© ISO/IEC

Table 7.3.4 Attractiveness metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Attractive interface	How attractive is the interface to the user?	Questionnaire to users	Questionnaire to assess the attractiveness of the interface to users, after experience of usage						
Interface appearance customisability	What proportion of interface elements can be customised in appearance to the user's satisfaction?	User test	X=A/B A=Number of interface elements customised in appearance to user's satisfaction B=Number of interface elements that the user wishes to customise NOTE: This metric is generally used as one of experienced and justified.	0 <= X <= 1 The closer to 1, the better.	Absolute.	X= Count/ Count A= count B= count			
User operational frequency	Does user like to use software frequently?	Observation of usage	User's operational frequency X = N / OT N= number of turns which user use the specific software functions OT= operation time NOTE 1. User may be enforced to use the software frequently whether user likes it or not 2. It is recommended to ensure whether user actually feels good and is in favor of using the software by interviewing end user.	0<X The more is the better.	Ratio	N= Count, OT= Time X= Count/ Time	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer

NOTE:
The following complementary metrics may be used.
1) Attractive interaction
Frequency of user's feeling attractive
$X = A / T$
Ratio of attractive presentation
$Y = (A / B)$

A= Number of turns which user feel attractive during operation
B= Number of turns which user is encountering presentations
T= User operating time (observation period)
$0 <= X$ The larger is the better.
$0 <= Y <= 1$ The closer to 1.0 is the better.

2) Favorable input/output expression selection availability
Favorable input/output expression selection availability $X = 1 - (A / N)$

A= Number of turns which user failed to select input/output expression
N= Number of turns which user tried to select input/output expression

Input/output expression may include multiple modal expressions such are fill-in-the-box, button, list-box, text, graph, map, pictogram, color, visual picture, voice recognition/reading and so on.
$0 <= X <= 1$ The closer to 1.0 is the better.

3) Favorable user popularity
Favorable user popularity
$X = A / B * 100(\%)$
A= Number of people who replied favorable answer
B= Number of people who answered questionnaires

$0 <= X <= 100(\%)$ The closer to 100% is the better.

Make questionnaires to user who once operate the software and analyse the answers statistically.

Table 7.3.5 Compliance metrics (compliance for usability)

Metric Name	Purpose	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Satisfaction coverage of compliance items relating to usability	How completely does the software adhere the standards, conventions, style guides or regulations relating to usability?	Previously specify required compliance items based on standards, conventions, style guides or regulations relating to usability which to be adhered by software. Design test cases in accordance with compliance items. Conduct functional testing for these test cases.	Ratio of satisfied compliance items relating to usability $X = 1 - (A / B)$ A= Number of failed compliance items during testing B= Number of total compliance items NOTE: It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied or not.	$0 <= X <= 1$ The closer to 1 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Specification of compliance and related standards, conventions, style guides or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

7.4 Efficiency metrics

An external efficiency metric should be able to measure such attribute as the behavior of computer system including software during testing or operating.

It is recommended that the maximal and distribution time are to be investigated for many cases of testing or operating, because the measure is affected strongly and fluctuated by condition of use, such are load of processing data, frequency of use, number of connecting sites and so on. Therefore, efficiency metrics may include the ratio of measured actual value with error fluctuation to the designed value with allowed error fluctuation range which are required in specification.

It is recommended to list and to investigate the role played by factors such as "CPU" and memory load by other software's, network traffic, scheduled background processes. Possible fluctuations and valid ranges for measured values should be established and compared to requirement specifications."

It is recommended that a task is identified and defined to be suitable for software application, for examples: a transaction as a task for business application; a switching or data packet sending as a task for communication application; an event control as a task for control application; an output of data produced by user callable function for common user application.

NOTE:
1. Response time: Time needed to get the result from pressing a transmissionkey. This means that response time includes processing time andtransmission time. Response time is applicable only for a interactive system. There isno significant difference when it is a standalone system. However, in the case of Internet system or other real time system, sometimes transmission time is much longer.

2. Processing time: The elapsed time in a computer after receiving a message to sending the result. Sometimes it includes operating overhead time, other times it only means time used for an application program.

3. Turn around time: Time needed to get the result from request. In many cases one turn around time includes many response. For example, in a case of banking cash dispenser, turn around time is a time from pressing initial key until you get money, mean while you must select type of transaction and wait a message, input password and wait next message etc.

7.4.1 Time behavior metrics

An external time behavior metric should be able to measure such attribute as the time behavior of computer system including software during testing or operating.

© ISO/II

7.4.2 Resource utilization metrics

An external resource utilization metric should be able to measure such attribute as the utilized resources behavior of computer system including software during testing or operating.

7.4.3 Compliance metrics

An external efficiency compliance metric should be able to measure an attribute such as the number of functions with, or occurrences of compliance problems, which is of the software product to failing to adhere to standards, conventions or regulations relating to efficiency which are required to be adhered.

Table 7.4.1 Time behavior metrics

Response time

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Response time	What is the wait time the user experiences when issuing a request and for the application to finish the process? How long does it take for users to wait for next response from the software?	Start a specified task. Measure the time it takes for the sample to complete its operation. Keep a record of each attempt.	Response time T = (time of gaining the result) - (time of command entry finished) NOTE: It is recommended to take account of time bandwidth and to use statistical analysis with measures for a lot of tasks (sample shots) not for only one task.	0 < T The sooner is the better.	Ratio	T= Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Mean response fulfillment ratio	What is the average wait time the user experiences once issuing a request and its completion within the specified load upon the system in terms of concurrent tasks and system utilisation?	Execute a number of concurrent tasks (or sample shots of these tasks). Measure the time it takes to complete the selected operation(s) in the given traffic. Keep a record of each attempt.	Mean response time satisfaction X = Tmean / TXmean Tmean = Σ(Ti) / N, (for i=1 to N) TXmean = required mean response time Ti= response time for i-th evaluation (shot) N= number of evaluations (sampled shots) NOTE: Required mean response time can be derived from specification of required real-time processing, user expectation of business needs or observation of user reaction. A user cognitive aspect of human ergonomics may be considerable.	0 <= X The nearer to 1 and less than 1 is the better.	Absolute	X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

Table 7.4.1 Time behavior metrics (Continued)

Response time

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Worst case response time ratio	What is the absolute limit on time required in fulfilling a function? In the worst case, can user still get response within the specified time limit? In the worst case, can user still get reply from the software within enough short time to be tolerable for user?	Calibrate the test. Emulate a condition whereby the system reaches a maximum load situation. Run application and monitor result(s)	X= Tmax / Rmax Tmax= MAX(Ti) (for i=1 to N) Rmax = required maximum response time MAX(Ti)= maximum response time among evaluations N= number of evaluations (sampled shots) Ti= response time for i-th evaluation (shot) NOTE: 1. Distribution may be measured like below. Statistical maximal ratio Y= Tdev / Rmax Tdev = Tmean + K (DEV) Tdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation. K: coefficient (2 or 3) DEV=SQRT{ Σ((Ti-Tmean) **2) / (N-1)} (for i=1 to N) Tmean = Σ(Ti) / N. (for i=1 to N) TXmean = required mean response time	0 < X The nearer to 1 and less than 1 is the better.	Absolute	X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA

Table 7.4.1 Time behavior metrics (Continued)

Throughput

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Throughput time	How many tasks can be successfully performed over a given period of time?	Calibrate each task according to the intended priority given. Start several job tasks. Measure the time it takes for the measured task to complete its operation. Keep a record of each attempt.	X = A / B A = number of completed tasks B = observation time period	0 < X The larger is the better.	Ratio	X= Count/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA
Mean throughput fulfillment ratio	What is the average number of concurrent tasks the system can handle over a set unit of time?	Calibrate each task according to intended priority. Execute a number of concurrent tasks. Measure the time it takes to complete the selected task in the given traffic. Keep a record of each attempt.	X = Xmean / Rmean Xmean = Σ(Xi)/N Rmean = required mean throughput Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N = number of evaluations	0 < X The larger is the better.	Absolute	X = Time / Time	Testing report Operation report showing elapse time	5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA

 © ISO/IEC

Table 7.4.1 Time behavior metrics (Continued)

