International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.3930

(08/2012)

SERIES Q: SWITCHING AND SIGNALLING

Signalling requirements and protocols for the NGN – Testing for next generation networks

Performance testing of distributed systems – Concepts and terminology

Recommendation ITU-T Q.3930

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4, 5, 6, R1 AND R2	Q.120–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	Q.800–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR THE NGN	Q.3000–Q.3999
General	Q.3000–Q.3029
Network signalling and control functional architecture	Q.3030–Q.3099
Network data organization within the NGN	Q.3100–Q.3129
Bearer control signalling	Q.3130–Q.3179
Signalling and control requirements and protocols to support attachment in NGN environments	Q.3200–Q.3249
Resource control protocols	Q.3300–Q.3369
Service and session control protocols	Q.3400–Q.3499
Service and session control protocols – supplementary services	Q.3600–Q.3649
NGN applications	Q.3700–Q.3849
Testing for next generation networks	**Q.3900–Q.3999**

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Q.3930

Performance testing of distributed systems – Concepts and terminology

Summary

Recommendation ITU-T Q.3930 describes terminology and concepts of performance tests with a generalized view of performance characteristics as the starting point.

What kind of characteristics are indicators of a product's performance and what kind of measurement data is captured and processed to provide relevant figures on requested performance are in this view the core of performance testing.

The Recommendation includes a set of terminology and descriptions of concepts in performance testing that could be accepted as a common base for descriptions of performance and performance tests.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T Q.3930	2012-08-13	11

Keywords

Performance testing.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Introduction

The background to the present Recommendation is that few standard specifications define any kind of performance or performance targets. A common explanation to this is that capacity issues are not a matter for standardization bodies, but for product vendors and product buyers to decide. This opinion excludes the need for definition and application of other performance characteristics such as reliability, stability, efficiency and many others.

Another more fundamental reason for writing the present Recommendation is that there are no strict definitions of what kind of characteristics should be regarded as indicators of performance.

In the absence of strict definitions in this area, one tends to talk about performance characteristics in terms of types of performance tests, such as robustness tests or availability tests. A consequence of this is that conformance testing as specified in ISO/IEC 9646-1 does not cover performance tests explicitly.

A starting point is therefore a set of terminology and descriptions of concepts in performance testing that could be accepted as a common base for discussions about performance and performance tests.

Recommendation ITU-T Q.3930

Performance testing of distributed systems – Concepts and terminology

1 Scope

The present Recommendation describes terminology and concepts of performance tests with a generalized view of performance characteristics as the starting point.

What kind of characteristics are indicators of a product's performance and what kind of measurement data is captured and processed to provide relevant figures on requested performance is in this view the kernel of performance testing.

Methods for performance testing will consequently be guided by the requirements on expected output.

A set of following Recommendations will describe strategies, methodologies and techniques of performance testing.

1.1 Organization of the present Recommendation

The present Recommendation has two parts.

Part one is about performance characteristics and performance metrics. It contains a general view of performance characteristics followed by a general view of measured objects, or what is measured in performance tests. Part one also contains general objectives and attributes of performance data and performance metrics, i.e., a conceptual view of performance metrics that is followed by a catalogue of abstract performance metrics that covers the performance characteristics described earlier. At the end of part one is a conceptual view of performance data processing, i.e., the steps between captured performance data and presented results on performance.

Part two is about performance testing concepts. It starts with a general view of performance testing followed by a conceptual view of the test environment and finally a conceptual view of performance test specifications.

2 References

None.

3 Definitions

3.1 Term defined elsewhere

None.

3.2 Terms defined in this Recommendation

The core text of this Recommendation defines the terms used in performance testing.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API Application Programming Interface

CPU Central Processing Unit

CSCF Call Session Control Function

DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-line Memory Module
DoS	Denial of Service
GMT	Greenwich Mean Time
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPsec	Internet Protocol Security
IPTV	Internet Protocol TV
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IUT	Implementation Under Test
KSR	Kilo Service Request
MSR	Mega Service Request
MTBF	Mean Time Between Failures
OSI	Open Systems Interconnection
SC	Supporting Components
S-CSCF	Serving-Call Session Control Function
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SMS	Short Messaging Service
SOAP	Simple Object Access Protocol
SUT	System Under Test
TC	Tested Components
TCP	Transmission Control Protocol
TS	Test System
UDP	User Datagram Protocol
UE	User Equipment

5 Conventions

None.

6 Performance characteristics

6.1 Classifying performance characteristics into categories

An almost infinite number of performance characteristics can be applied on any computer system. However, measuring a complete set of performance characteristics of a system (if possible) is not only impractical, costly and time consuming, it can also be argued if it will improve or confuse the understanding of the tested systems performance.

Performance measurements are focused on obtaining figures about a selected set of performance attributes of a system.

To simplify the selection of performance characteristics for a system it is convenient to group performance characteristics about similar or related aspects of performance into performance categories and select desired performance characteristics in each category. Examples of such categories can be powerfulness, reliability or efficiency characteristics of a system. These performance categories can be exemplified by a car where:

– A powerfulness characteristic is "top speed".

– A reliability characteristic is "maintenance intervals".

– An efficiency characteristic is "mileage or fuel consumption per 100 km".

6.2 Powerfulness characteristics

The powerfulness category of performance characteristics contains indicators of speed and quantity of service production. The powerfulness category can be applied on all levels from system or application level down to low-level services of different components. The powerfulness category has subcategories for responsiveness, capacity and scalability.

6.2.1 Responsiveness characteristics

Time definitions

The time to handle a service request can be split into a large number of steps, where each step causes a time delay. The responsiveness of a service request is the sum of all these time delays, the total time it takes to handle it. Every time delay falls into one of three time delay categories:

1. transmission time
2. queuing time
3. processing time

Transmission time and transmission rate

Transmission time is delays caused by transferring data related to the processing of a service request. Transmission time can be measured on any level from sending a service request to the SUT down to transmission of data between CPU and memory.

Transmission time in a network is the time to send a specific amount of data between two nodes over a network. The transmission time starts when the first bit is transmitted by the sender and ends when the last bit is received by the receiver. The transmission time is a function of the amount of data being transmitted and the transmission rate or bandwidth of the used link. A network link with a bandwidth of 100 Mbit/s has a nominal transmission rate of 12.5 Mbyte/s. However, the transport and network layer protocols used to send the data between the end nodes in a packet-switched network will decrease the nominal transmission rate by 20-40%. The resulting bandwidth is the effective transmission rate. Another factor that governs the effective transmission rate is the packet size.

Queuing time

Queuing time are delays caused by waiting for a service of some kind related to the processing of a service request. Queuing time can, like transmission time, also be measured on multiple levels from a system service request queuing to be handled by an application server down to a process queuing for CPU time.

Every case of queuing time is caused by a mismatch between available resources and requested resources.

Processing time

Processing time are delays caused by processing a service request. Processing time can be measured on multiple levels from sending a service request to the SUT down to application code execution time in the CPU.

Responsiveness definitions

A responsiveness value is the sum of a large number of time elements classified as transmission time, queuing time or processing time. From a performance measurement perspective such collections of time elements do not describe the nature of measured system responsiveness. Therefore, other collections of measured time elements are used to better describe the SUT responsiveness to service requests.

Responsiveness characteristics (Figure 1) describe different kinds of service processing time delivered by a system and include:

1. response time
2. round-trip time
3. latency time
4. timeliness

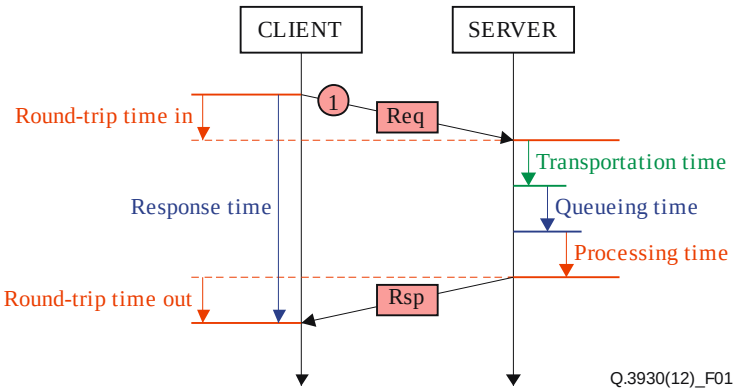

Figure 1 – Examples of responsiveness time elements

Response time

Response time is the time to respond to a service request. It is usually measured from the moment the last byte of a service request is sent until the moment when the first byte of the response arrives. Response time is measured for individual types of responses to a requested service type. This applies also to performance tests with mixes of service requests.

