
Failure identification personnel	θ_I	μ_I	P_I	
Failure correction personnel	0	μ_f	P_f	
Computer time	θc	μ_c	P_c	

Fig. : Calendar time component resources and parameters

Software Reliability

Hence, to be more precise, we have

$$X_C = {}_c\Delta \quad + \theta_c \Delta\tau$$ (for computer time)

$$X_f = {}_f\Delta$$ (for failure correctio

$$X_I = {}_I\Delta \quad + \theta_I \Delta\tau$$ (for failure identificat

$$dx_T / d\tau = \theta_r + {}_r\lambda$$

Calendar time to execution time relationship

$$dt / d\tau = (1 / P_r p_r) dx_T / d\tau$$

$$dt / d\tau = (\theta_r + \quad {}_r\lambda) / P_r p_r$$

Software Reliability

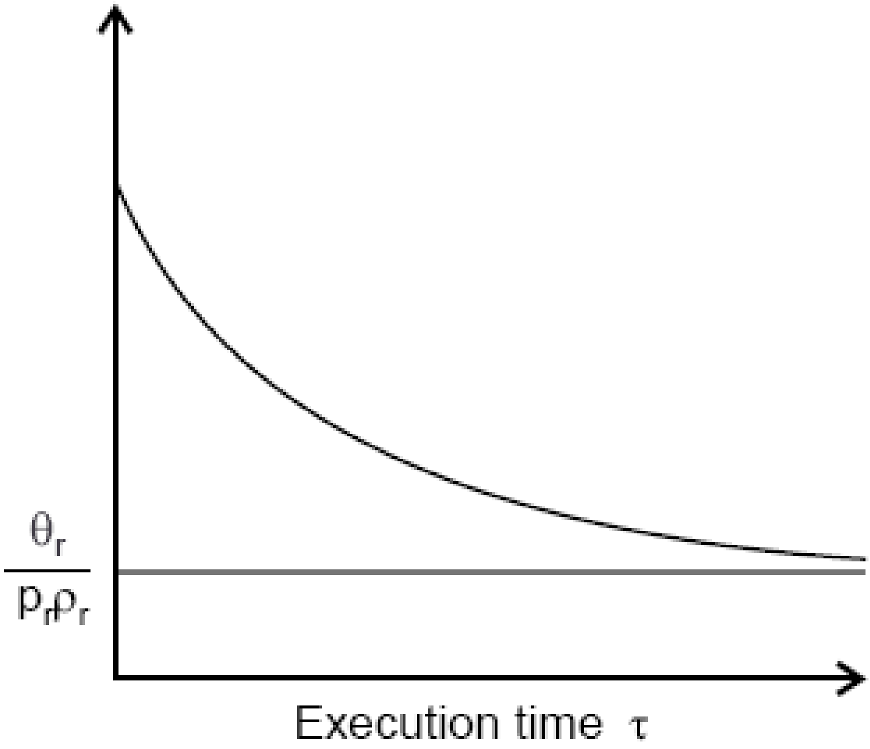


Fig.7.20: Instantaneous calendar time to execution time ratio

Software Reliability

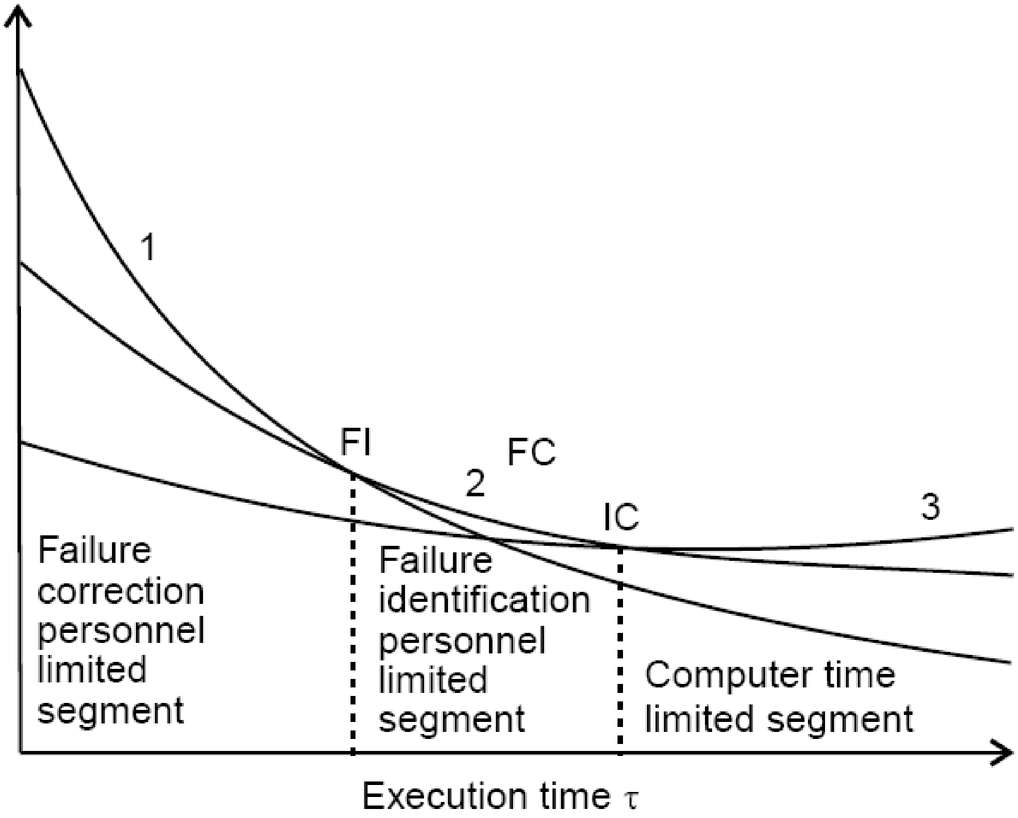


Fig.7.21: Calendar time to execution time ratio for differe
limiting resources

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

Example- 7.4

A team run test cases for 10 CPU hrs and identifies 25 failure required per hour of execution time is 5 person hr. Each failu hr. on an average to verify and determine its nature. Calcula identification effort required.

Solution

As we know, resource usage is:

$$X_r = \theta_r \tau + \quad _r$$

Here $\theta_r = 15$ person hr. $\quad = 25$ failures

$\tau = 10$ CPU hrs. $\quad _r = 2$ hrs./failure

Hence, $X_r = 5\ (10) + 2\ (25)$

$= 50 + 50 = 100$ person hr.

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Example- 7.5

Initial failure intensity (λ_0) for a given software is 20 failures/ failure intensity objective (λ_F) of 1 failure/CPU hr. is to Assume the following resource usage parameters.