Throughput

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Worst case throughput ratio	What is the absolute limit on the system in terms of the number and handling of concurrent tasks as throughput?	Calibrate the test. Emulate the condition whereby the system reaches a situation of maximum load. Run job tasks concurrently and monitor result(s).	X = Xmax / Rmax Tmax = MAX(Xi) (for i = 1 to N) Rmax = required maximum throughput. MAX(Xi) = number of job tasks for i-th evaluation. Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N= number of evaluations. NOTE: 1. Distribution may be measured like below. Statistical maximal ratio Y= Xdev / Xmax Xdev = Xmean + K (DEV) Xdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation. K: coefficient (2 or 3) DEV=SQRT{ Σ((Xi-Xmean) **2) / (N-1)} (for i=1 to N) Xmean = Σ(Xi)/N	0 < X The larger is the better.	Absolute	X = Time / Time	Testing report Operation report showing elapse time	5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

Table 7.4.1 Time behavior metrics (Continued)

Turnaround time

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Turnaround time	What is the wait time the user experiences when issuing an instruction to start a group of related tasks and the completion of these related tasks?	Calibrate the test accordingly. Start the job task. Measure the time it takes for the job task to complete its operation. Keep a record of each attempt.	T = Time between user's finishing getting output results and user's finishing request. NOTE: It is recommended to take account of time bandwidth and to use statistical analysis with measures for a lot of tasks (sample shots) not for only one task (shot).	0 < T The shorter the better.	Ratio	T= Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA
Mean turnaround fulfillment ratio	What is the average wait time the user experiences when issuing an instruction to start a group of related tasks and the completion within a specified load the system has in terms of concurrent tasks and system utilisation?	Calibrate the test. Emulate a condition where a load is placed on the system by executing a number of concurrent tasks (sampled shots). Measure the time it takes to complete the selected job task in the given traffic. Keep a record of each attempt.	X = Tmean/TXmean Tmean = Σ(Ti)/N, (for i=1 to N) TXmean = required mean turnaround time Ti = turnaround time for i-th evaluation (shot) N = number of evaluations (sampled shots)	0 < X The shorter is the better.	Absolute	X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA

© ISO/IEC

Table 7.4.1 Time behavior metrics (Continued)

Turnaround time

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Worst case turnaround time ratio	What is the absolute limit on time required in fulfilling a job task? In the worst case, how long does it take for software system to perform specified tasks?	Calibrate the test. Emulate a condition where by the system reaches maximum load in terms of tasks performed. Run the selected job task and monitor result(s).	X= Tmax / Rmax Tmax= MAX(Ti) (for i=1 to N) Rmax = required maximum turnaround time MAX(Ti)= maximum turnaround time among evaluations N= number of evaluations (sampled shots) Ti= turnaround time for i-th evaluation (shot) NOTE: 1. Distribution may be measured like below. Statistical maximal ratio Y= Tdev / Rmax Tdev = Tmean + K (DEV) Tdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation. K: coefficient (2 or 3) DEV=SQRT{ Σ((Ti-Tmean) **2) / (N-1)} (for i=1 to N) Tmean = Σ(Ti) / N, (for i=1 to N) TXmean = required mean turnaround time	0 < X The nearer to 1 and less than 1 is the better.	Absolute	X= Time/ Time Tmax = Time Tdev = Time	Testing report Operation report showing elapse time	5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

Table 7.4.2 Resource utilisation metrics

I/O devices resource utilization

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
I/O devices utilisation satisfaction	Is software system capable to perform tasks without using I/O devices long time?	Execute concurrently a lot of tasks and observe time of I/O devices occupied.	I/O devices utilisation X = A / B A = time of I/O devices occupied B = specified time which is designed to occupy I/O devices	0 <= X <= 1 The less than and nearer to the 1 is the better.	Absolute	A= Time B= Time X= Time/ Time	Testing report Operation report	5.3 Qualification testing 5.4 Operation Maintenance	Developer Maintainer SQA
Mean I/O fulfillment ratio	What is the average number of I/O related error messages and failures over a specified length of time and specified utilisation?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to I/O failure and warnings.	X = Amean / Rmean Amean = Σ(Ai)/N Rmean = required mean number of I/O messages. Ai = number of I/O messages for ith evaluation N = number of evaluations	0<= X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA
User waiting time of I/O devices utilisation	Can user perform tasks without waiting for finish of I/O devices operation?	Execute concurrently a lot of tasks and observe time of I/O devices occupied with user environment.	User waiting time of I/O devices utilisation T T = Time spent to wait for finish of I/O devices operation NOTE: It is recommended that the maximal and distributed time are to be investigated for several cases of testing or operating, because the measures are tend to be fluctuated by condition of use.	0 < T The shorter is the better.	Ratio	T= Time	Testing report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA

Table 7.4.2 Resource utilisation metrics (continued)

I/O devices resource utilization

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Visible I/O utilisation	How many I/O errors were experienced over a set period of time and specified resource utilisation?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum I/O load. Run the application and record number of errors due to I/O failure and warnings.	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA
Worst case I/O utilisation	What is the absolute limit on I/O required in fulfilling a function?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s).	X = Amax / Rmax Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum I/O messages MAX(Ai) = Maximum number of I/O messages from 1st to i-th evaluation. N= number of evaluations.	0<= X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

Table 7.4.2 Resource utilisation metrics (continued)

Memory resource utilization

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Mean memory fulfillment ratio	What is the average number of memory related error messages and failures over a specified length of time and a specified load on the system?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.	X = Amean / Rmean Amean = Σ(Ai)/N Rmean = required mean number of memory messages. Ai = number of memory messages for i-th evaluation N = number of evaluations	0<= X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Visible memory utilisation	How many memory errors were experienced over a set period of time and specified resource utilisation?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA
Worst case memory utilisation	What is the absolute limit on memory required in fulfilling a function?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s)	X = Amax / Rmax Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum memory messages MAX(Ai) = Maximum number of memory messages from 1st to i-th evaluation. N= number of evaluations.	0<= X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

© ISO/IEC

Table 7.4.2 Resource utilisation metrics (continued)

Transmission resource utilization

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficceries
Mean transmission fulfillment ratio	What is the average number of transmission related error messages and failures over a specified length of time and specified utilisation?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to Transmission failure and warnings.	X = Amean / Rmean Amean = Σ(Ai)/N Rmean = required mean number of transmission related error messages and failures Ai = Number of transmission related error messages and failures for i-th evaluation N = number of evaluations	0<= X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA
Transmission capacity utilisation satisfaction	Is software system capable to perform tasks within expected transmission capacity?	Execute concurrently specified tasks with multiple users, observe transmission capacity and compare specified one.	Transmission capacity utilisation satisfaction X = A / B A = transmission capacity B = specified transmission capacity which is designed to be used by the software during execution. NOTE: It is recommended to measure dynamically peaked value with multiple users.	0 <= X <= 1 The less than and nearer to the 1 is the better.	Absolute	A= Size B= Size X= Size / Size	Testing report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer SQA
Visible transmission utilisation	How many transmission error messages were experienced over a set period of time and specified resource utilisation?	Calibrate the test conditions. Emulate a condition whereby the system to reaches a situation of maximum transmission load. Run the application and record number of errors due to transmission failure and warnings.	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Maintainer SQA

Table 7.4.2 Resource utilisation metrics (continued)

Transmission resource utilization

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Worst case transmission utilisation	What is the absolute limit on transmission required in fulfil a function?	Evaluate what is required for the system to reach a situation of maximum load. Emulate this condition. Run application and monitor result(s)	X = Amax / Rmax Amax = MAX(Ai). (for i = 1 to N) Rmax = required maximum number of transmission related error messages and failures MAX(Ai) = Maximum number of transmission related error messages and failures from 1st to i-th evaluation. N= number of evaluations.	0<=X The smaller is the better	Absolute	X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Developer Maintainer SQA
Visible synchronisation	What is the degree of synchronisation between dissimilar media over a set period of time?	Calibrate the test conditions. Emulate a condition whereby the system to reaches a situation of maximum transmission load. Run the application and record the delay in the processing of different media types.	X = SyncTime/T SyncTime = Time devoted to a continuous resource T = required time period during which dissimilar media are expected to finish their tasks with synchronisation	The smaller the better	Ratio	A = size T = Time X = ratio	Testing report Operation report showing elapse time		User Maintainer SQA

© ISO/IEC

Table 7.4.3 Compliance metrics (compliance for efficiency)

Metric Name	Purpose	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Satisfaction coverage of compliance items relating to efficiency	How completely does the software adhere the standards, conventions or regulations relating to efficiency?	Previously specify required compliance items based on standards, conventions or regulations relating to efficiency which to be adhered by software. Design test cases in accordance with compliance items. Conduct functional testing for these test cases.	Ratio of satisfied compliance items relating to efficiency $X = 1 - (A / B)$ A= Number of failed compliance items during testing B= Number of total compliance items NOTE: It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied or not.	$0 <= X <= 1$ The closer to 1 is the better.	Absolute	A= Count B= Count X= Count/ Count	Specification of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

7.5 Maintainability metrics

An external maintainability metric should be able to measure such attributes as the behavior of maintainer, user, or system including the software, when the software is maintained or modified during testing or maintenance.