Round-trip time

The transfer time of data between the service requesting tool and the system under test is an important component of the response time of a service request. Round-trip time is used to separate transfer time from time spent in the SUT processing a request. Round-trip time is the time to send a

data packet to an Internet node and get a response message back, i.e., the signalling time between two nodes. Round-trip time is usually referred to as time to "ping" a node.

Latency time

Latency time is a general term for "invisible" delays in service delivery, i.e., time spent waiting for some reaction to a service request. Latency measurements can be applied on many levels in performance tests from service request processing down to "reaction time" of individual hardware components such as a disk. In the responsiveness context any kind of latency has a negative impact on the responsiveness of a service. Here follow some examples of latency time.

Service latency (Figure 2) is the time to process a service request in a system. Service latency is very close to response time but does not include transmission time between requester and server.

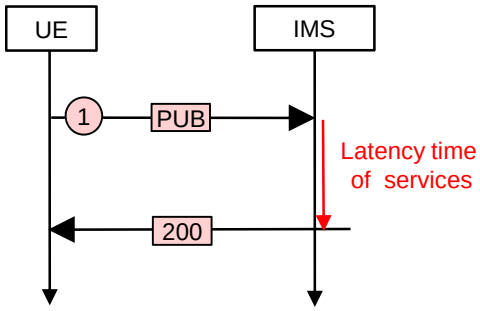

Figure 2 – Service latency

Load level latency (Figure 3) is the time to adjust a system to a rapid increase of incoming service requests. This can be compared to the delay from pushing the accelerator in a car to the bottom, until the car starts responding with engine working at its maximum accelerating the car.

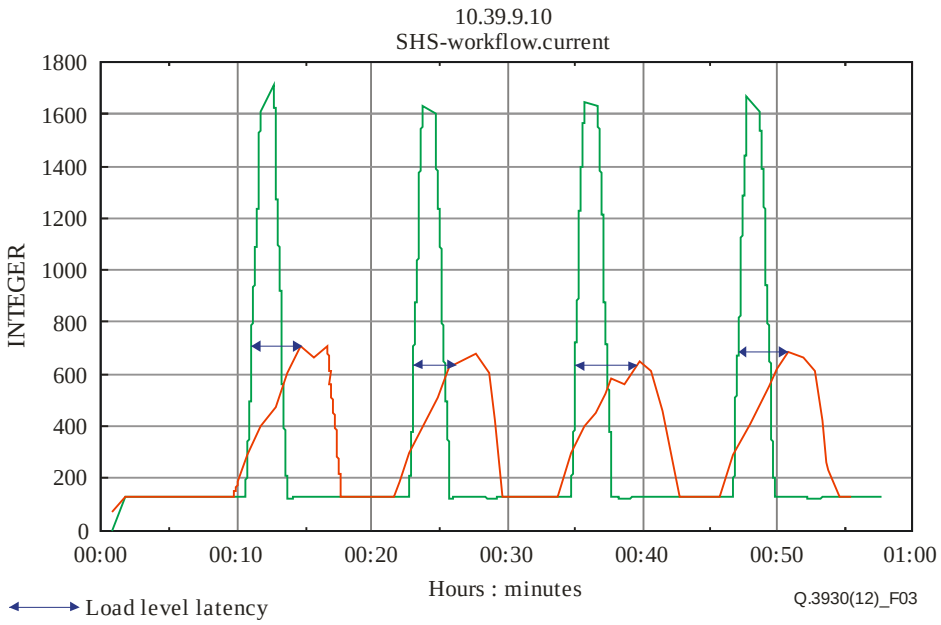

Figure 3 – Load level latency

Latency of coupled services (Figure 4) applies to situations where execution of a service results in execution of another "coupled" service. An example of this is the presence service in the SIP protocol. In this context, latency of coupled services is the time from update of a publication's status with a PUBLISH request, until the publication server starts sending NOTIFY messages to every active subscriber of the publication.

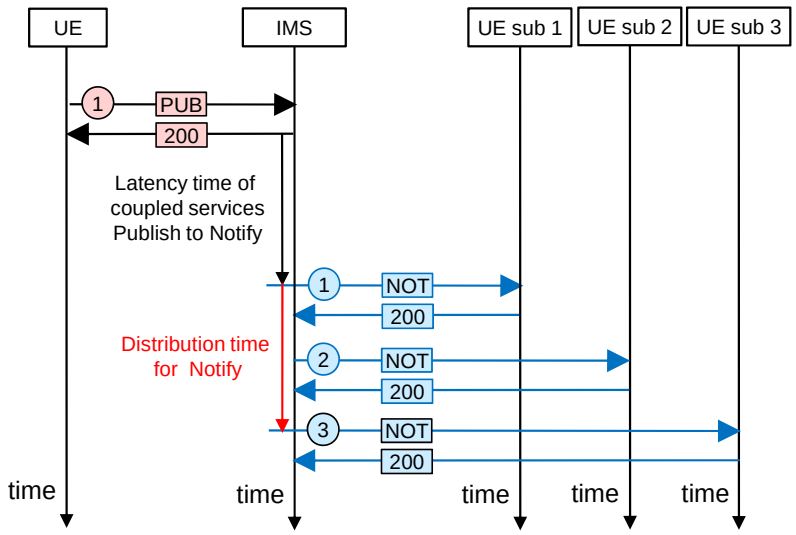

Figure 4 – Latency of coupled service

Network latency is the delay in sending data between nodes over a network.

Network latency contains several components. Each component belongs to one of two groups – static network latency or dynamic network latency.

Static network latency components are included in every data transmission and contain:

1. the transmission time to get the data across the network with the given effective transmission rate;
2. the delay time caused by the physical signalling distance between the end nodes;
3. the delay time caused by the network and transport protocol software on each node in the network chain between the end nodes; and
4. the delay time caused by security functions, such as encryption/decryption of data.

The dynamic network latency components appear occasionally and contain:

1. the queuing time for transmission due to network congestions; and
2. retransmission time to get data across the network due to poor transmission quality.

Timeliness or time accuracy

Timeliness refers to measurements of delay time between received frames in a streamed service. The purpose is to measure if a data frame arrives in time to avoid noticeable disturbances in a media stream or not.

Timeliness can also be regarded as a reliability characteristic of accuracy in delivery time.

6.2.2 Capacity characteristics

Capacity characteristics describe different service request volumes handled by a system including:

1. arrival capacity
2. peak capacity

3. in progress capacity
4. streaming capacity
5. throughput capacity

Arrival capacity

Arrival capacity characteristics describe a system's ability to accept incoming service requests per time unit continuously on a given hardware configuration. Arrival capacity can be measured for individual services or mixes of services.

Peak capacity

Peak capacity characteristics describe a system's ability to handle an overload of incoming service requests during a specified period of time on a given hardware configuration. Peak capacity can be measured for individual services or mixes of services.

In progress capacity

In progress capacity characteristics describe a system's ability to handle multiple services requests concurrently on a given hardware configuration. In progress capacity can be measured for individual services or mixes of services.

Active load in progress capacity is a performance characteristic for the maximum number of concurrent services requests causing active load. Services causing active load require fast delivery and demand processing resources (CPU). Services can be stateless or stateful.

Passive load in progress capacity is a performance characteristic for the maximum number of concurrent services requests causing passive load. Services causing passive load have long duration and require no or little processing resources (CPU). Services are stateful. A typical passive load service is a pending user session. A pending user session is a prerequisite for other service requests and occupies some resources such as memory and timer functions.

Streaming capacity

Streaming capacity characteristics describe a system's ability to handle multiple flows of data streams concurrently. The data streams might be multimedia streams where any variation in arrival rate of frames is critical.

Throughput capacity

Throughput capacity characteristics describes a system's ability to deliver completed service requests per time unit continuously on a given hardware configuration. Throughput capacity can be measured for individual services or mixes of services.

Throughput capacity in a network describes the ability to transmit application data per time unit. The throughput capacity is limited by the bandwidth of a used transmission link. The bandwidth is a measure of the maximum number of bits that can be transmitted on a link per second. The bandwidth value is set by the hardware sending and receiving data on a link. The throughput capacity can be expressed as a percentage value of the bandwidth. Transmitted data used by the transport layer protocol and the network layer protocol in the packets surrounding the transmitted application data are not included in the throughput capacity figures. The throughput capacity will therefore always be less than 100% of the bandwidth value.