Resource Usage	Per hour	Per
Failure identification effort	2 Person hr.	1 Per
Failure Correction effort	0	5 Per
Computer time	1.5 CPU hr.	1 C

Software Reliability

(a) What resources must be expended to achieve th improvement? Use the logarithmic Poisson execution time failure intensity decay parameter of 0.025/failure.

(b) If the failure intensity objective is cut to half, what is t requirement of resources ?

Solution

(a) $$\Delta \quad = \frac{1}{\theta} Ln\left(\frac{\lambda_P}{\lambda_F}\right)$$

$$= \frac{1}{0.025} Ln\left(\frac{20}{1}\right) = 119 \text{ failures}$$

$$\Delta\tau = \frac{1}{\theta}\left(\frac{1}{\lambda_F} - \frac{1}{\lambda_P}\right)$$

$$= \frac{1}{0.025}\left(\frac{1}{1} - \frac{1}{20}\right) = \frac{1}{0.025}(1 - 0.05) = 38 \text{ CPU}$$

Software Reliability

Hence $$X_1 = {}_1\Delta \;\; + \theta_1 \Delta \tau$$

= 1 (119) + 2 (38) = 195 Person hrs.

$$X_F = {}_F\Delta$$

= 5 (119) = 595 Person hrs.

$$X_C = {}_c\Delta \;\; + \theta_c \Delta \tau$$

= 1 (119) + (1.5) (38) = 176 CPU hr.

(b) $\lambda_F = 0.5$ failures/CPU hr.

$$\Delta \quad = \frac{1}{0.025} Ln\left(\frac{20}{0.5}\right) = 148 \text{ failures}$$

$$\Delta\tau = \frac{1}{0.025}\left(\frac{1}{0.5} - \frac{1}{20}\right) = 78 \text{ CPU hr.}$$

So, X_I = 1 (148) + 2 (78) = 304 Person hrs.

X_F = 5 (148) = 740 Person hrs.

X_C = 1 (148) + (1.5)(78) = 265 CPU hrs.

Hence, if we cut failure intensity objective to half, resources r are not doubled but they are some what less. Note t approximately doubled but increases logarithmically. Thus, th increase will be between a logarithmic increase and a linear changes in failure intensity objective.

Example- 7.6

A program is expected to have 500 faults. It is also assumed t may lead to one failure only. The initial failure intensity was 2 hr. The program was to be released with a failure intensity o failures/100 CPU hr. Calculated the number of failure experie release.

Solution

The number of failure experienced during testing can be cal
the equation mentioned below:

$$\Delta \quad = \frac{V_0}{\lambda_0}\left(\lambda_P - \lambda_F\right)$$

Here $\quad V_0 = 500$ because one fault leads to one failu

$\lambda_0 = 2$ failures/CPU hr.

$\lambda_F = 5$ failures/100 CPU hr.

$= 0.05$ failures/CPU hr.

Software Reliability

So $$\Delta = \frac{500}{2}\left(2-0.05\right)$$

$= 487$ failures

Hence 13 faults are expected to remain at the release
the software.

Software Reliability

- The Jelinski-Moranda Model

$$\lambda(t) = \phi(N - i + 1)$$

where

ϕ = Constant of proportionality

N = Total number of errors present

I = number of errors found by time interval t_i

Software Reliability

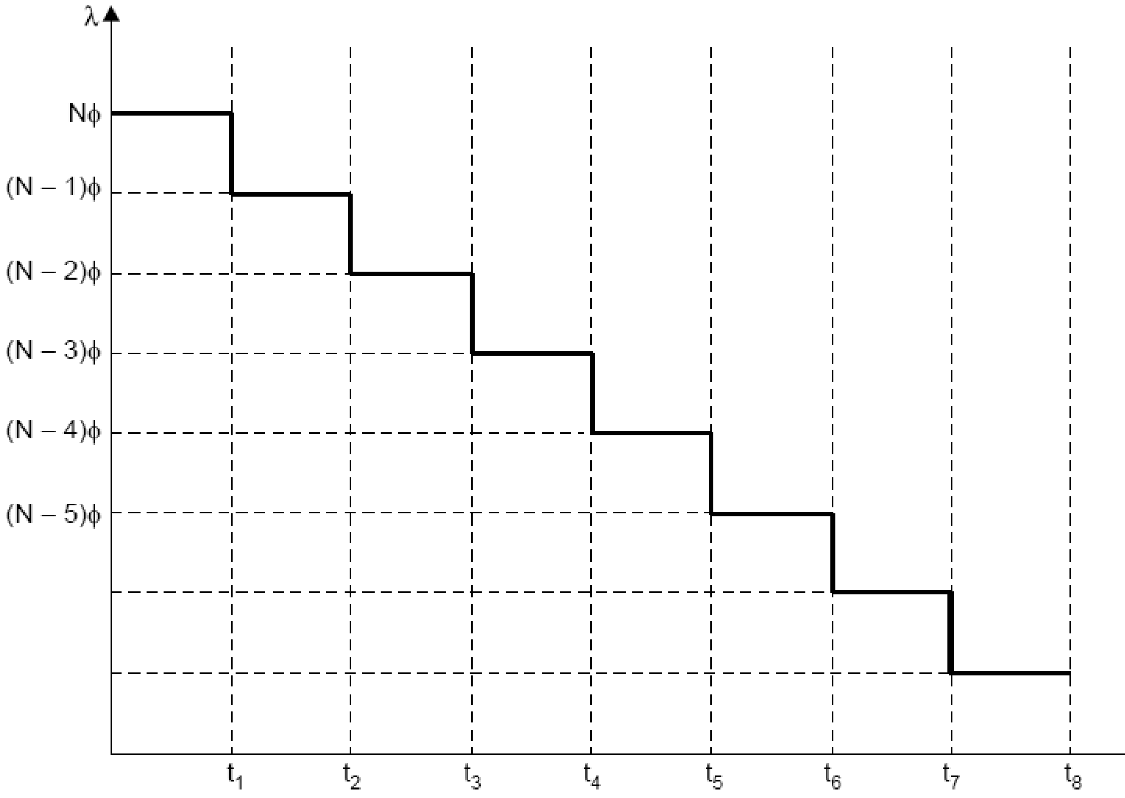


Fig.7.22: Relation between t & λ

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

Example- 7.7

There are 100 errors estimated to be present in a progra experienced 60 errors. Use Jelinski-Moranda model t failure intensity with a given value of ϕ=0.03. What wi intensity after the experience of 80 errors?