7.5.1 Analyzability metrics

An external analyzability metric should be able to measure such attributes as maintainer's or user's effort or spent resources when they try to diagnose for deficiencies or causes of failures, or for identification of parts to be modified.

7.5.2 Changeability metrics

An external changeability metric should be able to measure such attribute as maintainer's or user's effort by measuring the behavior of maintainer, user or system including the software when they try to implement a specified modification.

7.5.3 Stability metrics

An external stability metric should be able to measure attributes related to unexpected behavior of system including the software when the software is tested or operated after modification.

7.5.4 Testability metrics

An external testability metric should be able to measure such attributes as maintainer's or user's effort by measuring the behavior of maintainer, user or system including software when they try to test the modified or non modify software.

7.5.5 Compliance metrics

An external maintainability compliance metric should be able to measure an attribute such as the number of functions with, or occurrences of compliance problems, which is of the software product to failing to adhere to standards, conventions or regulations relating to maintainability which are required to be adhered.

© ISO/IEC

Table 7.5.1 Analysability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Diagnostic function support	How many causes of failures are identified by the diagnostic function ? Can user identify specific operation which caused failure? (User may be able to avoid falling into the same failure occurrence again with alternative operation.) Can maintainer easily find cause of failure?	Observe behavior of user or maintainer who is trying to resolve failures using diagnostics function	X= A / B A= Number of failures of which maintainer can diagnostic (using the diagnostics function) to understand cause effect relation B= Total number of registered failures	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A=Count B=Count X=Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Data recording during operation	Can user identify specific operation which caused failure? Can maintainer easily find specific operation which caused failure?	Observe behavior of user or maintainer who is trying to resolve failures.	Activity recording during operation ratio X= A / B A= Number of data actually recorded during operation B= Number of data planned to be recorded enough to monitor status of software during operation	0<=X The closer to 1.0 is the better.	Absolute	A=Count B=Count X=Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.5.1 Analysability metrics (continued)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Failure analysis time	Can user easily analyse cause of failure? (User sometimes performs maintenance by setting parameter.) Can maintainer easily find cause of failure? How easy to analyse the cause of failure?	Observe behavior of user or maintainer who is trying to resolve failures.	X= Sum(T) / N T= Tout - Tin Tout = Time at which the causes of failure are found out (or reported back to user) Tin = Time at which the failure report is received N= Number of registered failures	0<=X The shorter is the better.	Ratio	X= Time/ Count T= Time	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
	NOTE: 1. It is recommended to measure maximal time of the worst case and time bandwidth to represent deviation. 2. It is recommended to exclude number of failures of which causes are not yet found when measurement is done. Though, the ratio of such obscure failures should be also measured and presented together.			NOTE: 3. From individual user's view, time is concerned, while effort may also concerned from maintainer's view. Therefore, person-hours may be used instead of time.					
Finding results of failure case	Can user identify specific operation which caused failure? Can maintainer easily find cause of failure?	Observe behavior of user or maintainer who is trying to resolve failures.	Failure cause finding success ratio X=1- (A / B) A= Number of failures of which causes are still not found B= Total number of registered failures	0<=X<= 1 The closer to 1.0 is the better.	Absolute.	A=Count B=Count X=Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Status monitoring during operation	Can user identify specific operation which caused failure? Can maintainer easily find cause of failure?	Observe behavior of user or maintainer who is trying to resolve failures.	Successful status monitoring ratio X= 1- (A / B) A= Number of turns which maintainer (or user) failed to get monitor data B= Number of turns which maintainer (or user) attempted to get monitor data recording status of software during operation	0<=X<= 1 The closer to 1.0 is the better.	Absolute.	A=Count B=Count X=Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.5.2 Changeability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Change recordability	Can the user easily identify revised versions? Can the maintainer easily change the software to resolve problems?	Observe the behavior of user or maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	Change recording ratio X= A / B A= Number of change log data actually recorded B= Number of change log data planned to be recorded enough to trace software changes	0<=X<= 1 The closer to 1.0 is the better or the closer is 0 the less changes have taken place	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator
Ease of parameterisa-tion	Can the user or the maintainer easily change parameter to change software and resolve problems?	Observe behavior of the user or the maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	Change by using parameter success ratio X=1- (A / B) A= Number of turns which maintainer fails to change software by using parameter B= Number of turns which maintainer attempts to change software by using parameter	0<=X<= 1 The closer to 1.0 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator User
Readiness for change	Can the maintainer easily change the software to resolve problem?	Observe behavior of maintainer who is trying to change the software. Otherwise, investigate problem resolution report or maintenance report and product description.	Required effort (in person hours) to change software T = Sum (A / B) / N A= Work time spent to change B= Changed software size N= Number of changes NOTE: 1. A changed software size may be changed executable statements of program code, number of changed items of requirements specification, or changed pages of document etc.	0<T The shorter is the better or the required number of changes were excessive	Ratio.	T= Time A= Time B= Size N= Count	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator

Table 7.5.2 Changeability metrics (continued)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Time spent to implement the change for user's satisfaction	Can the user's problem be solved to his satisfaction within an acceptable time scale?	Monitor interaction between user and supplier. Record the time taken from the initial user's request to the resolution of problem.	Time spent to update user's version Average Time : Tav = Sum(Tu) / N Tu= Trc - Tsn Tsn= Time at which user finished to send request for maintenance to supplier with problem report. Trc= Time at which user received the revised version release (or status report) N= Number of revised versions	0<Tav The shorter is the better., except of the number of revised versions was large.	Ratio	Tu= Time Trc, Tsn = Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Maintainer Operator
Time spent to implement change by the maintainer	Can the maintainer easily change the software to resolve the failure problem?	Observe the behavior of the user and maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	Time to change for maintainer Average Time : Tav = Sum(Tm) / N Tm=Tout - Tin Tout= Time at which the causes of failure are removed with changing the software (or status is reported back to user) Tin= Time at which the causes of failures are found out N= Number of registered and removed failures	0<Tav The shorter is the better, except of the number of failures was large.	Ratio	Tm= Time Tin, Tout = Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

NOTE: 1. It is recommended to measure maximal time of the worst case and time bandwidth to represent deviation.
2. It is recommended to exclude number of failures of which causes are not yet found when measurement is done. Though, the ratio of such obscure failures should be also measured and presented together.
3. From individual user's view, time is concerned, while effort may also concerned from maintainer's view. Therefore, person-hours may be used instead of time.

© ISO/IEC

Table 7.5.3 Stability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Less encountering failures after change	Can user operate software system without failures after maintenance? Can maintainer easily mitigate failures caused by maintenance side effects?	Observe behavior of user or maintainer who is operating software system after maintenance. Count failures which user or maintainer encountered during operating software before and after maintenance. Otherwise, investigate problem resolution report, operation report or maintenance report.	Frequency of encountering failures after change $X = Na / Ta$ Fluctuated frequency of encountering failures before/after change $Y = \{(Na / Ta) / (Nb / Tb)\}$ Na = Number of turns which user encounters failures during operation after software was changed Nb = Number of turns which user encounters failures during operation before software is changed Ta = Operation time during specified observation period after software is changed Tb = Operation time during specified observation period before software is changed	0<X,Y The smaller is the better. The closer to 0.0 is the better.	Ratio.	A= Count T= Time X= Count/ Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

NOTE: 1. User may need specified period to determine side effects of software changes, when revision-up of software is introduced for resolving problems.
2. It is recommend to compare this frequency before and after change.
3. If changed function is identified, it is recommended to sort out whether encountered failures are detected in the changed function itself or the any other ones. The extent of impacts may be rated for each failure.