6.2.3 Scalability characteristics

Scalability characteristics describe how service processing of a system can be expanded.

For distributed systems, scalability can be applied in three dimensions:

1. capacity scaling

2. distribution scaling
3. functionality scaling

Capacity scaling

Capacity scaling are indicators of the relation between hardware resource increases and related service capacity increases, i.e., the SUT's ability to increase the service capacity by addition of more hardware resources to the current configuration.

Distribution scaling

Distribution scaling are indicators of how service capacity and service responsiveness are affected by adding or changing the distribution nodes in a distributed system.

Functionality scaling

Functionality scaling are indicators of how service capacity and service responsiveness are affected by adding or changing the functionality.

6.3 Reliability characteristics

The reliability category of performance characteristics contains indicators of how predictable a system's service production is. The performance category has sub-categories for quality of service, stability, availability, robustness, recovery and correctness.

6.3.1 Quality-of-service characteristics

Quality-of-service characteristics are stated requirements on service delivery performance. The requirements are usually stated as statistical values for a long period of time or a large number of service requests. Quality-of-service characteristics are therefore regarded as indicators of reliability in this context. Quality-of-service characteristics are also covered by other indicators of reliability in this clause such as:

- stability figures for acceptable/unacceptable performance
- correctness figures for correctly processed services such as transferred media codec.

Quality-of-service characteristics are also applied as measurements of network reliability. Quality-of-service characteristics of network services includes several figures such as:

- transmission failures due to network problems with following time-outs;
- transmission errors in terms of poor media quality due to corrupted data
- transmission timeliness such as poor media quality due to media packets arriving late.

6.3.2 Stability characteristics

Stability characteristics are indicators of a system's ability to maintain measured performance figures for powerfulness and efficiency during service delivery regardless of time. Stability characteristics are related to measured results of powerfulness and efficiency. Stability characteristics can also be evaluations of measured performance figures versus stated figures for acceptable performance, i.e., stability as frequencies of services with acceptable/unacceptable performance. Stability figures for acceptable/unacceptable performance can also serve as performance measurements of quality of service. Stability characteristics can also be indicators of performance trends (see Figure 5), i.e., problems resulting in gradually deteriorating performance figures.

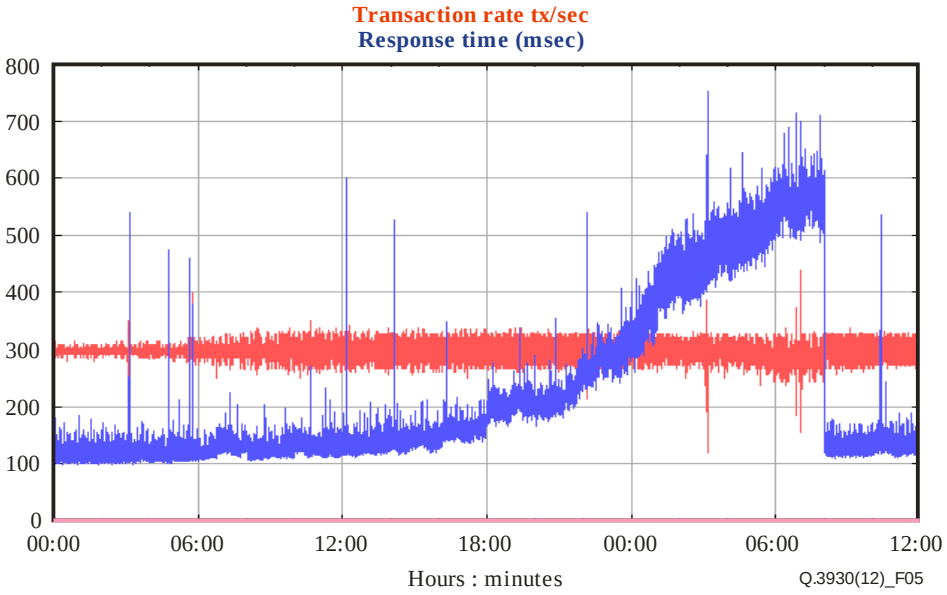

Figure 5 – Examples of performance trends

6.3.3 Availability characteristics

Availability characteristics are indicators of a system's ability to delivery services over time. Different availability characteristics are applied on hardware (physical availability) and on software (logical availability).

Planned and unplanned downtime

Availability characteristics cover unplanned downtime of services only. Planned downtime of services is not regarded as unavailability, however, consequences of planned downtime are usually measured in other reliability characteristics, such as recovery characteristics for time to start/restart services.

Physical availability characteristics

Physical availability characteristics are statistical indicators of operational time between failures for hardware equipment. Physical availability characteristics are usually expressed as uptime figures. Physical availability characteristics are usually excluded in performance measurements because they are extremely costly (broken equipment) and require testing time that is far beyond the scope of a performance test project.

Hardware equipment may include network components.

Logical availability characteristics

Logical availability characteristics are external measurements of frequencies of error responses to service requests due to application software problems. Error responses to bad or inadequate requests are not regarded as logical unavailability. Logical availability characteristics are usually expressed as probability figures for service delivery, but can also be expressed as uptime or frequencies for service delivery. Logical availability characteristics imply physical availability.

6.3.4 Robustness characteristics

Robustness characteristics are indicators of a system's services levels, i.e., services capacity and/or service responsiveness under extreme conditions. Extreme conditions can be caused internally by hardware failure or software malfunctioning, or externally by extreme peak load conditions, or by denial-of-service attacks or other malice attempts.

Service capacity reduction

One consequence of extreme conditions is how the system's service capacity is affected. An ideal system is not affected at all by extreme conditions, i.e., the reduction of service level is 0%. Most likely, the service capacity is reduced to some extent, i.e., a reduction of service level in the range 1% to 99%. The worst consequence is a total stop in service production, i.e., a reduction of service level by 100%.

Service responsiveness deterioration

Another consequence of extreme conditions is how the system's responsiveness deteriorates. An ideal system is not affected at all by extreme conditions, i.e., the response time is unchanged. Most likely, the service responsiveness gets worse to some extent, i.e., the response time increases. The worst possible consequence is a total stop in service production, i.e., the response time is endless, or the service request gets a time out.

6.3.5 Recovery characteristics

Recovery characteristics are indicators of production disturbances from hardware or software malfunctioning. Recovery covers a large number of different operations. In this context, system recovery and service recovery are reviewed.

System recovery characteristics

System recovery characteristics are usually different measurements of time to bring a system into a fully operational state after a major production disturbance. Service recovery includes all operations from replacement of failing hardware to recovery of middleware and application components such as databases, etc.

Service restart characteristics

Service restart characteristics are usually different measurements of time to resume full service availability after system recovery procedures are complete. A more extreme version of service restart is resuming services on a back-up system or a virtual server.

6.3.6 Correctness characteristics

Correctness characteristics are indicators of a system's ability to deliver correctly processed service requests under high or odd load conditions. Correctness can in this context also include streamed services, such as correctly transferred media codecs, i.e., quality-of-service characteristics for speech or multimedia transfers.

6.4 Efficiency characteristics

The performance category efficiency contains different types of indicators of resource usage and resource utilization. Efficiency is measured on three levels:

1. service level
2. platform level
3. network level

On the service level, the efficiency of application service resources usage is measured. Efficiency characteristics on the service level have sub-categories for service resource usage, service resource linearity, service resource scalability and service resource bottlenecks.

On the platform level, the efficiency of platform resource distribution is measured for how well supply and demand of resources are met. Efficiency characteristics on the platform level have sub-categories for platform resource utilization, platform resource distribution and platform resource scalability.

On the network level, the efficiency of resource distribution is measured, i.e., how well does supply meet the demand for resources. Efficiency characteristics on the platform level have sub-categories for platform resource utilization, platform resource distribution and platform resource scalability.

Efficiency characteristics on the platform level also include resource utilization on network components.

6.4.1 Service resource usage characteristics

The service resource usage characteristics are indicators of the amount of resources required for processing a tested service or a mix of services. The level of service resource usage is a measure of the efficiency of the tested service or mix of services. A low level of resource usage for a service implies a more efficient implementation compared to higher level of resource usage. Measured resources can be physical (hardware resources) and logical (software resources or referenced services).

6.4.2 Service resource linearity characteristics

The service resource linearity characteristics are indicators of a system's ability to use a constant amount of resources for the production of a service regardless of the actual load level on the system. Service resource linearity is measured at different service request rates and for different services or mixes of services. Measured resources can be physical (hardware resources) and logical (software resources or referenced services).