Software Reliability

Solution

N = 100 errors

i = 60 failures

ϕ = 0.03

We know $\lambda(t) = 0.03(100 - 60 + 1)$

= 0.03(100-60+1)

= 1.23 failures/CPU hr.

After 80 failures $\lambda(t) = 0.03(100 - 80 + 1)$

= 0.63 failures/CPU hr.

Hence, there is continuous decrease in the failure inte
number of failure experienced increases.

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Software Reliability

- ## The Bug Seeding Model

The bug seeding model is an outgrowth of a technic estimate the number of animals in a wild life population pond.

$$\frac{N_t}{N+N_t}=\frac{n_t}{n+n_t}$$

$$\hat{N}=\frac{n}{n_t}N_t$$

$$N=\frac{n}{n_s}N_s$$

Software Reliability

- ## Capability Maturity Model

It is a strategy for improving the software process, irresp actual life cycle model used.

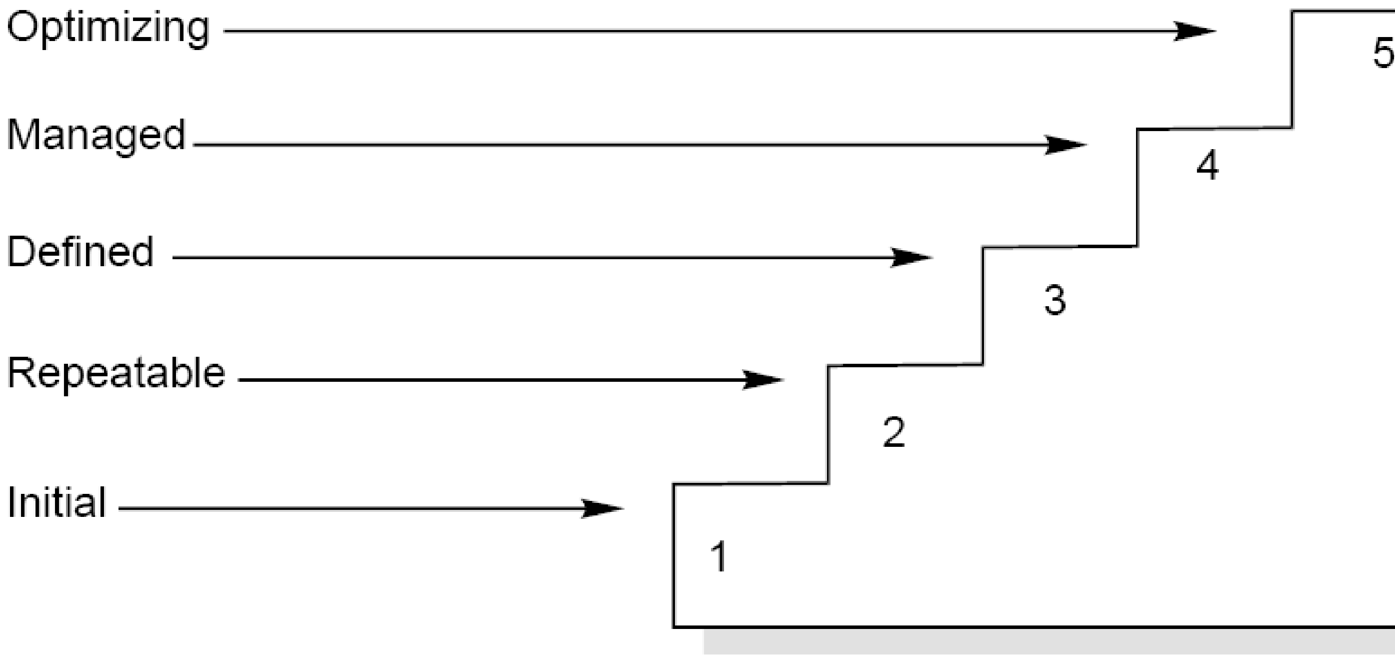


Fig.7.23: Maturity levels of CMM

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Maturity Levels:

- ✓ Initial (Maturity Level 1)
- ✓ Repeatable (Maturity Level 2)
- ✓ Defined (Maturity Level 3)
- ✓ Managed (Maturity Level 4)
- ✓ Optimizing (Maturity Level 5)

Software Reliability

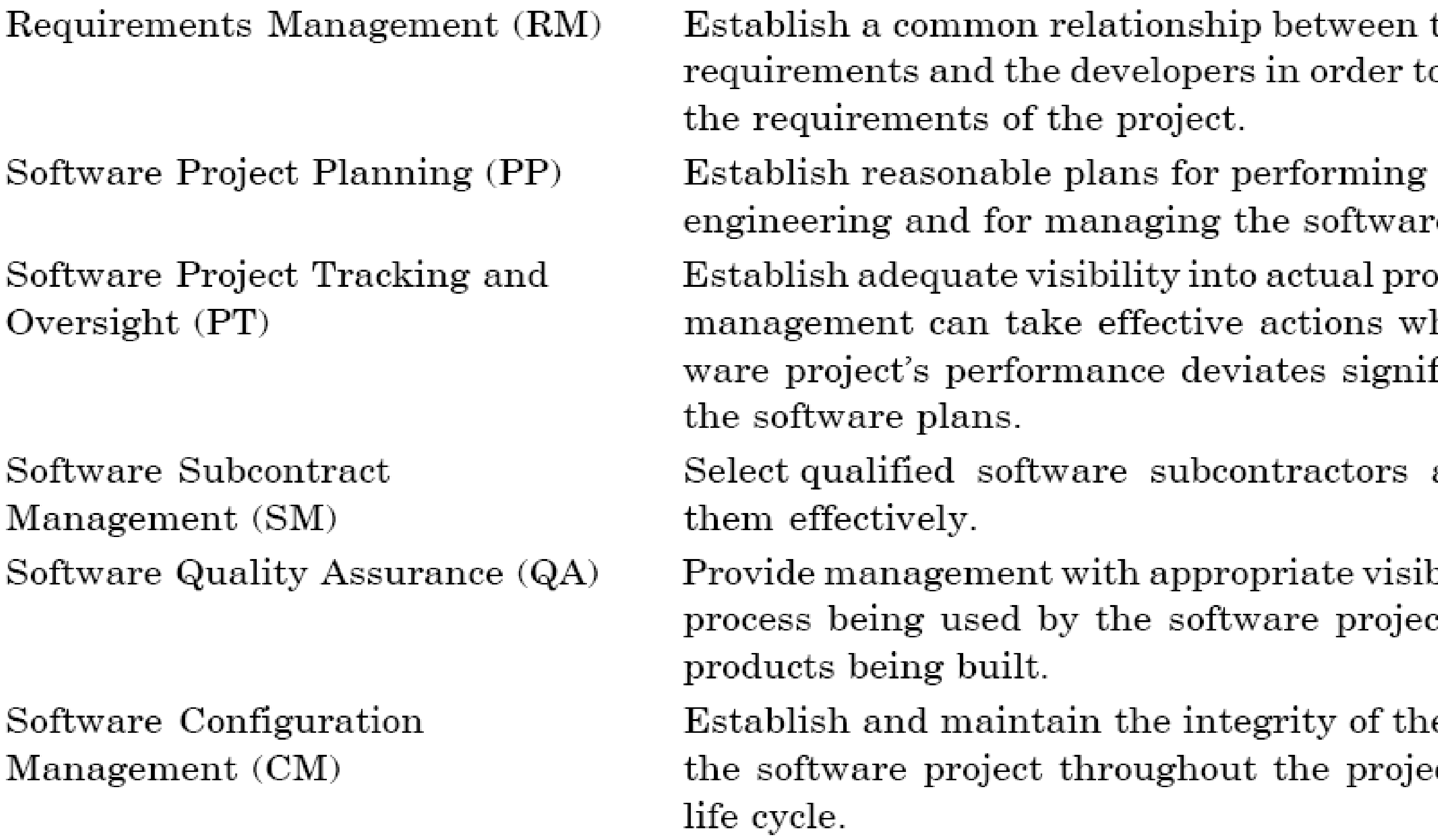
Maturity Level	Characterization
Initial	Adhoc Process
Repeatable	Basic Project Managemen
Defined	Process Definition
Managed	Process Measurement
Optimizing	Process Control