Table 7.5.3 Stability metrics (continued)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Localisation of modification (Emerging failure after change)	Can user operate software system without failures after maintenance? Can maintainer easily mitigate failures caused by maintenance side effects?	Count failures occurrences after change, which are mutually chaining and affected by change.	Chaining failure emerging per resolved failure X= A / N A= Number of failures emerged after failure is resolved by change during specified period N= Number of resolved failures NOTE: It is recommend to give precise measure by checking whether cause of current failure is attributed to change for previous failure resolution, as possible.	0<X The smaller and the closer to 0.0 is the better.	Absolute.	A= Count N= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

© ISO/IEC

Table 7.5.4 Testability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Effortless testing	Can user and maintainer easily perform operational testing and determine whether the software is ready to operation or not?	Observe behavior of user or maintainer who is testing software system after maintenance.	Time (effort) to test after failure resolution Average X= Sum(T) / N T = Time spent to test to make sure whether reported failure which was resolved or not. N= Number of resolved failures NOTE: If failure which is not resolved or degraded, exclude them and separately measure ratio of such failures.	0<X The smaller is the better.	Ratio	T= Time N= Count X= Time /Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Readiness of built-in test function	Can user and maintainer easily perform operational testing after maintenance and get ready for operation?	Observe behavior of user or maintainer who is testing software system after maintenance.	Built-in test function use success ratio X= A / B A= Number of turns which maintainer can use suitably built-in test function B= Number of turns of test opportunities NOTE: Examples of built-in test functions include simulation function, pre-check function for ready to use and so on.	0 < X <1 The larger and the closer to 1.0 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Test restartability	Can user and maintainer easily perform operational testing with checking step by step after maintenance?	Observe behavior of user or maintainer who is testing software system after maintenance.	Pause and restart of executing test run X = A / B A = Number of turns which maintainer can pause and restart executing test run at desired points to check step by step B= Number of turns of pause of executing test run	0 < X <1 The larger and the closer to 1.0 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.5.5 Compliance metrics (compliance for maintainability)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficiaries
Satisfaction coverage of compliance items relating to maintainability	How completely does the software adhere the standards, conventions or regulations relating to maintainability?	Previously specify required compliance items based on standards, conventions or regulations relating to maintainability which to be adhered by software. Design test cases in accordance with compliance items. Conduct functional testing for these test cases.	Ratio of satisfied compliance items relating to maintainability $X = 1 - (A / B)$ A= Number of failed compliance items during testing B= Number of total compliance items NOTE: It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied or not.	$0 <= X <= 1$ The closer to 1 is the better.	Absolute.	A= Count B= Count X= Count/Count	Specification of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

© ISO/II

7.6 Portability metrics

External portability metrics should be able to measure such attribute as the behavior of the operator or system during the porting activity.

7.6.1 Adaptability metrics

External adaptability metric should be able to measure such attribute as the behavior of user who is trying to adapt software to different specified environments. When user has to apply an adaptation procedure other than previously provided by software for a specific adaptation need, user's effort required for adapting should be measured.

7.6.2 Installability metrics

External installability metrics should be able to measure such attribute as the behavior of user who is trying to install the software in a user specific environment.

7.6.3 Replaceability metrics

External replaceability metrics should be able to measure such attribute as the behavior of user who is trying to use the software in place of other specified software in the environment of that software.

7.6.4 Co-existence metrics

External co-existence ability metrics should be able to measure such attribute as the behavior of user who is trying to the software with other independent software in a common environment sharing common resources.

7.6.5 Compliance metrics

An external portability compliance metric should be able to measure an attribute such as the number of functions with, or occurrences of compliance problems, which is of the software product to failing to adhere to standards, conventions or regulations relating to portability which are required to be adhered.

Table 7.6.1 Adaptability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Adaptable data	Can user or maintainer easily adapt software to data sets in new environment?	Observe user's or maintainer's behavior when user is trying to adapt software to operation environment?	Limitation free operation after data adaptation X = (A / B) A = The number of data which are operable and not observed being incomplete operations caused by adaptation limitations; B= The number of data which is expected to be operable in the environment to which the software is adapted. NOTE: These data mainly include data files, data tupples or databases to be adapted to different data volumes, data items or data structures when for example, business scope is extended.	0<X<1 The larger and close to 1.0 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Environmental adaptability (Organization adaptability to infrastructure of organization)	Can user or maintainer easily adapt software to environment? Is software system enough capable to adapt itself to operation environment?	Observe user's or maintainer's behavior when user is trying to adapt software to operation environment?.	Adaptability to environmental business environments X= 1 - (A / B) A= Number of operated functions of which tasks were not completed or not enough resulted to meet adequate level during operation testing with user's business environment B= Total number of functions which were tested. NOTE: It is recommended to conduct testing which takes account of varieties of combination of infrastructure components of possible user's business environments.	0<X<1 The larger is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.6.1 Adaptability metrics (continued)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Environmental hardware adaptability (adaptability to hardware devices and network facilities)	Can user or maintainer easily adapt software to environment? Is software system enough capable to adapt itself to operation environment?	Observe user's or maintainer's behavior when user is trying to adapt software to operation environment?.	Adaptability to environmental hardwares X= 1 - (A / B) A= Number of operated functions of which tasks were not completed or not enough resulted to meet adequate level during co-operating testing with environmental hardwares B= Total number of functions which were tested. NOTE: It is recommended to conduct overloaded combination testing with environmental hardwares which are possibly co-operated in variety user operation environments.	0<X<1 The larger is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Environmental software adaptability (adaptability to OS, network software and co-operated application software)	Can user or maintainer easily adapt software to environment? Is software system enough capable to adapt itself to operation environment?	Observe user's or maintainer's behavior when user is trying to adapt software to operation environment?.	Adaptability to environmental softwares X= 1 - (A / B) A= Number of operated functions of which tasks were not completed or were not enough resulted to meet adequate level during co-operating testing with operating system softwares or concurrent application softwares B= Total number of functions which were tested. NOTE: It is recommended to conduct overloaded combination testing with operating system softwares or concurrent application softwares which are possibly co-operated in variety user operation environments.	0<X<1 The larger is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.6.1 Adaptability metrics (continued)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
User effortless adaptation	Can user or maintainer easily adapt software to environment?	Observe user's or maintainer's behavior when user is trying to adapt software to operation environment?.	User effort required to adapt to user's environment T=Sum of user operating time to be spared to complete adaptation of the software to user's environment, when user attempt to install or change setup. (Person-hour may be used instead of time.)	0<T. The shorter is the better.	Ratio	T=Time	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.6.2 Installability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficceries
Easiness of Setup Re-try	Can user or maintainer easily re-try setup installation of software?	Observe user's or maintainer's behavior when user is trying to re-try setup installation of software?	X = 1 - (A / B) A = Number of turns which user fail to re-try setup during setup operation B = Total number of turns which user attempt to re-try setup during setup operation NOTE: 1. This metric is suggested as experimental use.	0<=X<= 1 The closer to 1.0 is the better.	Absolute.	A=Count B=Count X=Count/Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Operational installation flexibility	Can user or maintainer easily install software to operation environment?	Observe user's or maintainer's behavior when user is trying to install software to operation environment	X = A / B A = Number of turns which a user succeeded to change install operation for his/her convenience B = Total number of turns which a user attempted to change install operation for his/her convenience NOTE: 1. This metric is suggested as experimental use.	0<=X<= 1 The closer to 1.0 is the better.	Absolute.	A=Count B=Count X=Count/Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.6.2 Installability metrics (continued)

NOTE:
The following complementary metrics may be used.

1) Effortless installation
User manual actions for installation X = A
A= The number of user manual actions for installation
0<X
The smaller is the better.

2) Installation easiness
Installation supporting level X = A
A is rated with, for example:
- Just only executing installation program and nothing more is needed (excellent);
- Instucting function guide for installation (good);
- Source code of program is needed to be modified for installation (poor).

X= Direct Interpretation of measured value

3) Operational installation effort reduction
User Install Operation Procedure Reduction Ratio X = 1- (A / B)
A = Number of install operation procedures which a user had to do after reduced
B = Number of install operation procedures normally
0< X <1
The closer to 1.0 is the better.