6.4.3 Service resource scalability characteristics

The service resource scalability characteristics are indicators of a system's ability to accommodate additional platform resources for increased production of a service. Service resource scalability is for different services or mixes of services.

6.4.4 Service resource bottleneck characteristics

The service processing capacity of every system has an upper limit. If the limit is reached due to a single cause it is usually referred to as a bottleneck, since elimination of the single cause will expand the service processing capacity. The implication of a bottleneck is that available resources cannot be used efficiently to produce services. The service processing capacity will, consequently, increase significantly if an identified bottleneck is eliminated. Most systems have many bottlenecks and the most limiting bottleneck hides all the others. Therefore, the elimination of one bottleneck will only increase the service processing capacity up the limit set by the second-worse bottleneck and so on. There are five types of bottlenecks:

- configuration bottlenecks;
- processing or implementation bottlenecks;
- resource interference bottlenecks;
- design bottlenecks;
- architectural bottlenecks.

Configuration bottlenecks

Configuration bottlenecks are capacity limitations due to wrong software configurations or other similar reasons such as poor balance between hardware resources. Examples can be too few threads in a database server, or mismatch or a server that cannot use all CPU power due to lack of memory. Configuration bottleneck are usually the easiest to correct.

Processing bottlenecks

Processing or implementation bottlenecks, also called hot spots, are capacity limitations due to processing intensive spots in the code, i.e., pieces of code that are frequently executed.

Resource interference bottlenecks

Resource interference bottlenecks, also called resource lock, are capacity limitations due to locked resources, i.e., a service puts a lock on a logical resource, such as a database lock, which in turn disables other services to execute concurrently.

Design bottlenecks

Design bottlenecks are capacity limitations due to system design limitations (a weak design). Design bottlenecks require major efforts in redesign and implementation to get resolved, if possible.

Architectural bottlenecks

Architecture bottlenecks, also called ultimate bottlenecks, are capacity limitations due to severely underestimated capacity requirements leading to an incorrect architecture for the system. Architecture bottlenecks can rarely be fixed. It is usually both cheaper and faster to replace a system with an architecture bottleneck. A typical example of architecture bottlenecks is an old single-threaded application that when moved to a modern multi-CPU multi-core hardware system, cannot utilize the additional processing resources.

6.4.5 Platform resource utilization characteristics

The platform resource utilization characteristics are indicators of to what level available platform resources can be utilized for production of a service or a mix of services. Platform resource utilization can be applied on an individual resource or a mix of resources.

The highest possible value for platform resource utilization is of course 100% of all resources used. In reality, as one hardware resource is used to its maximum there are still other types of resources unused. The difference between the least and the most utilized resource may be an indicator of a configuration bottleneck. The bigger the difference the worse is the bottleneck.

Platform resource utilization is measured on a system with a fixed amount of resources. The platform resource scalability characteristics are indicators of platform resource utilization for additional resources.

Efficiency characteristics for network resource utilization are slightly different measures with a different purpose. Efficient utilization of network bandwidth can be measured as a percentage of the effective transmission rate on a network component (the percentage of the nominal bandwidth used for successful transmission of application data).

An increase of the packet size will reduce the ratio between transmitted data and overhead data (data in a packet used by transport and network protocols to get the application data across the network) and consequently increase the effective transmission rate, i.e., lower the overhead data as a percentage of transmitted application data. However, an increase of the packet size also increases the probability of a transmission error followed by a retransmission of a failed packet. Therefore, the effective transmission rate must be reduced by the transmission rate spent on retransmitting data. From a certain point, any increase of packet size will have a negative impact on the effective transmission rate.

Another factor is the impact of the packet size on the packet switching capacity, which in turn is related to network congestion problems. With smaller packet sizes the amount of switched packets increase per time unit, which improves the likelihood of receiving data in time. Smaller packets can also be more evenly distributed across alternative network links and thereby shorten and reduce the consequences of network congestions.

6.4.6 Platform resource distribution characteristics

The platform resource distribution characteristics are indicators of how fast and evenly platform resources are distributed to requesting services. The platform resource distribution can be measured for an individual service or a mix of services.

A system with poor resource distribution runs out of one type of resource while there are still plenty of other resources. A system with poor resource distribution will also be less capable of handling sudden resource shortage situations following system resource outages. The platform resource distribution characteristics discussed here are of two kinds:

1. demand driven resource distribution
2. outage driven resource distribution

Demand-driven resource distribution characteristics are indicators of a system's ability to distribute available resources to demanding service requests.

Outage-driven resource distribution characteristics are indicators of a system's ability to redistribute available resources to demanding service requests after various outage situations. The objectives of outage driven resource distribution is to minimize the effects of various outage situations.

6.4.7 Platform resource scalability characteristics

The platform resource scalability characteristics are indicators of to what level additional system resources can be utilized for production of a service or a mix of services, i.e., resource utilization applied on additional resources. Platform resource scalability can be measured for an individual service or a mix of services.

The highest possible value for platform resource scalability is of course 100% of additional resources. However, there are some limitations to what is possible to reach set by the platform resource utilization, measured before the addition of resources. Few systems have a perfectly linear platform resource scalability. The effect of adding more resources of some kind is in reality limited by the capacity of other related resources. For example, the effect of adding more processing power and memory to a system is in reality limited by the transmission capacity between the CPU and memory.

7 Measured objects

7.1 Measured services

When requesting information about the performance of a car figures for top speed, acceleration, maximum load, mileage, service intervals, etc., are usually provided.

The performance figures apply to the tested car as a whole, i.e., on system level or top level of the tested system. However, the measured performance of a car is a result of the car design and the performance of the various components of the car involved in producing its services. Examples of the components of a car contributing to measured powerfulness values are the performance of the engine, the transmission system, the electrical system, etc. To design the performance of the car we need to measure and evaluate the performance of its components and how the components interact.

A similar approach can be applied on a computer system. At the system (application) level the performance of system service delivery is measured, such as transaction processing capacity, or response time of various services, etc.

7.2 Measured components

Similar to a car, the measured performance of a computer system service is not atomic, but the end result of the performance of many levels of processing services or components. The performance of application services depends on the performance of requested middleware services. The performance of middleware services depends on requested operating system services. The performance of operating system services depends on the performance of requested hardware components services, etc.

The performance of each component involved in delivering an application service is not required to measure the performance of an application service. However, performance is built from inside out, i.e., to design an application that meets defined performance requirements, the measure and control of the performance of all components is required.

7.3 Service concepts

A distributed system provides its services to users over a published interface. If a service is general enough it can be used as a shared service by multiple applications. Access to a service in a distributed system is open to any client that has the authority to use the service and is authenticated as a valid user. The rationale of this concept is reuse of software as an online service.

7.3.1 Service and component performance

An application service is normally resolved by a set of internal services.

The measured performance of a system service is the aggregated result of all components (hardware and software) involved in, and contributing to, the results.

The performance of an application service depends on the performance of requested middleware services. The performance of the middleware services depend on requested operating system services. The performance of operating system services depend on the performance of requested hardware components services, etc. The track can basically be followed down to execution of CPU instructions.

7.3.2 Service topology and topology performance

The service topology describes how an application service is dependent on other application services to resolve its task.

Performance tests of service topology focus on the responsiveness of distributed services processing components. Performance tests of service topology cover, for example:

- tests of latency in accessing distributed services;
- tests of capacity in accessing distributed services.

A distributed system is not only built on several layers of services, but each layer of services may also be distributed across a large number of system components (servers).

The system topology describes how the system components are interconnected and the requirements to traverse the system between any two components.

For example (see Figure 6) the registration of an IMS user is initiated by the UE sending a REGISTER request on the IMS Gm interface (SIP) to the user's servicing CSCF (S-CSCF). The S-CSCF in turn requests services from the operators HSS to identify and authenticate the user and set up secured IPsec channels to the UE. This is achieved by sending a request on the IMS Cx interface (Diameter) to the HSS.