Fig.7.24: The five levels of CMM

Software Reliability

- ## Key Process Areas

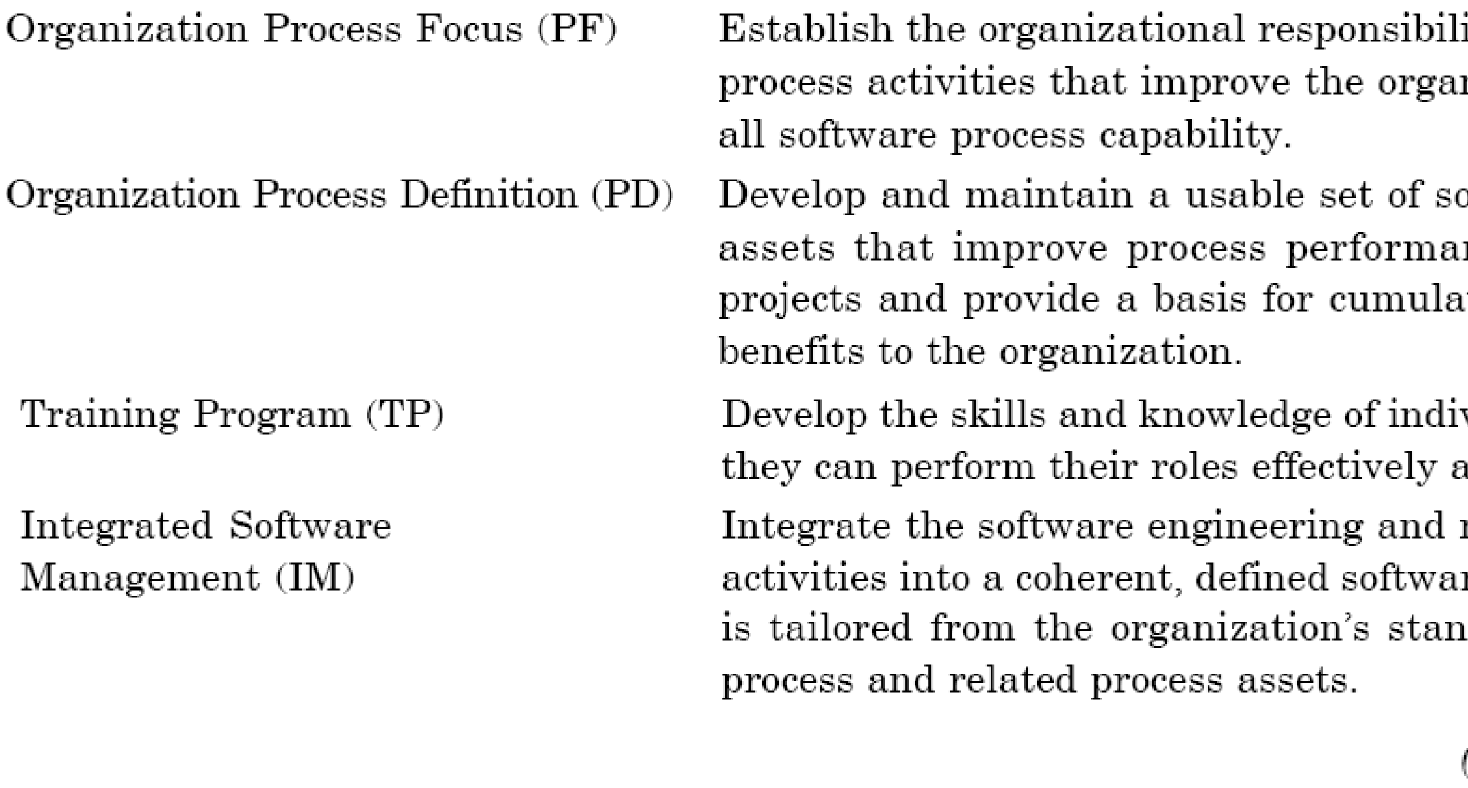
The key process areas at level 2 focus on the softwa
concerns related to establishing basic project managem
as summarized below:



Requirements Management (RM)	Establish a common relationship between t requirements and the developers in order to the requirements of the project.
Software Project Planning (PP)	Establish reasonable plans for performing engineering and for managing the softwar
Software Project Tracking and Oversight (PT)	Establish adequate visibility into actual pro management can take effective actions wh ware project's performance deviates signif the software plans.
Software Subcontract Management (SM)	Select qualified software subcontractors a them effectively.
Software Quality Assurance (QA)	Provide management with appropriate visib process being used by the software projec products being built.
Software Configuration Management (CM)	Establish and maintain the integrity of the the software project throughout the projec life cycle.