4) Easiness of user's manual install operation
Easiness level of user's manual install operation
S = Score of easiness level of user's manual operation
Examples of easiness level are following:
[very easy] only user's watching except just start install or setup functions;
[easy] only user's answering to question from install or setup functions;
[not so easy] user's looking up parameters from tables or fill-in-boxes to be changed and setting them;
[complicated] user's seeking parameter files, looking up parameters from files to be changed and writing them.

X= Direct Interpretation of measured value

© ISO/IEC

Table 7.6.3 Replaceability metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficeries
Data continuation	Can user or maintainer easily continue to use the same data after replacing this software to previous one? Is software system migration going on successfully?	Observe user's or maintainer's behavior when user is replacing software to previous one?	Data continuously use ratio $X = A / B$ A = number of data which are used in other software to be replaced and are confirmed that they are able to be continuously used. B = number of data which are used in other software to be replaced and planned to be continuously reusable.	$0 <= X <= 1$ The larger is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Function inclusiveness	Can user or maintainer easily continue to use similar functions after replacing this software to previous one? Is software system migration going on successfully?	Observe user's or maintainer's behavior when user is replacing software to previous one?	Function inclusion ratio $X = A / B$ A = number of functions which produce as enough similar results as used to be produced and of which changes have not to be required; B = number of tested functions which are similar to functions provided by other software to be replaced.	$0 <= X <= 1$ The larger is the better.	Absolute.	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 7.6.4 Co-existence metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficieries
Concurrent multiple software use with less constraints	How often user encounter any constraints or unexpected failures when user operate concurrently other software?	Use concurrently the software to be evaluated and several other software which user often use.	Concurrent multiple software usable ratio X = A / T A = Number of any constraints or unexpected failures which user encounter during operating concurrently other software T = Time to operate concurrently other software	0<=X<= 1 The closer to 0 is the better.	Ratio	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer SQA Operator

© ISO/IEC

Table 7.6.5 Compliance metrics (compliance for portability)

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Beneficiaries
Satisfaction coverage of compliance items relating to portability	How completely does the software adhere the standards, conventions or regulations relating to portability?	Previously specify required compliance items based on standards, conventions or regulations relating to portability which to be adhered by software. Design test cases in accordance with compliance items. Conduct functional testing for these test cases.	$X = 1 - (A / B)$ A= Number of failed compliance items during testing B= Number of total compliance items NOTE: It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied or not.	$0 <= X <= 1$ The closer to 1 is the better.	Absolute.	A= Count B= Count X= Count/ Count	Specification of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

Annex A (Informative)
Descriptions of the metrics tables

The purpose of these descriptions is to clarify the reader comprehension for the quality metrics formulas proposal in the clause 7 in the technical reports 9126-2, 3and 4.

A.1 Metrics name

Metrics name characterizes measureable attribute of software and represents a unique or group of measurements.

Metrics name has the same or similar name in internal, external and quality in use metrics, when they are intended to be mutually corresponded respectively.

A.2 Purpose of the metrics

This helps to identify what user of metric can know by using the metric.

NOTE: This is described as a questionary style to look up easily in accordance with Goal / Question / Metric flamework.

A.3 Method of application

This helps to understand what way are useful and recommended to apply metrics.

A.4 Measurement, formula and data element computations

This helps to understand what kind of measurement, formula and data element are used to compute measure.

A.5 Interpretation of measured value

This helps to understand the range of measured value and the interpreted better range.

A.6 Metric scale types

The following measurement metric scale types should be identified for each measure, when a user of metrics has the result of a measurement and uses the measure for calculation or comparision. The average, ratio or difference values may have no meaning for some measures. Such Metric scale types are: Nominal scale, Ordinal scale, Intervals scale, Ratio scale, Absolute scale. M'=F(M), where F is the admissible function, explain what the admissible function is (if M is a metric then M'=F(M) is also a metric).

a) **Nominal scale**

M'=F(M) where F is any one-to-one mapping.

This includes classification, for example, software fault types (data, control, other) .An average has a meaning only if it is calculated with frequency of the same type. A ratio has a meaning only when it is calculated with frequency of each mapped type. Therefore, the ratio and average may be used to represent a difference in frequency of only the same type between early and later cases or two similar cases. Otherwise, they may be used to compare mutually frequency of each other type respectively.

b) **Ordinal scale**

M'=F(M) where F is any monotonic increasing mapping that is, M(x)>=M(y) implies M'(x)>=M'(y).

This includes ordering, for example, software failure by severity (negligible, marginal, critical, catastrophic). An average has a meaning only if it is calculated with frequency of the same mapped order. A ratio has a meaning only when it is calculated with frequency of each mapped order. Therefore, the ratio and average may be used to represent a difference in frequency of only the same order between early and later cases or two similar cases. Otherwise, they may be used to compare mutual frequency of each order.

c) **Intervals scale**

M'=aM+b (a>0)

This includes artificial rating scales, for example, rating scales of sensitive questionnaire for asking about usability. These rating scales are comparable and an average has meaning only if it is calculated with the same rated scales for the same asking, but a ratio of such rating scales has no meaning. Even when double scores, it is pointed out only that a score is twice as much as the another, but not that anything is double.

d) **Ratio scale**

M'=aM (a>0)

This includes time interval, for example, time between software failures occurred. An average and a ratio has meaning respectively and they give actual meaning to the values. When the value is double, it is pointed out it takes twice as long as the another.

e) **Absolute scale**

M'=M

they can be measured only in one way.

As an appropriate statistics, Mode and Frequency can be used for Nominal type, Median and Percentile for Ordinal type, Mean and Standard deviation for Interval type, and Geometric mean and Coefficient of variation for Ratio type. All arithmetic analysis of resulting count is meaningful for Absolute type.

For examples, line of code is Ratio type, because there are many different way to measure it(such as LOC, number of characters, and number of bytes). A count of the number of failure and the number of people working on a software project are Absolute type, because they can be measured only in one way.

A.7 Measurements types

For designing procedure for collecting data, interpreting fair meanings, and normalizing measures for comparison, a user of metrics should identify and take account of measure type of measurement employed by a metric.

A.7.1.Size measure type

A measure of this type represents a particular size of software according what it claims to measure within its definition.

NOTE: that software may have many representations of size (like any entity can be measured in more than one dimension - mass, volume, surface area etc.).

Normalizing other measures with a size measure can give comparable values in terms of units of size. The size measures in this report can be used for software quality metrics.

A.7.1.1 Functional size type

Functional size is an example of one type of size (one dimension) that software may have. Any one instance of software may have more than one functional size depending on, for example:

- the purpose for measuring the software size (It influences the scope of the software included in the measurement);
- the particular functional sizing method used (It will change the units and scale).

The definition of the concepts and process for applying a functional size measurement method (FSM Method) is provided by the standard ISO/IEC 14143-Part1.

In order to use Functional Size for normalization a quality assessor needs to ensure that the same functional sizing method is used and that the different software being compared have been measured for the same purpose and consequently have a comparable scope.

Although the following often claim that they represent functional sizes, it is not guaranteed they are equivalent to the functional size obtained from applying an FSM Method and compliant with ISO/IEC 14143-Part1. However, they are widely used in software development:

1. number of spread sheets;
2. number of screens;
3. number of files or data sets which are processed;
4. number of itemized functional requirements described in user requirements specifications.

A.7.1.2 Program size type

In this clause, the term 'programming' represents the number of executions resulting in an action, and the term 'language' represents the type of expression used.

1) Program source size

Programming language should be explained and it should be provided how the non executable statements, such as comment lines, are treated. The following measures is commonly usable.

a) Non-comment source statements (NCSS)

 © ISO/IEC

Non-comment source statements (NCSS) include executable statements and data declaration statements with logical source statements.

b) New program size

A developer may use newly developed program size to represent development and maintenance work product size.

c) Language use

For a same executable statement, the program size depend on the language use.

d)Changed program size A developer may use changed program size to represent size of software containing modified components.

NOTE: Example of computed program size formula is: new lines of code + 0.2 x lines of code in modified components (NASA Goddard).