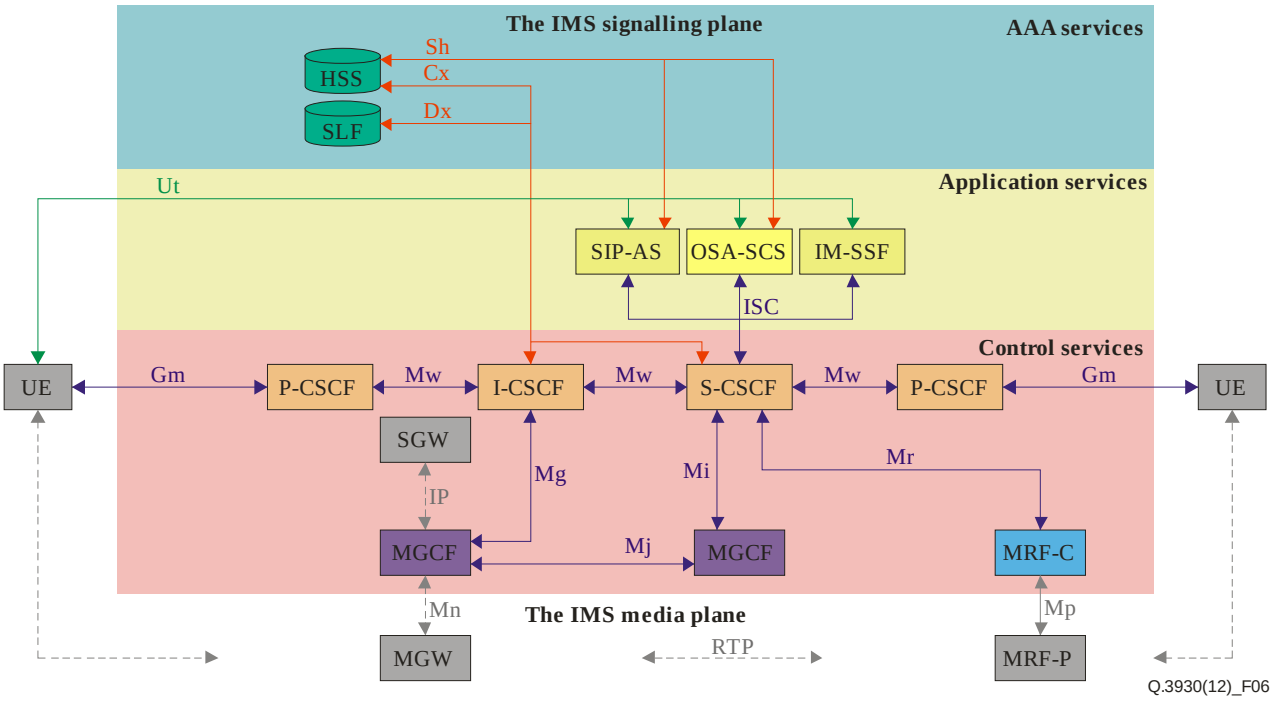

Figure 6 – Example of service topology from IMS

7.4 Service characteristics

Service characteristics are service attributes that determine how performance tests of a service should be designed, and include:

- service initiation characteristics;
- service duration characteristics;
- service resource and load characteristics;
- service design characteristics;
- service flow characteristics.

The purpose of specifying the characteristics of each service is to design correct performance test cases. A well-written specification of the characteristics of a tested service improves the understanding of what to measure and how.

7.4.1 Service initiation characteristics

Service initiation characteristics describe how a service is invoked. There are two types of services in this context:

- pulled services;
- pushed services.

Pulled services

A pulled service is initiated by a client in a service request and responded to by the service in one or more response messages.

Pushed services

A pushed service is initiated by a service in a service request to a subscribing client and responded to by the client in one or more response messages. A pushed service is usually triggered by an event, usually causing a state change in the service. The service is distributed to any client with a pending subscription to the service.

An example of a pushed service is the following: An IMS user with an active publication sends a PUBLISH of a status change to the publication server. The publication server updates the publication and sends NOTIFY messages to all active subscribers of the publication.

7.4.2 Service duration characteristics

Service duration characteristics describe the combination of stateless or stateful services and the service duration (See Figure 7):

- services with short duration;
- services with variable duration;
- services with long duration.

System services with short duration

For system services with short duration a short response time is essential. Services are usually stateless. A service request of this kind usually has a timer at the application protocol level that terminates the request if no response message can be returned within a standard response time limit. Examples of services with short duration are any type of simple request-response service, such as web browsing or a Google search.

System services with variable duration

For system services with variable duration the requested service has no time constraints and consequently changes from case to case. Services with variable duration are stateful. An example of a service with variable duration is a call, where ring time and/or hold time (the actual duration of the conversation) varies from call to call.

System services with long duration

For system services with long duration the requested service is usually a prerequisite for other subsequent services during a user session and lasts consequently until the list of subsequent services is finished. Services are stateful. A service with long duration usually has a timer, for security reasons, that expires when no activities are registered during a specified time. An example of a service with long duration is a user session or a subscription to presence status.

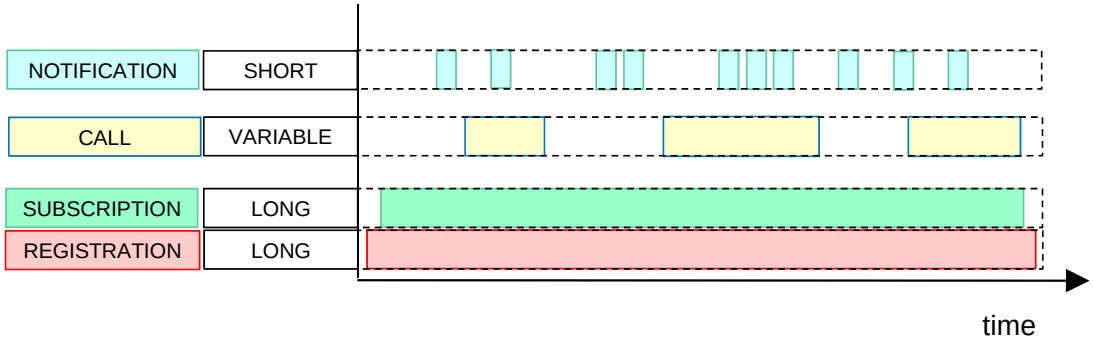

Figure 7 – Examples of different service durations

7.4.3 Service resource and load characteristics

Service resource characteristics describe the mix of requirements on the following hardware resources:

- processing (CPU) requirements;
- storage (memory) requirements;
- transmission (bandwidth) requirements.

Service load characteristics describe the type of system load caused by the resource profile and the duration of a service. The load profile has two values: services causing active load and services causing passive load.

Services causing active load

A service with short or variable duration, such as a web transaction or streaming multimedia in a call, typically causes an active load on system resources. A service causing active load on a system is characterized by:

- high requirements on processing resources (CPU) or transmission resources; and
- variable requirements on memory space.

The processing capacity for services causing active load is limited by processing and transmission resources.

Services causing passive load

A service with long duration, such as a user session or a user subscription, typically causes a passive load on system resources. A service causing passive load on a system is characterized by:

- low requirements on processing resources (CPU) or transmission resources; and
- memory space, usually related to the context of the service, is occupied throughout the duration of the service, which can be long.

The processing capacity for services causing passive load is limited by available memory for service. Even small amounts of memory per service request can lead to large demands on memory. The registration service in an IMS system where each pending user registration occupies 25 kilobytes will need 25 gigabytes of memory for one million concurrently registered users.

7.4.4 Service design characteristics

Service design characteristics (see Figure 8) describe the complexity of a service. There are many types of service constructions. In this context, the following types will be reviewed:

- single-step services;
- multi-step services;
- composite services.

Single-step services

A single-step service contains a single request with related responses on a specified interface.

Multi-step services

A multi-step service contains several requests with related responses on a specified interface.

Composite services

A composite service contains several logically separate subservices, where each subservice has a defined interface with a set of one or more requests and their related responses.

Figure 8 shows a service design containing several logically separate subservices using separate interfaces.