Software Reliability

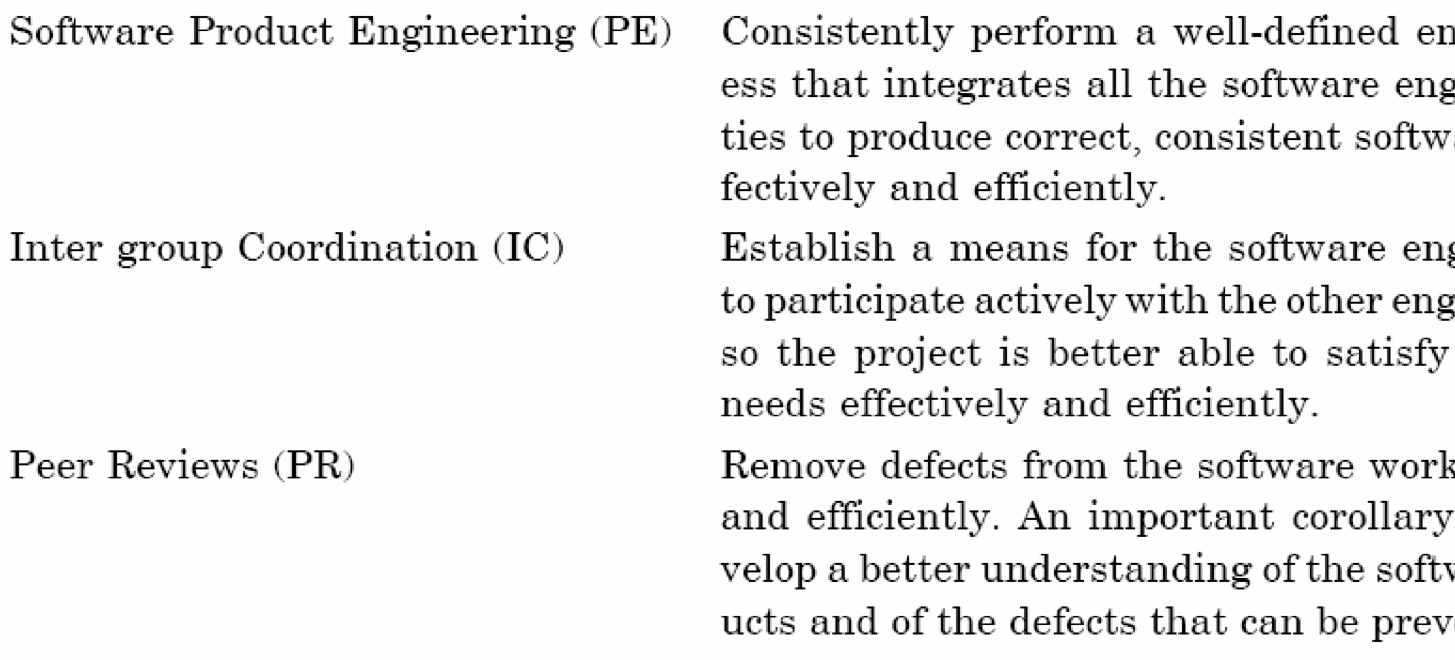
The key process areas at level 3 address both
organizational issues, as summarized below:



Organization Process Focus (PF)	Establish the organizational responsibili process activities that improve the orga all software process capability.
Organization Process Definition (PD)	Develop and maintain a usable set of so assets that improve process performa projects and provide a basis for cumula benefits to the organization.
Training Program (TP)	Develop the skills and knowledge of indiv they can perform their roles effectively a
Integrated Software Management (IM)	Integrate the software engineering and r activities into a coherent, defined softwa is tailored from the organization's stan process and related process assets.

(

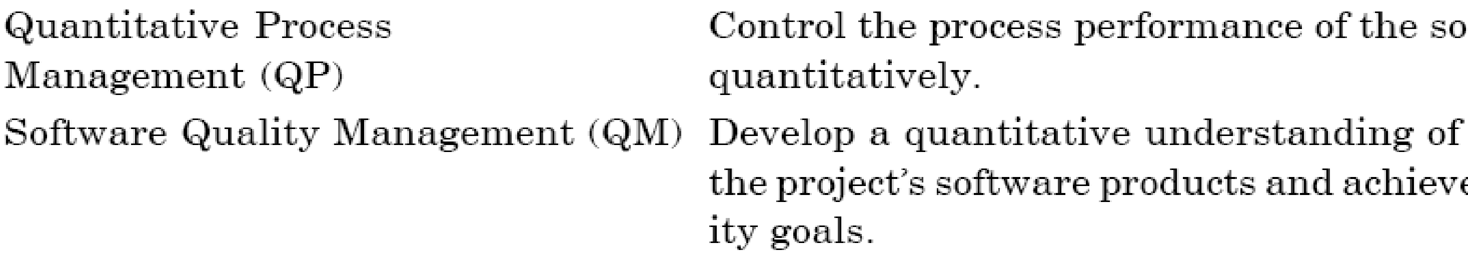
Software Reliability



Software Product Engineering (PE)	Consistently perform a well-defined en ess that integrates all the software eng ties to produce correct, consistent softw fectively and efficiently.
Inter group Coordination (IC)	Establish a means for the software eng to participate actively with the other eng so the project is better able to satisfy needs effectively and efficiently.
Peer Reviews (PR)	Remove defects from the software work and efficiently. An important corollary velop a better understanding of the softw ucts and of the defects that can be prev

Software Reliability

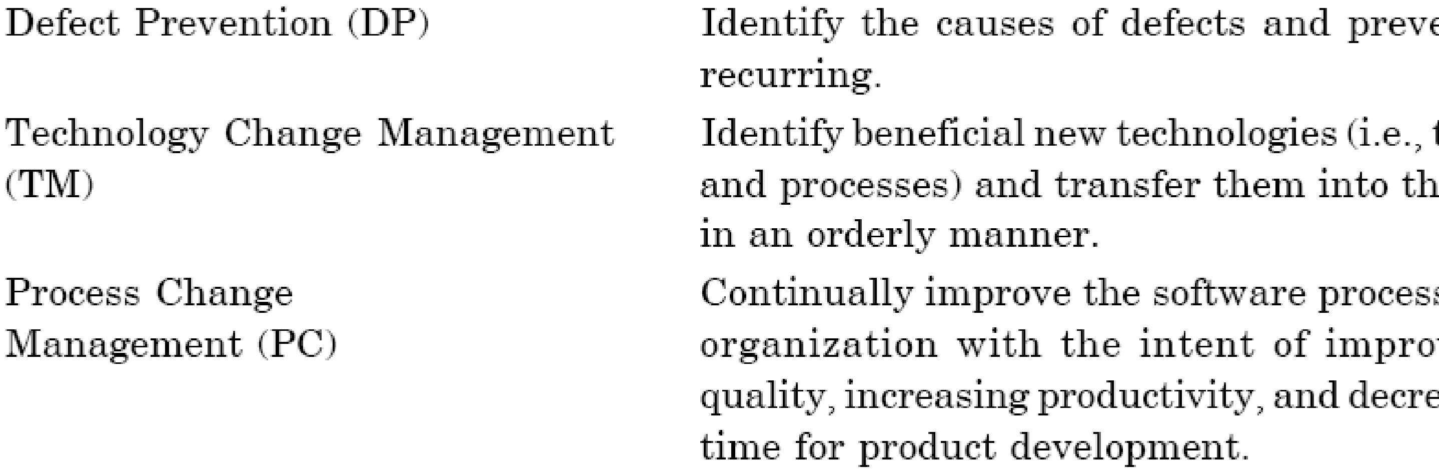
The key process areas at level 4 focus on establishing a understanding of both the software process and the so products being built, as summarized below:



Quantitative Process Management (QP)	Control the process performance of the so quantitatively.
Software Quality Management (QM)	Develop a quantitative understanding of the project's software products and achieve ity goals.