It may be needed to distinguish a type of statements of source code into more detail like followings.

i) Statement type

- **Logical Source Statement(LSS).** The LSS measures the number of software instructions. The statements are irrespective of its relationship to lines and independent of the physical format in which they appear.
- **Physical Source Statement(PSS)** The PSS measures the number of software source lines of code.

ii) Statement attribute

-Executable statements;

-Data declaration statements;

- Compiler directive statements;

- Comment source statements.

iii) Origin

- Modified source statements;

- Added source statements;

- Removed source statements;

  - Newly Developed source statements: (= added source statements + modified source statements);
  - Reused source statements: (= original - modified - removed source statements);

2) Program word count size

The measurement may be computed by the following so-called Halstead's measure:

Program vocabulary = n1+n2; Observed program length = N1+N2, where is:

- n1: Number of distinct operator words which are prepared and reserved by program language in a program source code;
- n2: Number of distinct operand words which are defined by programmer in a program source code;
- N1: Number of occurrences of distinct operators in a program source code;
- N2: Number of occurrences of distinct operands in a program source code.

3) Number of modules

The measurement is counting the number of modules of a program.

A.7.1.3 Utilized resource size measure type

This type identifies resources utilized by the operation of the software being evaluated. Examples are:

a) **Amount of memory** ex.) amount of disk memory occupied temporally or stable during the software execution;

b) **I/O load** ex.) bit size of communication data (meaningful for backup tools on a network);

c) **CPU load** ex.) percentage of occupied CPU instruction sets per second (meaningful for CPU utilization and efficiency of process distribution in multi-thread software's running on concurrent/parallel systems);

d) **Files and data records** ex.) bit size of file or record;

e) **Documents** ex.) number of document pages.

It may be important taking note of peak (maximal) and average values, as well as periods of time and number of observations done.

A.7.1.4 Specified operating procedure step type

This type identifies static steps of procedure which are specified in a human-interface design specification or a user manual.

The measured value may differ depending on what kinds of description are used for measurement, such as a diagram or a text representing user operating procedure.

A.7.2 Time measure type

The user of metrics of time measure type should record time periods, how many examined sites and how many users took part of measurements.

The user of metrics should be aware that there are many ways in which time can be measured as a unit, including following:

a) Real time unit

This is physical time: i.e. second, minute, or hour. This unit is usually used for describing task processing time of real time software.

b) Computer machinery time unit

This is computer processor's clock time: i.e. second, minute, or hour of CPU time.

c) Official scheduled time unit

This includes working hours, calendar days, months or years.

d) Component time unit

When there are multiple sites, component time identifies individual site and it is an accumulation of individual time of each site. This unit is usually used for describing component reliability, for example, component failure rate.

e) System time unit

When there are multiple sites, system time does not identify individual site but all the sites running, because whole sites are involved into one system. This unit is usually used for describing system reliability, for example, system failure rate.

A.7.2.1 System operation time type

System operation time type provides a basis for measuring time of software availability. This is mainly used for reliability evaluation. It should be identified whether the software is under discontinuous operation or continuous operation. If the ,oftware operates discontinuously, it should be assured that time measurement is done just on the periods the software is active (this is obviously extended to continuous operation).

a) Elapsed time

When use is constant, for example in systems operating for the same length of time each week.

b) Machine powered-on time

For real time, embedded or operating system software that is in full use the whole time the system is operational.

c) Normalized machine time

As in "machine powered-on time" , but pooling data from several machines of different power and applying a correction factor.

A.7.2.2 Execution time type

Execution time type is the time which is needed to execute software to complete a specified task. The distribution of several attempts should be analyzed and mean, deviation or maximal should be computed. The execution under the specific conditions, particularly overloaded condition, should be examined. Execution time measure type is mainly used for efficiency evaluation.

A.7.2.3 User time type

User time type is measured upon time periods spent by individual user on completing tasks by using operations of the software. Some examples are:

a) **Session time**

Measured between start and end of a session. Useful, as example, for drawing behavior of users of a home banking system. For an interactive program where idling time is of no interest or where interactive usability problems only are to be studied.

b) **User operating time (task time)**

Time spent by an individual user to accomplish a task by using operations of the software at an each attempt. It should be well defined what will be the start and end points of the measurement.

c) **User time**

Time spent by an individual user to use the software from the getting started to now. (Approximately, it is how many hours or days user uses the software from beginning.)

A.7.2.4 Effort type

Effort type is the productive time associated with a specific project task.

a) Individual effort

This is the productive time which is needed for the individual person who is developer, maintainer, or operator to work to complete a specified task. Individual effort assumes productive hours only according to a certain number of productive hour per day.

b) Task effort

Task effort measure is an accumulated value of individual for all the project personnel: developer, maintainer, operator, user or others who worked to complete a specified task.

A.7.2.5 Time interval of events types

This measure type is the time interval between event and next one during observation time period. The frequency with observation time period may be used in place of this measure. This is typically used for describing the time between failures occurring successively.

A.7.3 Count measure type

This measure type identifies number of counted number, turn, event or incident with investigation activity.

Investigation activity includes reviewing, testing and operating, and using from view of human-engineering and ergonomics. This measure type should be attached to informative description on the context including periods, and number and profile of experimented site and users during counting are performed, because the measure strongly depends on those.

If attributes of documents or software product are counted, they are static count type. If events or human actions are counted, they are kinetic count type. The followings are belong to static count type.

© ISO/IEC

A.7.3.1 Number of detected fault type

The measurement is to count the detected faults during reviewing, testing, correcting, operating or maintaining. Severity level may be used to categorize them for taking account of impacts.

A.7.3.2 Program structural complexity number type

The measurement is to count program structural complexity. Examples are number of distinct paths or McCabe's cyclomatic number.

A.7.3.3 Number of detected inconsistency type

This measure is to count detected inconsistent items which are prepared for investigation.

a)Number of failed conforming items

Examples:

- Conformance to specified items of requirements specifications;
- Conformance to rule, regulation, or standard;
- Conformance to protocol, data format, media format, character code, or other.

b)Number of failed user expectation

The measurement is to count satisfied/unsatisfied list items, which describe gaps between user's reasonable expectation and software product performance.

The measurement use a questionnaires for asking tester, customer, operator, or end user which kinds of deficiencies are discovered.

The followings are example items for specified user callable function:

- Actually available or not;
- Actually operable effectively or not;
- Actually operable to user's specific intended use;
- Actually Expected/needed or not.

A.7.3.4 Number of change type

This type identifies software configuration items which are detected they have been changed. Example is number of changed places in source code.

NOTE : Changed source lines of code may be counted as changed places in source code.

The followings are belong to kinetic count type.

A.7.3.5 Number of detected failure type

The measurement is to count the detected failures during product development, testing, operating or maintaining. Severity level may be used to categorize them for taking account of impacts.

A.7.3.6 Number of attempt (trial) type

This measure is to count intended operation action of user, checking and finding action of personel, questionnaires and replies, or test cases for investigation during reviewing, testing, correcting, operating or maintaining. Examples are number of detection of design anomalies during the review, number of misunderstanding requirements at the specification level, number of actually tried test cases, number of cancellation operation and so on.

A.7.3.7 Stroke of human operating procedure type

This measure is to count strokes of user human action as kinetic steps of procedure during user is interactively operating the software. This measure quantifies an ergonomic usability as well as an effort to use. Therefore, this is used for usability metrics. Examples are number of strokes to perform a task, number of eye movements, and so on.

A.7.3.8 Score type

This type identifies the score or arithmetic calculation result. Score may includes count and calculation of checking on/off on check lists. Examples: Score of check list; score of questionnaire; Delphi method; etc.

A.8 Input to measurement

This helps to understand what kinds of information or documents are generally required to do measurement.

A.9 ISO/IEC 12207 SLCP Reference

This suggest processes of software life-cycle processes (SLCP) defined in ISO/IEC 12207, in which the metric is beneficially applicable.

A.10 Beneficeries

This helps to understand whose view and benefit are strogly related with the metric, though any other personel and party related to the software are also receive benefit.

Annex B (Informative) Remarks for better use of metrics

B.1 Take accounts of constraints of applied metrics

The measures may not be interpret adequately and not understood sufficiently without context describing situation and condition during data are gathered and metrics are applied.

For examples, 1)"time required to learn operation" measure is often different between skillful operators in similar software systems and unskillful operators. 2) If testing is not so much intensive, measures acquired as testing result may be different so much from the true value.

Therefore, it is recommended to take accounts the following contexts to avoid to interpret inadequately and to misunderstand the measures of metrics.

a) Differences between testing environment and actual user operation one

Are there any remarkable differences between testing environment and actual user operation one?