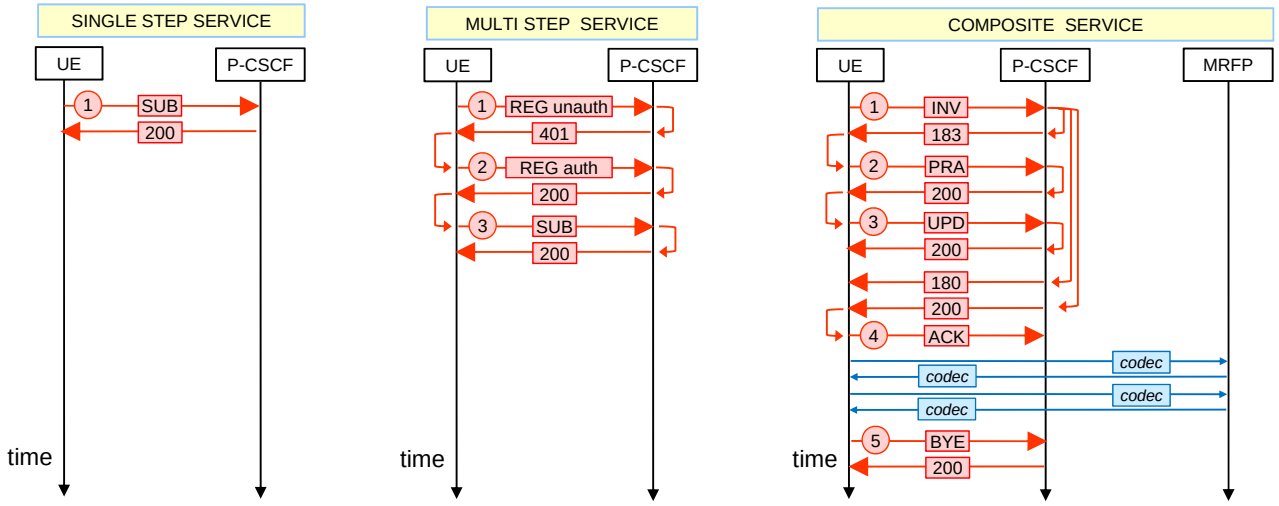

Figure 8 – Examples of different service designs

7.4.5 Service flow characteristics

Service flow characteristics describe how a service is communicated. Two types of service flows (see Figure 9) are discussed in the present Recommendation:

- transaction services;
- streamed services.

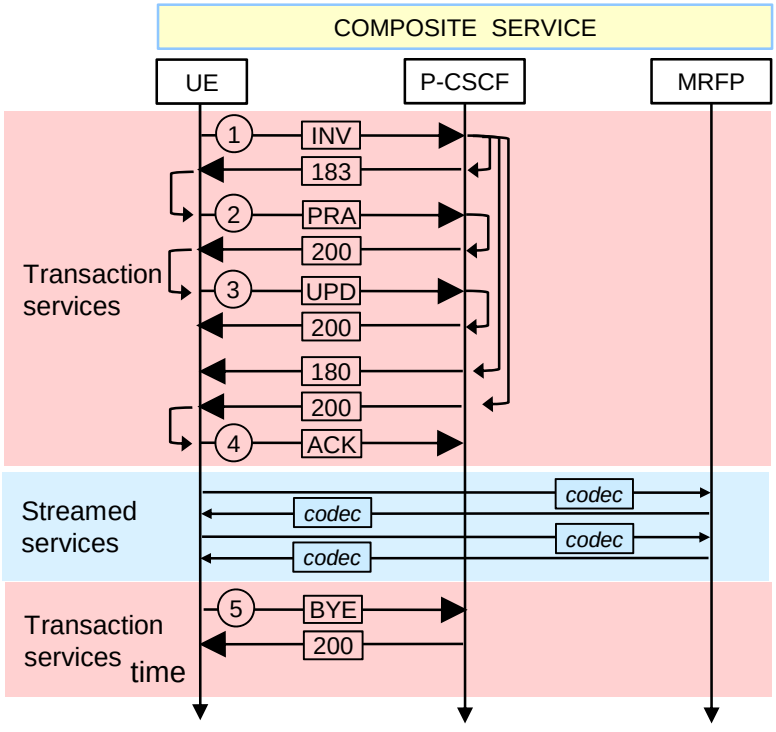

Figure 9 – Examples of transaction service and streamed service

Transaction services

A transaction service is communicated in a relatively limited number of interactions between client and server. Transaction services are often tied to some kind of processing of centralized services or databases.

Streamed services

A streamed service is communicated in a continuous flow of interactions between client and server that can last from a few seconds up to several hours and more. A streamed service can be bidirectional such as a multimedia call between two persons or unidirectional such as an IPTV media transfer. Performance requirements and performance attributes of a streamed service are quite different from a transaction service.

7.5 Service interfaces

The services of a system are accessible on one or more system interfaces, where different services might use different interfaces. An interface between the system under test and the performance test tools can be an application programming interface (an API) or a communications protocol interface.

7.5.1 Application programming interfaces

An application programming interface provides a library of functions for a call-based dialogue between the tested system and the test tools. An application programming interface hides the actual network between the client and the server. The actual network between the client and the server is in most cases a communications protocol interface. See Figure 10.

7.5.2 Communication protocol interfaces

A communications protocol interface is a protocol stack with protocols from the following three of the OSI layers:

1) application layer protocols (OSI layer 7)

2) transport layer protocols (OSI layer 4)

3) network layer protocols (OSI layer 3)

Examples of application layer protocols are HTTP, SOAP, SIP, Radius, Diameter, DHCP, etc., or subsets thereof.

Examples of transport layer protocols are TCP, UDP, SCTP, etc.

Examples of network layer protocols are IP (IPv4 and/or IPv6), IPsec, etc.

The client requests a service from the server by sending a message formatted according to the application layer protocol used by the communication protocol interface. A communications protocol interface usually defines a subset of the used application layer protocol.

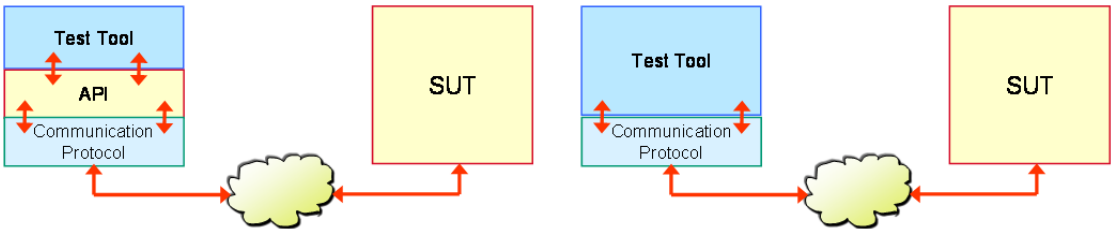

Figure 10 – Test tools where the SUT interface is an API (left), or a communication protocol interface (right)

8 Performance measurement data objectives and attributes

8.1 Performance metric objectives

The following set of characteristics applies to good and useable performance metrics:

- understandable;
- reliable;

- accurate;
- repeatable;
- linear;
- consistent;
- computable.

Understandability characteristics

Measured values for a good performance metric are easy to interpret and understand. Understandability is an important characteristic of performance metrics for presentations and reports.

Reliability characteristics

A good performance metric is reliable if the measured values in a performance test are in accordance with the measured values in real production. Such performance metrics can be trusted. Reliability is an important characteristic for performance metrics used as input to other performance metrics.

Accuracy characteristics

A good performance metric is accurate or precise if the measured values are very close to real values. The actual deviations from real values express the precision of a performance metric. Accuracy and precision is an important characteristic for performance metrics used as input to other performance metrics. Computed performance metrics based on input from performance metrics with varying accuracy and precision are of questionable value.

Repeatability characteristics

A good performance metric delivers the same value every time a performance test is repeated identically. Repeatability is an important characteristic for performance metrics used in regression testing.

Linearity characteristics

A performance metric with a linear characteristic is a metric that has a linear relation to values that are proportional to changes in the measured object. This makes the performance metric easier to understand. Another aspect of linearity is that mixing linear and non-linear performance metrics in computations of other performance metrics is of questionable value. Linearity is therefore an important characteristic for performance metrics used as input to other performance metrics.

Consistency characteristics

A performance metric is consistent if the metric units and the definitions of metric units are the same on different systems. When the units of a metric are not consistent on different platforms, performance metric values cannot be compared. Linearity is therefore an important characteristic for performance metrics used as input to other performance metrics. Consistency is therefore an important characteristic for performance metrics used in benchmarks.

Computability characteristics

A good performance metric has precise definitions of measurement units, measurement accuracy and measurement methods. This is a prerequisite for using the measurement variable in the various computations of performance. Computability is therefore an important characteristic for performance metrics used as input to other performance metrics.

8.2 Measurement data attribute sets

To support and verify the performance metric objectives above, performance measurement data have several sets of attributes (Figure 11) telling different aspects of what they represent and how they can be used, such as:

- processing attributes or metric types;
- identification attributes or metric identifiers;
- unit attributes or metric units; and
- conditional attributes or measurement conditions (reproducible).