Software Reliability

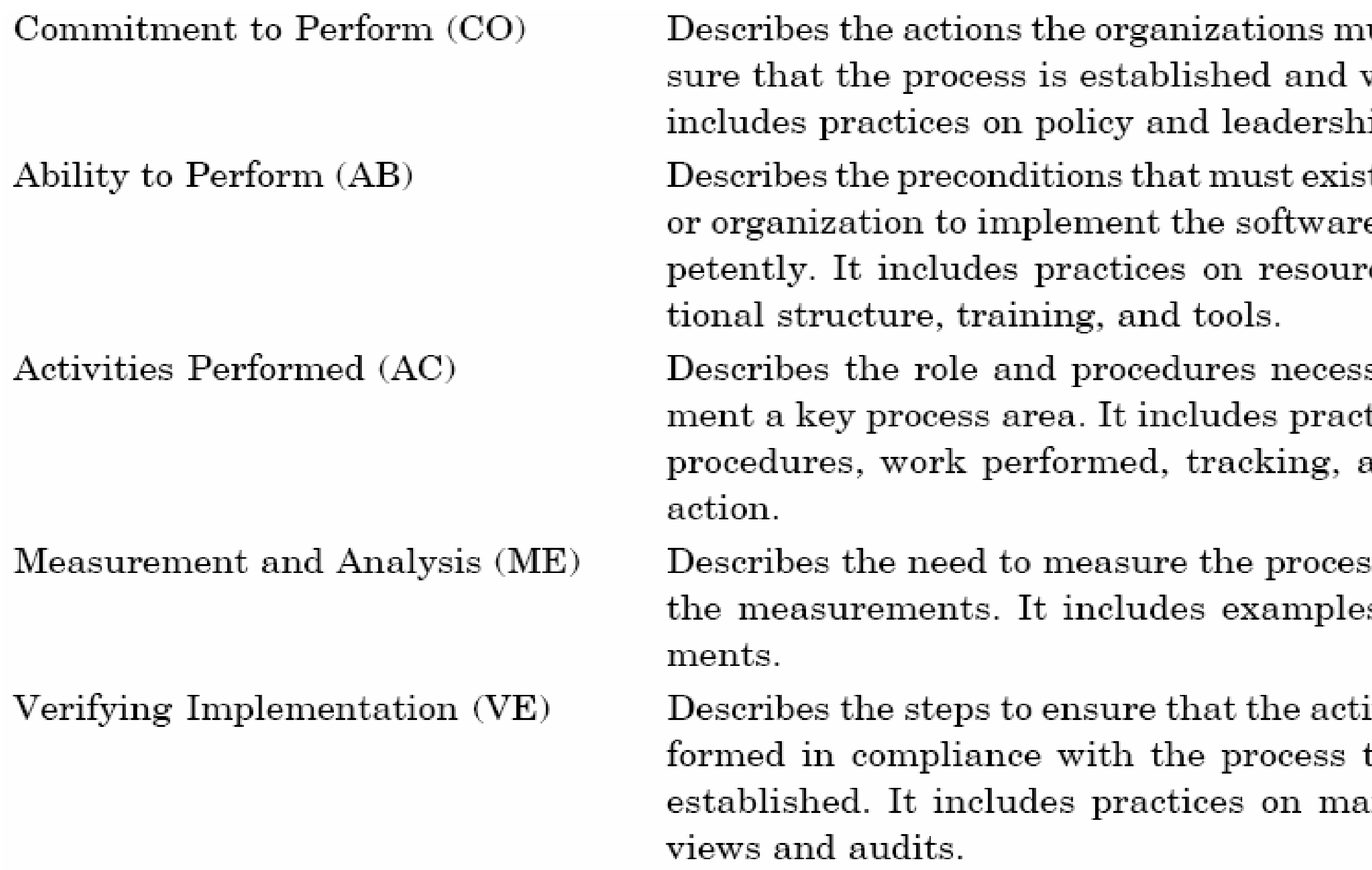
The key process areas at level 5 cover the issues th organization and the projects must address to implemen and measurable software process improvement, as below:



Defect Prevention (DP)	Identify the causes of defects and preve recurring.
Technology Change Management (TM)	Identify beneficial new technologies (i.e., t and processes) and transfer them into th in an orderly manner.
Process Change Management (PC)	Continually improve the software process organization with the intent of impro quality, increasing productivity, and decre time for product development.

Software Reliability

- ## Common Features



Commitment to Perform (CO)	Describes the actions the organizations mu sure that the process is established and w includes practices on policy and leadershi
Ability to Perform (AB)	Describes the preconditions that must exist or organization to implement the software petently. It includes practices on resour tional structure, training, and tools.
Activities Performed (AC)	Describes the role and procedures necess ment a key process area. It includes pract procedures, work performed, tracking, a action.
Measurement and Analysis (ME)	Describes the need to measure the proces the measurements. It includes examples ments.
Verifying Implementation (VE)	Describes the steps to ensure that the acti formed in compliance with the process t established. It includes practices on ma views and audits.

Software Reliability

- ISO 9000

The SEI capability maturity model initiative is an attemp software quality by improving the process by which developed.

ISO-9000 series of standards is a set of document quality systems that can be used for quality assuranc ISO-9000 series is not just software standard. It is a s related standards that are applicable to a wide variety activities, including design/ development, production, and servicing. Within the ISO 9000 Series, standard IS quality system is the standard that is most applicable development.

Software Reliability

- Mapping ISO 9001 to the CMM

1. Management responsibility
2. Quality system
3. Contract review
4. Design control
5. Document control
6. Purchasing
7. Purchaser-supplied product

Software Reliability

8. Product identification and traceability
9. Process control
10. Inspection and testing
11. Inspection, measuring and test equipment
12. Inspection and test status
13. Control of nonconforming product
14. Corrective action

Software Reliability

15. Handling, storage, packaging and delivery
16. Quality records
17. Internal quality audits
18. Training
19. Servicing
20. Statistical techniques

- Contrasting ISO 9001 and the CMM

There is a strong correlation between ISO 9001 anc although some issues in ISO 9001 are not covered in th some issues in the CMM are not addressed in ISO 9001.