The followings are examples:

-testing with higher / comparable / lower performance of CPU of users' computer;

-testing with higher / comparable / lower performance of network and communication;

-testing with higher / comparable / lower performance of operating system;

-testing with higher / comparable / lower performance of user interface type provided by operating system.

b) Differences between testing execution and actual user operational execution

Are there any remarkable difference between testing execution and actual user operational execution?

The followings are examples:

-coverage of functional specification or program by test cases;

-test case sampling ratio;

-high speed real time processing testing;

-over loaded data processing testing;

-non-stop operation testing;

-abnormal, exception, or fault injected data input testing;

-frequency of use, such are daily, weekly, monthly, or only at emergency;

-combination of considerable other software, devices, equipment, and systems.

c) User profile under observation

Are there any remarkable different user profile between tested and actual?

The followings are examples:

-heavy, moderate, less or temporal user;

-trained level, e.g., skillful operator or first entry operator;

-expert user or ordinal user;

-office user or home user.

d) Data validation level

Are there any problems come from what kinds of data collection procedures and level of data validation?

The followings are examples.

-procedures of data collection:

automatically with tools or facilities/ manually collected / questionnaires or interviews;

-data source report level:

developers' self reports / reviewers' report / inspected report by evaluator;

-data validation activity:

developers' self check / inspection by independent evaluators.

e) Balance of the extent of the investigation performance and measures

Are there remarkable problems in investigation performance?

It is important to take consider to keep balance between the extent of the investigation performance and measures (detected results) in fair range during testing;

Investigation includes reviewing and testing performance.

Most of the external metrics use measurement value derived from testing cases and results of problem detection in a validation testing or operation testing. Most of internal metrics also use derived from review. Therefore, the measured value of external metrics are affected by the extent of the testing effort, that is, investigation performance. Internal metrics also may be similarly affected the extent of the reviewing, that is, investigation performance.

The user of metrics should identify the extent of justification of the measured values depends on the extent of the investigation performance.

The extent of investigation to detect problems as input affects the extent of detected problems (failures, faults, miss-matching, etc.) and the extent of residual covered problems as outputs.

f) Balance of the extent of the specification clearance and conformance testing

Are there remarkable problems in specification clearance?

Metrics often use measurements which count problems by comparing specification with testing results and by determining whether they are consistently matching or not, that is, whether the software conform to its specification or not. However, specification may not be enough mature to do that appropriately. Therefore, it is recommended to conduct to review and to improve the specification from a view of software product fulfilling its specific intended use in actual, during specification based matching testing case are designed and tested.

B.2 Validity demonstration and use of metrics

The user of metrics should identify the methods for metrics validity demonstration, which are following below.

a) **Correlation**

The variation in the quality characteristics values (the measures of principal metrics in operational use) explained by the variation in the metric values, which is given by the square of the linear coefficient.

An evaluator can predict quality characteristics without measure directly by using these metrics which have correlation ability.

b) **Tracking**

If a metric M is directly related to a quality characteristics values Q (the measures of principal metrics in operational use), for a given product or process, then a change value Q(T1) to Q(T2), would be accompanied by a change metric value from M(T1) to M(T2), which is the same direction (for example, if Q increase, M increase).

An evaluator can detect movement of quality characteristics along time preriod without measure directly by using these metrics which have tracking ability.

c) **Consistency**

If quality characteristics values (the measures of principal metrics in operational use) Q1, Q2,..., Qn, corresponding to products or process 1, 2,..., n, have the relationship Q1 > Q2 > ..., Qn, then the correspond metric values would have the relationship M1 > M2 > ..., Mn.

An evaluator can notice exceptional and error prone components of software by using these metrics which have consistent ability.

d) **Predictability**

If a metric is used at time T1 to predict a quality characteristics values Q (the measures of principal metrics in operational use) at T2, prediction error, which is { (predicted Q(T2)- actual Q(T2)) / actual Q(T2) }, would be with allowed prediction error.

An evaluator can predict the movement of quality characteristics in the future by using these metrics which have predictability.

e) **Discriminative**

A metric would be able to discriminate between high quality software components and low quality software components.

An evaluator can categorize software components and rate quality characteristics values by using these metrics which have discriminative ability.

B.3 Prediction use of metrics

Early detection and prediction of quality of the software product is a one of the most effective use of metrics.

a) Future prediction

- Future measure prediction

To estimate the future values of the same measure by using the current measured values, it is estimated based on trend along with time.

For example, the measured value trend of mean time between failures during testing can be used to estimate the value of mean time in actual operation.

- Future invisible other measure prediction

To estimate the future values of the invisible other measure by using the current measured values, it is estimated based on correlative relations.

For example, the complexity of modules during coding may be used to predict time or effort of program change and test during maintenance.

b) Current fact finding prediction

- Invisible other measure prediction

To estimate the current values of other measure which is supposed to be strongly related mutually by using the current measured values, it is estimated based on correlative relations.

For example, because the number of remaining faults in a software product is an not measurable, it may be estimated by using the number and trend of detected faults.

The metrics which are designed for predicting the attributes should be documented with explanations including below:

-models for predicting the attribute;

-formula for predicting the attribute;

-experience for predicting the attribute;

-justification for predicting the attribute.

The metrics which are designed for predicting the attributes may be sophisticated with the metrics validation procedure containing below:

-identify the samples of measures of attributes which are to be predicted

-identify the metrics which are supposed to be capable for prediction

-perform a statistical analysis

-document the results

-iterate above periodically

B.4 Detect quality problem prone components

The following methods are usually employed to detect quality problem prone components:

-extremely deviated in distribution;

-exceeding boundaries of adequate pair of upper or lower limit in trend;

-extremely deviated or exceeding boundaries of adequate pair of upper or lower limits correlation.

It is recommended to use chrats such Pareto, trend along the time or histograms.

B.5 Displatying measurement results

a) Displaying quality characteristics evaluation result

For example,following graphical presentations are useful to display quality evaluation result for each quality characteristics and subcharacteristics:

Radar chart; Bar chart and so on.

b) Displaying measures

They are useful grafical presentations such Pareto chart, trend chart along the time, histograms, correlation chart and so on.

Annex C (Informative)

Example of Quality Measurement in the industry using ISO/IEC 9126 concepts

This clause presents a step by step quality approach using ISO/IEC 9126 concepts and quality metrics. This example is a summary of the steps use to put in place, execute a quality process and measure the software quality. This example is based on five-year experience and implementation in the industry. The objective of this clause is to procure to the reader an idea on how he could implement a quality process using ISO9000 and ISO/IEC 9126 concepts in the development process.

C.1 Overview of development and quality process

a) Development process view for software

These figure present an overview of the software life cycle process, quality approach steps and quality metrics (Internal, external and quality in use) at each step of the development process. At this stage, Quality criteria and internal quality and metrics are use to evaluate the software quality.

	Step #1	Step #2	Step #3	Step #4
Development Process (ISO/IEC 12207 Software Life-Cycle Processes)	Software Requirements Analysis	Software Architectural Design	Software Detailed Design	Software Coding and Testing (Coding)
Quality Approach and Quality Deliverables	-Goal Quality (GO) -Required Product Quality (RPQ) -Quality Criteria and Weight -Quality Strategy Plan -Testing Strategy Plan -Configuration Plan	-Design Quality (DQ) -Development methodology -Standards -Design standards -Program structure	-Estimated / Predicted Product Quality -Internal Quality Characteristics and Measurement	-Estimated / Predicted Product Quality -Internal Quality Characteristics and Measurement

Figure C.1a Quality Approach related to Development Process of Software Life Cycle Process

 © ISO/IEC

b) Software qualification Testing process for software

At this stage, external metric is use to evaluated the software quality.