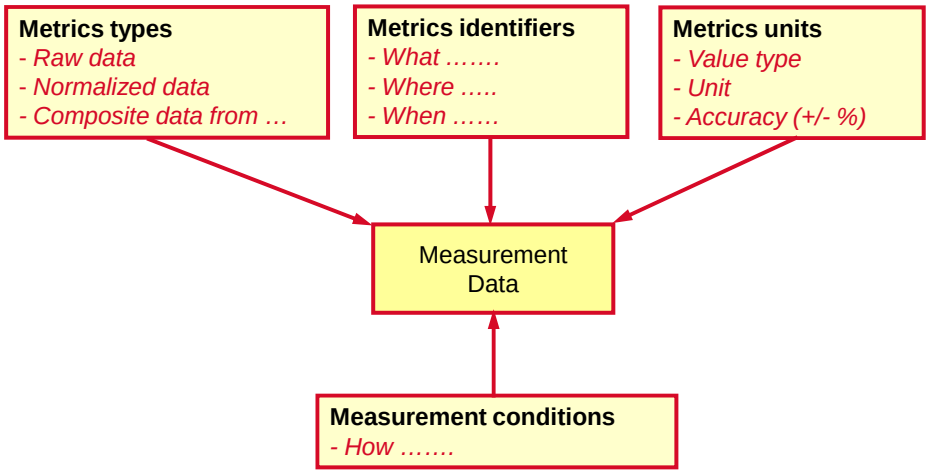

Figure 11 – Measurement data attribute sets

8.3 Processing attributes or metric types

Processing attributes describe how a metric variable value is derived and makes it understandable and reproducible. Metrics can be based on:

- raw performance data;
- normalized performance data;
- transformed performance data;
- composite performance data.

8.3.1 Metrics based on raw performance data

Raw performance metrics are performance data collected during a performance test and recorded in native form, i.e., data are not yet processed or formatted in any way, such as collected response time values for a specific type of transaction.

8.3.2 Metrics based on normalized performance data

Normalized performance metrics are performance data transformed to a common norm, for example, transactions per second or rejected requests per million service requests, etc.

Time-based metrics

In graphs, performance metrics are usually displayed with metrical values on the y-axis and recording time on the x-axis. We call this type of presentation time-based metrics, for example, variations in response time during a test.

Value-based metrics

Presentation of metrics can also be based on figures other than time, i.e., show other variables on the x-axis. Presentation of metrics is value-based when the metrical values are displayed on the x-axis and frequencies of metrical values are displayed on the y-axis.

An example of value-based metrics is response time distribution, usually described in a histogram with response time interval values on the x-axis and the frequency or percentage of each response time interval on the y-axis.

8.3.3 Metrics based on transformed performance data

Transformed performance metrics are performance metrics processed into other logically related performance metrics, such as response time data transformed into average response time metrics or standard deviation of response time values.

8.3.4 Metrics based on composite performance data

Composite performance metrics are based on the processing of multiple performance data sources. Input to composite performance metrics can be any kind of computable performance data.

An example of composite performance metrics is resource usage per processed request of a service.

8.4 Identification attributes or metric identifiers

Identification attributes contain reference values that together make collected performance measurement data unique. There are three types of identification attributes:

- measurement type;
- measurement point;
- measurement recording time.

Performance data without identification attribute values are of questionable value with no references to what they represent or why.

8.4.1 Measurement type

Measurement type identifies the type or name of collected measurement data.

8.4.2 Measurement points

Measurement points identify where performance data are captured. There are two types of measurement points (see Figure 12):

1) external
2) internal

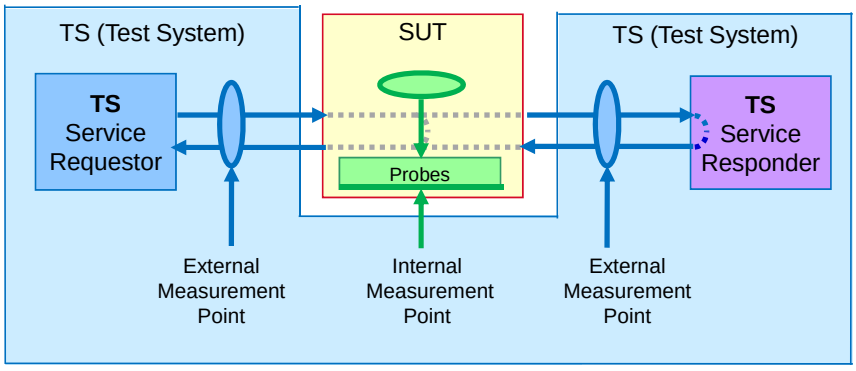

Figure 12 – External and internal measurement points

External measurement points

External measurement points are data collection locations outside the SUT, usually at the test tools.

At an external measurement point, performance data about the flow of requested services of different types and related service responses are collected.

An external measurement point can be a requested service or all types of responses to requested services including failures such as timeout or closed connections.

In this context, i.e., identification of performance measurement data, external measurement points are also processes producing composite performance metrics based on multiple sources of recorded performance data.

Internal measurement points

Internal measurement points are data collection locations inside the SUT. Data collection at internal measurement points is usually done by probes managed by the test tools.

At an internal measurement point, performance data are collected about how resources are managed under different load conditions inside the SUT. Internal measurement points can be global for a server, or local for a process group.

Internal measurement points can also be located inside a process group capturing data about resource management in application code, such as queues, object instances, etc.

8.4.3 Measurement recording time

Measurement recording time is usually a timestamp with high resolution identifying at what point in time a performance measurement value was recorded. The measurement recording time can be relative or absolute.

Relative time

Relative time shows elapsed time since the start of a performance test (time zero). There are several reasons for applying relative time in performance tests.

- Relative time enables a simple way of comparing different test runs of the same performance test. For instance, it is easy to see if a certain behaviour appears after a certain period of time in all performance tests.
- Relative time makes it easy to calculate elapsed time between different events in a performance test.

Absolute time

Absolute time is calendar time of a measurement recording. There are several reasons for applying absolute time in performance tests. Absolute time is preferred when there is no need for comparing different test runs or other kinds of analysis of product behaviour, such as monitoring service production. In monitoring service production it is important to see at what time different situations happen when they reappear, such as a repeated situation during the night at 02:30 every working day. In some situations, such as when a test project is geographically distributed on separate locations with different time zones, it is convenient to use absolute time from a single time zone such as GMT for all test sites.

8.5 Unit attributes or metric formats

Unit attributes make performance data comparable, accurate and computable. Performance data have several attributes telling different aspects of what a measurement figure represents:

- unit type attributes;
- unit resolution attributes;

- unit accuracy attributes;
- unit recording attributes.

Unit type attributes

Examples of unit type attributes are elapsed time, timestamps, counters, percentages or other units.

Unit resolution attributes

Examples of unit resolution attributes are time in days, hours, seconds or milliseconds.

Unit accuracy attributes

Examples of accuracy attributes are deviations from correct values (+/-, %).

Unit recording attributes

Examples of unit recording attributes are accumulative or instantaneous values.

8.6 Conditional attributes

Conditional attributes are references to stated and actual conditions that apply on collected performance measurement data and consequently are of importance to make performance data reproducible.

There are two types of conditional attributes:

- requested conditions;
- actual conditions.

8.6.1 Requested conditions

Requested condition attributes are links to requested measurement conditions, i.e., stated requirements on the SUT and test system conditions for capturing performance data. Requested conditions can be internal or external.

External conditions

External conditions describe what the SUT should be exposed to during a performance test, i.e., the workload specifications.

Internal conditions

Internal conditions describe expected situations inside the SUT during a performance test, such as maximum CPU load, memory usage, etc.

8.6.2 Actual conditions

Actual conditions are links to collected measurement data about actual measurement conditions during a performance test. Actual measurement conditions include metrics such as load deviations – the differences between intended load and actual load during a performance test.

9 Abstract performance metrics

9.1 Abstract performance metrics and performance categories

Abstract performance metrics are formal representations of performance characteristics and, in this context, classified into three categories: powerfulness, reliability and efficiency.

9.2 Abstract powerfulness metrics

Abstract powerfulness metrics express measurements of volume and speed of service production.

Abstract powerfulness metrics have sub-categories for capacity, responsiveness and scalability.

9.2.1 Capacity metrics and related attributes

Capacity metrics express the maximum number of service requests handled by an SUT per time unit, where time unit usually is seconds.

Sustained arrival capacity

Definition:	A performance metric for the maximum number of service requests of some kind that can be accepted by the SUT per time unit continuously.
Measurement unit:	Counter value as frequency per time unit: number of requests/second.
Resolution:	1 request/second.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Recorded value normalized to number of requests per second.
Example:	Sustained arrival capacity of service "xx" is 2160 requests/second.