The biggest difference, however, between these two d the emphasis of the CMM on continuous process improv

The biggest similarity is that for both the CMM and IS bottom line is **"Say what you do; do what you say"**.

Multiple Choice Questions

Note: Choose most appropriate answer of the following q

7.1 Which one is not a phase of "bath tub curve" of hardware reliabil

(a) Burn-in (b) Useful life

(c) Wear-out (d) Test-out

7.2 Software reliability is

(a) the probability of failure free operation of a program for a spec a specified environment

(b) the probability of failure of a program for a specified time in a environment

(c) the probability of success of a program for a specified time in environment

(d) None of the above

7.3 Fault is

(a) Defect in the program (b) Mistake in the progr

(c) Error in the program (d) All of the above

7.4 One fault may lead to

(a) one failure (b) two failures

(c) many failures (d) all of the above

Multiple Choice Questions

7.5 Which ‘time’ unit is not used in reliability studies

(a) Execution time (b) Machine time

(c) Clock time (d) Calendar time

7.6 Failure occurrences can be represented as

(a) time to failure (b) time interval betwe

(c) failures experienced in a time interval (d) All of the above

7.7 Maximum possible value of reliability is

(a) 100 (b) 10

(c) 1 (d) 0

7.8 Minimum possible value of reliability is

(a) 100 (b) 10

(c) 1 (d) 0

7.9 As the reliability increases, failure intensity

(a) decreases (b) increases

(c) no effect (d) None of the above

Multiple Choice Questions

7.10 If failure intensity is 0.005 failures/hour during 10 hours of software, its reliability can be expressed as

(a) 0.10 (b) 0.92

(c) 0.95 (d) 0.98

7.11 Software Quality is

(a) Conformance to requirements (b) Fitness for the purpo

(c) Level of satisfaction (d) All of the above

7.12 Defect rate is

(a) number of defects per million lines of source code

(b) number of defects per function point

(c) number of defects per unit of size of software

(d) All of the above

7.13 How many product quality factors have been proposed in McCal

(a) 2 (b) 3

(c) 11 (d) 6

Multiple Choice Questions

7.14 Which one is not a product quality factor of McCall quality mod

(a) Product revision (b) Product operation

(c) Product specification (d) Product transition

7.15 The second level of quality attributes in McCall quality model ar

(a) quality criteria (b) quality factors

(c) quality guidelines (d) quality specificatio

7.16 Which one is not a level in Boehm software quality model ?

(a) Primary uses (b) Intermediate constru

(c) Primitive constructs (d) Final constructs

7.17 Which one is not a software quality model?

(a) McCall model (b) Boehm model

(c) ISO 9000 (d) ISO 9126

7.18 Basic execution time model was developed by

(a) Bev.Littlewood (b) J.D.Musa

(c) R.Pressman (d) Victor Baisili

Multiple Choice Questions

7.19 NHPP stands for

(a) Non Homogeneous Poisson Process (b) Non Hetrogeneous I

(c) Non Homogeneous Poisson Product (d) Non Hetrogeneous I

7.20 In Basic execution time model, failure intensity is given by

$(a)\ \lambda(\ \ ) = \lambda_0\left(1 - \frac{{}^{2}}{V_0}\right)$ $(b)\ \lambda(\ \ ) = \lambda_0\left(1 - \frac{\ }{V_0}\right)$

$(c)\ \lambda(\ \ ) = \lambda_0\left(1 - \frac{V_0}{\ }\right)$ $(d)\ \lambda(\ \ ) = \lambda_0\left(1 - \frac{V_0}{2}\right)$

7.21 In Basic execution time model, additional number of failure achieve a failure intensity objective $(\Delta\ \ )$ is expressed as

$(a)\ \Delta\ \ = \frac{V_0}{\lambda_0}(\lambda_P - \lambda_F)$ $(b)\ \Delta\ \ = \frac{V_0}{\lambda_0}(\lambda_F - \lambda_P)$

$(c)\ \Delta\ \ = \frac{\lambda_0}{V_0}(\lambda_F - \lambda_P)$ $(d)\ \Delta\ \ = \frac{\lambda_0}{V_0}(\lambda_P - \lambda_F)$

Multiple Choice Questions

7.22 In Basic execution time model, additional time required to ach
intensity objective $(\Delta\tau)$ is given as

$(a)\ \Delta\tau = \dfrac{\lambda_0}{V_0} Ln\left(\dfrac{\lambda_F}{\lambda_P}\right)$ $(b)\ \Delta\tau = \dfrac{\lambda_0}{V_0} Ln\left(\dfrac{\lambda_P}{\lambda_F}\right)$

$(c)\ \Delta\tau = \dfrac{V_0}{\lambda_0} Ln\left(\dfrac{\lambda_F}{\lambda_P}\right)$ $(d)\ \Delta\tau = \dfrac{V_0}{\lambda_0} Ln\left(\dfrac{\lambda_P}{\lambda_F}\right)$

7.23 Failure intensity function of Logarithmic Poisson execution mod

$(a)\ \lambda(\quad) = \lambda_0 LN(-\theta\quad)$ $(b)\ \lambda(\quad) = \lambda_0 \exp(\theta\quad)$

$(c)\ \lambda(\quad) = \lambda_0 \exp(-\theta\quad)$ $(d)\ \lambda(\quad) = \lambda_0 \log(-\theta\quad)$