Step #5	Step #6	Step #7	Step #8	Step #9
Software Coding and Testing (Unit Test)	Software Integration, Software Qualification Testing (Integrated Testing)	System Integration, System Qualification Testing (System Testing)	Software Acceptance Support (Acceptance Testing)	Software Installation Support, Operational Testing
-Testing Quality Quality Measure -Incident Report	-External Quality Characteristics and Measurement -Testing log -Test Incident Report -Test Summary Report -Pass/Fail Criteria -Signoff	-Complete Functionality -Testing Condition: 1) Normal 2) Exceptional 3) Stress and Volume -Quality -External Quality Characteristics and Measurement -Test log -Test Incident Report -Pass/Fail Criteria -Signoff	-Complete Functionality -Performance Testing -Quality -External Quality Characteristics and Measurement -Test log -Test Incident Report -Pass/Fail Criteria -Signoff	-External Quality, Quality In Use Characteristics and Measurement -Test log -Test Incident Report

Figure C.1b Quality approach related to Development process of Software Life cycle Process

C.2 Quality Approach Steps

Step #1 Goal Quality

Software Requirements analysis represent the necessary requirements from the real user. Goal Quality (GQ) represents the necessary and sufficient quality reflecting real user needs. The following relative weights for each quality characteristics have been assigned, based on the quality goal, objectives and the requirement (present at the software requirement analysis and quality requirement). Assigning relative weights will allow the developers and users to focus their efforts on the most important aspects of the software /system. The Quality Strategy Plan will describe the quality requirement, measurement and signoff (pass/fail quality criteria) process for each step of the software development process. The testing strategy plan will focus on the certification process using quality characteristic, subcharacteristic and measuring defect.

Table C.2.1 Example of overall quality characteristics and weights

CHARACTERISTIC	WEIGHT (High/Medium/Low)
Functionality	
Reliability	
Usability	
Efficiency	
Maintainability	
Portability	
Quality in Use	

© ISO/IEC

Table C.2.2 Example of overview of quality subcharacteristics

Characteristic	Subcharacteristic	User WEIGHT	Technical WEIGHT	Goal Quality Required Product Quality
Functionality	Suitability			
	Accuracy			
	Interoperability			
	Compliance			
	Security			
Reliability	Maturity (hardware/software/ data)			
	Fault tolerance			
	Recoverability (data, process, technology)			
	Redundancy			
Usability	Understandability			
	Learnability			
	Operability			
	Attractiveness			
Efficiency	Time behavior			
	Resource utilisation			
Maintainability	Analyzability			
	Changeability			
	Stability			
	Testability			
Portability	Adaptability			
	Installability			
	Replaceability			
	Co-existence			
Quality In Use	Effectiveness			
	Productivity			
	Safety			
	Satisfaction			

Step #2: Design Conformity

All the next steps are driven by the quality criteria and characteristic, subcharacteristic weight.

Step #2 identify the conformity for Data, Process and Technology, using metrics. For each step, a pass or fail criteria are verify using metrics.

		QUALITY ASSURANCE FORM					No : 1
Design Phase (1/1)		Process Name: ____ Date: ____ Process # : ____ Architect : ____					
Charac.	Sub-Char.	Indicators to measure			Value	Freq.	Resp.
Functionality	Suitability and Accuracy	FSu-1a	Process specification change ratio	NB of change to a process due to mis-understanding of the requirements / NB Total process		after reviews and at each change request	QA Team
		FSu-1b	Process specification change ratio	Nb of change to process due to change in environments(legal, organizational, technological) / Nb Total Process		after reviews and at each change request	QA Team
		FSu-1c	Process specification change ratio	Nb of change to Process due to change in the requirements / Nb Total Process		after reviews and at each change request	QA Team

Figure C.2.1a An example of Metrics Form Sheet

 © ISO/IEC

Step #3: Software Detailed Design Conformity

Step #3 identify the requirements conformity for data, function and technology, using metrics.

		QUALITY ASSURANCE FORM					No : 1
Software Detailed Design Phase (1/1)		Function Name: ____ Date: ____ Function # : ____ Analyst : ____					
Charac.	Sub-Char.	Indicators to measure			Value	Freq.	Resp.
Functionality	Suitability and Accuracy	FSu-1a	Functional specification change ratio	NB of change to functions due to mis-understanding of the requirements / NB Total Function		after reviews and at each change request	QA Team
		FSu-1b	Functional specification change ratio	Nb of change to functions due to change in environments(legal, organizational, technological) / Nb Total Function		after reviews and at each change request	QA Team
		FSu-1c	Functional specification change ratio	Nb of change to functions due to change in the requirements / Nb Total Function		after reviews and at each change request	QA Team

Figure C.2.1b An example of Metrics Form Sheet

Step #4: Software Coding Conformity

Step #4 identify the conformity for coding using internal metrics.

	QUALITY ASSURANCE FORM **No : 1**
Software Coding Phase (1/1)	**Program Name:** ______ **Date:** ______ **Program #** : ______ **Programmer** : ______

Charac.	**Sub-Char.**	**Indicators to measure**			**Value**	**Freq.**	**Resp.**
Functionality	**Suitability and Accuracy**	FSu-1a	Program specification change ratio	**NB of change to program due to mis-understanding of the requirements** / **NB Total program**		after reviews and at each change request	QA Team
		FSu-1b	Program specification change ratio	**Nb of change to program due to change in environments(legal, organizational, technological)** / **Nb Total program**		after reviews and at each change request	QA Team
		FSu-1c	Program specification change ratio	**Nb of change to program due to change in the requirements** / **Nb Total program**		after reviews and at each change request	QA Team

Figure C.2.1c An example of Metrics Form Sheet

© ISO/IEC

Step #5,6,7: Software Testing

Step # 5,6,7,8 measuring the software quality using external metrics.

		QUALITY ASSURANCE FORM					No : 2
Unit Testing (1/1)		Program Name : ____________ DATE : ________ Program # : ________ Programmer : ____________					
Charac.		Indicators to measure			Value	Freq.	Resp.
F n c t i o n a l i t y	Suitability	FSu-1d	Functional Cosmetic change ratio	Nb of change to functions due to Cosmetic change ________ Nb Total Function		after reviews and at each change request	QA Team
	Interoper ability	FSu-6	Matched data format ratio	Nb of transactions with each of the other programs ________ Nb Total of transactions		Once	Progmer

Figure C.2.1d An example of Metrics Form Sheet

	QUALITY ASSURANCE FORM	No : 3
Integrated Testing (1/2)	Function Name: ____________ DATE: ________ Function # : ________ Analyst resp. : ____________	

Charac.	Sub-Charac	Indicators to measure			Value	Freq.	Resp.
Functionality	Suitability	FSu-1a	Functional specification change ratio	Nb of changes to functions due to mis-understanding of the requirements / Nb Total Function		at each change request	QA Team
		FSu-1b	Functional specification change ratio	Nb of changes to functions du to change in environments(legal, organizational, technological) / Nb Total Function		at each change request	QA Team
		FSu-1c	Functional specification change ratio	Nb of changes to functions du to change in the requirements / Nb Total Function		at each change request	QA Team
		FSu-1d	Functional Cosmetic change ratio	Nb of change to functions due to Cosmetic change / Nb Total Function		after reviews and at each change request	QA Team
		FSu-1e	Ratio DataBase Change	Nb of change request / Total of Data Element		after reviews and at each change request	QA Team
	Accuracy	FAc-1	Anomalies ratio 1	Nb of anomalies that hinders the function to give results / Nb of tests for the function		Daily	Analyst (Dev. Team)
		FAc-2	Anomalies ratio 2	Nb of anomalies where function give results but wrong / Nb of tests for the function		Daily	Analyst (Dev. Team)
		FAc-4	Anomalies ratio 3	Nb of good but unexpected results because of script incorrectness / Volume of tests		Daily	Analyst (Dev. Team)
	Interoper ability	FIn-1	Matched data format ratio	Nb of matched transactions with each function / Nb Total of transactions matched		Once	Analyst (Dev. Team)

Figure C.2.1e An example of Metrics Form Sheet

 © ISO/IEC

C.3 Measuring Quality

At each step, quality is measured using metrics for the Software. Multiple measurement could be done according to the step you are measure

Table C.3.1 An example of table representing measuring quality and step you are measure

Characteristic	Subcharacteristic	Software Architectural Design	Software detailed Design	Software Coding and Testing	....	System Qualifica -tion Testing	Software Acceptance Suport
Functionality	Suitability						
	Accuracy						
	Interoperability						
	Security						
Reliability	Maturity (hardware/software/data)						
	Fault tolerance						
	Recoverability (data, process, technology)						
	Redundancy						
Usability	Understandability						
	Learnability						
	Operability						
	Attractiveness						
Efficiency	Time behavior						
	Resource utilisation						
Maintainability	Analyzability						
	Changeability						
	Stability						
	Testability						
Portability	Adaptability						
	Installability						
	Replaceability						
	Co-existence						