7.24 In Logarithmic Poisson execution model, ‘θ’ is known as

(a) Failure intensity function parameter (b) Failure intensity dec

(c) Failure intensity measurement (d) Failure intensity inc

Multiple Choice Questions

7.25 In jelinski-Moranda model, failure intensity is defined aseneous
Product

(a) $\lambda(t) = \phi(N - i + 1)$ (b) $\lambda(t) = \phi(N + i + 1)$

(c) $\lambda(t) = \phi(N + i - 1)$ (d) $\lambda(t) = \phi(N - i - 1)$

7.26 CMM level 1 has

(a) 6 KPAs (b) 2 KPAs

(c) 0 KPAs (d) None of the above

7.27 MTBF stands for

(a) Mean time between failure (b) Maximum time betw

(c) Minimum time between failures (d) Many time between

7.28 CMM model is a technique to

(a) Improve the software process (b) Automatically devel

(c) Test the software (d) All of the above

7.29 Total number of maturing levels in CMM are

(a) 1 (b) 3

(c) 5 (d) 7

Multiple Choice Questions

7.30 Reliability of a software is dependent on number of errors

(a) removed (b) remaining

(c) both (a) & (b) (d) None of the above

7.31 Reliability of software is usually estimated at

(a) Analysis phase (b) Design phase

(c) Coding phase (d) Testing phase

7.32 CMM stands for

(a) Capacity maturity model (b) Capability maturity

(c) Cost management model (d) Comprehensive ma

7.33 Which level of CMM is for basic project management?

(a) Initial (b) Repeatable

(c) Defined (d) Managed

7.34 Which level of CMM is for process management?

(a) Initial (b) Repeatable

(c) Defined (d) Optimizing

Multiple Choice Questions

7.35 Which level of CMM is for process management?

(a) Initial (b) Defined

(c) Managed (d) Optimizing

7.36 CMM was developed at

(a) Harvard University (b) Cambridge Universi

(c) Carnegie Mellon University (d) Maryland Universit

7.37 McCall has developed a

(a) Quality model (b) Process improvement

(c) Requirement model (d) Design model

7.38 The model to measure the software process improvement is calle

(a) ISO 9000 (b) ISO 9126

(c) CMM (d) Spiral model

7.39 The number of clauses used in ISO 9001 are

(a) 15 (b) 25

(c) 20 (d) 10

Multiple Choice Questions

7.40 ISO 9126 contains definitions of

(a) quality characteristics (b) quality factors

(c) quality attributes (d) All of the above

7.41 In ISO 9126, each characteristics is related to

(a) one attributes (b) two attributes

(c) three attributes (d) four attributes

7.42 In McCall quality model; product revision quality factor consist

(a) Maintainability (b) Flexibility

(c) Testability (d) None of the above

7.43 Which is not a software reliability model ?

(a) The Jelinski-Moranda Model (b) Basic execution tim

(c) Spiral model (d) None of the above

7.44 Each maturity model is CMM has

(a) One KPA (b) Equal KPAs

(c) Several KPAs (d) no KPA

Multiple Choice Questions

7.45 KPA in CMM stands for

(a) Key Process Area (b) Key Product Area

(c) Key Principal Area (d) Key Performance Ar

7.46 In reliability models, our emphasis is on

(a) errors (b) faults

(c) failures (d) bugs

7.47 Software does not break or wear out like hardware. What is your

(a) True (b) False

(c) Can not say (d) not fixed

7.48 Software reliability is defined with respect to

(a) time (b) speed

(c) quality (d) None of the above

7.49 MTTF stands for

(a) Mean time to failure (b) Maximum time to fa

(c) Minimum time to failure (d) None of the above

Multiple Choice Questions

7.50 ISO 9000 is a series of standards for quality management system

(a) 2 related standards (b) 5 related standards

(c) 10 related standards (d) 25 related standards

Exercises

7.1 What is software reliability? Does it exist?

7.2 Explain the significance of bath tube curve of reliability w
a diagram.

7.3 Compare hardware reliability with software reliability.

7.4 What is software failure? How is it related with a fault?

7.5 Discuss the various ways of characterising failure occ
respect to time.

7.6 Describe the following terms:

(i) Operational profile (ii) Input space
(iii) MTBF (iv) MTTF
(v) Failure intensity.

Exercises

7.7 What are uses of reliability studies? How can one use softw measures to monitor the operational performance of softwa

7.8 What is software quality? Discuss software quality attribute

7.9 What do you mean by software quality standards? Illustrate as well as benefits.

7.10 Describe the McCall software quality model. How many p factors are defined and why?

7.11 Discuss the relationship between quality factors and qua McCall's software quality model.

7.12 Explain the Boehm software quality model with the he diagram.

7.13 What is ISO9126 ? What are the quality characteristics anc

Exercises

7.14 Compare the ISO9126 with McCall software qualit highlight few advantages of ISO9126.

7.15 Discuss the basic model of software reliability. How Δ calculated.

7.16 Assume that the initial failure intensity is 6 failures/CPU intensity decay parameter is 0.02/failure. We assume th have been experienced. Calculate the current failure intensi

7.17 Explain the basic & logarithmic Poisson model and their reliability studies.

Exercises

7.18 Assume that a program will experience 150 failures in i has now experienced 80. The initial failure intensity was 1 hr.

(i) Determine the current failure intensity

(ii) Calculate the failures experienced and failure intensit 40 CPU hrs. of execution.

(iii) Compute additional failures and additional execution to reach the failure intensity objective of 2 failures/CPU hr.

Use the basic execution time model for the abov calculations.

7.19 Write a short note on Logarithmic Poisson Execution tim can we calculate Δ & $\Delta\tau$?

7.20 Assume that the initial failure intensity is 10 failures/ failure intensity decay parameter is 0.03/failure. We have e failures upto this time. Find the failures experienced and fa after 25 and 50 CPU hrs. of execution.

Exercises

7.21 The following parameters for basic and logarithmic Poiss given:

Basic execution time model	*Logarithmic Poisson execution time model*
λ_0 = 5 failures/CPU hr	λ_0 = 25 failures/CPU hr
V_0 = 125 failures	θ = 0.3/failure

Determine the additional failures and additional execution to reach the failure intensity objective of 0.1 failure/CPU models.

7.22 Quality and reliability are related concepts but are different in a number of ways. Discuss them.

7.23 Discuss the calendar time component model. Establish th between calendar time to execution time.

Exercises

7.24 A program is expected to have 250 faults. It is also assu fault may lead to one failure. The initial failure intensity is hr. The program is released with a failure intensity o failures/10 CPU hr. Calculate the number of failures expe release.

7.25 Explain the Jelinski-Moranda model of reliability theory relation between ‘t’ and $'\lambda'$?

7.26 Describe the Mill’s bug seeding model. Discuss few adva model over other reliability models.

7.27 Explain how the CMM encourages continuous impro software process.

7.28 Discuss various key process areas of CMM at various matu

7.29 Construct a table that correlates key process areas (KPAs with ISO9000.

7.30 Discuss the 20 clauses of ISO9001 and compare with the p CMM.

Exercises

7.31 List the difference of CMM and ISO9001. Why is it s
CMM is the better choice than ISO9001?

7.32 Explain the significance of software reliability engineerin
advantage of using any software standard for software deve

7.33 What are the various key process areas at defined lev
Describe activities associated with one key process area.

7.34 Discuss main requirements of ISO9001 and compare
capability maturity model.

7.35 Discuss the relative merits of ISO9001 certification and t
based evaluation. Point out some of the shortcomings of
certification process as applied to the software industry.