Peak arrival capacity

Definition:	A performance metric for the maximum arrival rate of service requests during a specified period of time.
Measurement unit:	Counter value as frequency per time unit: number of requests/second.
Resolution:	1 request/second.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Recorded value normalized to number of requests per second.
Example:	Peak arrival capacity of service "xx" is 3160 requests/second for up to 15 seconds.

Sustained throughput capacity

Definition:	A performance metric for the maximum number of service requests of some kind that can be completed by the SUT per time unit continuously.
Measurement unit:	Counter value as frequency per time unit: number of requests second.
Resolution:	1 request/second.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Recorded value normalized to number of requests per second.
Example:	Sustained throughput capacity of service "xx" is 2160 requests/second.

In progress active load capacity

Definition:	A performance metric for the maximum number of active load service requests of some kind concurrently in progress.

Measurement unit: Counter: number of concurrent requests.

Resolution: 1 request.

Accuracy:

Measurement recording: Number of requests per second and average response time.

Processing: Calculated as number of requests per second divided by average response time.

Example: In progress active load capacity of service "xx" is 300 requests.

In progress passive load capacity

Definition: A performance metric for the maximum number of passive load service requests of some kind concurrently in progress.

Measurement unit: Counter: number of concurrent requests.

Resolution: 1 request.

Accuracy:

Measurement recording: Accumulated value of service requests in progress per recording period.

Processing:

Example: In progress passive load capacity of service "xx" is 30 000 requests.

9.2.2 Responsiveness metrics and related attributes

Responsiveness metrics express some kind of time delay of a measured service. The metric units for responsiveness cover response time, response time percentiles, different kinds of latency, etc.

Average response time

Definition: A performance metric for the average elapsed time from sending a request until receiving a response.

Measurement unit: Elapsed time in seconds or milliseconds.

Resolution: 1 second or 1 millisecond.

Accuracy:

Measurement recording: Accumulated value per recording period of time and entire performance test time.

Processing: Accumulated response time value per recording period divided by number of completed requests.

Example: Average response time for service "xx" is 30 milliseconds.

Response time percentiles

Definition: A performance metric for the maximum response time recorded for a specified percentile value of all service requests. A response time percentile of 90% shows maximum response time for 90% of all service requests. A response time percentile can be calculated for the entire performance test period or a part thereof.

Measurement unit: Elapsed time in seconds or milliseconds.

Resolution: 1 second or 1 millisecond.

Accuracy:

Measurement recording: Accumulated value per recording period of time and entire performance test time.

Processing:	The highest recorded response time value for the specified percentage of processed service requests.
Example:	The maximum response time for the 90th percentile of service "xx" is 40 milliseconds.
	The maximum response time for the 95th percentile of service "xx" is 120 milliseconds.

Distribution latency

Definition:	A performance metric for the delay from receiving a request until passing the request to the next processing instance.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Accumulated latency time value per recording period divided by number of completed requests.
Example:	Average distribution latency time for service "xx" is 30 milliseconds.

Notification latency

Definition:	A performance metric for the delay in notifying a subscriber of a change in a subscribed object.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Accumulated latency time value per recording period divided by number of completed requests.
Example:	Average notification latency time for service "xx" is 12 milliseconds.

Disk access latency

Definition:	A performance metric for the average time to position the head on the correct cylinder, track and sector of a disk, i.e., the delay for positioning a disk for a read or write operation.
Measurement unit:	Elapsed time in milliseconds.
Resolution:	1 millisecond.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Calculated value from time to position disk head to desired cylinder plus time to rotate the disk a half turn.
Example:	Average disk access latency time is 4 milliseconds.

9.2.3 Scalability metrics and related attributes

Scalability metrics express the relation between hardware resource increases and related service capacity increases.

Scalability metrics can be measured for a single type of hardware resources or a balanced mix of hardware resources.

The measurement units for scalability metrics are service capacity increases in absolute numbers. The measurement units for capacity increases can also be percentage values, however, any percentage value depends on the current service capacity level if a fixed quantity of resources is added.

Service capacity per additional processing unit

Definition:	Performance metric for the increase of service capacity of a kind by adding a processing unit, such as a server, a CPU or more cores per CPU. The metric value is expressed as additional service requests processed per time unit. The scalability metric applies for services with active load.
Measurement unit:	Counter value as frequency per time unit: number of requests/second.
Resolution:	1 request/second.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Calculated as the sustained throughput capacity for the SUT with an additional processing unit, minus the sustained throughput capacity for the SUT with no additional processing unit.
Example:	Service capacity per additional processing unit for service "xx" is 850 requests/second.

Service capacity per additional memory unit

Definition:	Performance metric for the increase of service capacity of a kind by adding a memory unit, such as a DIMM. This scalability characteristic applies for services with passive load.
Measurement unit:	Counter: number of requests.
Resolution:	1 request.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Calculated as the in progress passive load capacity for the SUT with an additional processing unit, minus the in progress passive load capacity for the SUT with no additional memory unit.
Example:	Service capacity per additional memory unit for service "xx" is 2 000 requests.

9.3 Abstract reliability metrics

Abstract reliability metrics express measurements of how predictable a system's service production is.

Abstract reliability metrics have sub-categories for quality of service, stability, availability, robustness recovery and correctness.

9.3.1 Quality-of-service metrics and related attributes

Quality-of-service metrics are closely related to stability and availability metrics for service production.

The measurement units for quality-of-service metrics are events per number of service requests (such as 1 000 000), or events per production time unit (usually one year).

Service fail rate

Definition:	A performance metric for frequencies of denied services.
Measurement unit:	Counter value as frequency per mega service request (MSR): decimal value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated value for the entire performance test time.
Processing:	A composite metric value based on the number of rejected requests and the total number of requests.
Example:	Service fail rate per MSR for service "xx" is 4.2 requests.

9.3.2 Stability metrics and related attributes

Stability metrics express changes in measured performance figures for powerfulness characteristics and/or efficiency characteristics over long periods of time. Changes in performance figures are measured in two ways:

1) the response time spread, i.e., the difference between the highest and lowest response time value. A high value for response time spread means that actual service response time is unpredictable and therefore unstable.
2) the response time trend, i.e., an indication that response time values are increasing over time. An increasing response time trend is an indicator of increasing resource usage over time.

Service response time spread

Definition:	A performance metric for response time variations over a long period of time of constant load. The service response time spread is a metric for the difference between the shortest and longest average response time value during a test. A small service response time spread value is an indicator of a system with a stable service production. A large service response time spread value is an indicator of a system with unpredictable behaviour.
Measurement unit:	Elapsed time in milliseconds: response time difference.
Resolution:	1 millisecond.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	A calculated metric value based on the highest and lowest average response time value per recording period of time and entire performance test time.
Example:	The maximum response time spread for service "xx" is 13 milliseconds.

Service resource usage spread

Definition:	A performance metric for resource usage variations over a long period of time of constant load. The service resource usage spread is a measure of the difference between the highest and lowest average resource usage value during a test. A small service resource usage

	spread value is an indicator of a system with a stable service production. A large service resource usage spread value is an indicator of a system with unpredictable behaviour.
Measurement unit:	Resource usage difference: depending on type of resource.
Resolution:	Depending on type of resource.
Accuracy:	Depending on type of resource.
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	A calculated metric value based on the highest and lowest resource usage value per recording period of time and entire performance test time.
Example:	The maximum CPU usage spread for service "xx" is 2 milliseconds.

Service response time trends

Definition:	A performance metric for response time trends over a long period of time of constant load. The trend may indicate an increasing or a decreasing service response time. The service response time trend is presented as a probability figure in the range 0.0 to 1.0, where 0.0 indicates no trend and 1.0 a very strong trend.
Measurement unit:	Probability value: decimal value in the range 0.0 to 1.0.
Resolution:	Five decimals.
Accuracy:	
Measurement recording:	Average response time value per recording period of time.
Processing:	A metric value based on a recorded sequence of average response time values that cover the entire test period.
Example:	The response time trend for service "xx" is 0.02, i.e., no trend identified.

Service resource usage trends

Definition:	A performance metric for resource usage trends over a long period of time of constant load. The trend may indicate increasing or a decreasing service resource usage. The service resource usage trend is presented as a probability figure in the range 0.0 to 1.0, where 0.0 indicates no trend and 1.0 a very strong trend.
Measurement unit:	Probability value: decimal value in the range 0.0 to 1.0.
Resolution:	Five decimals.
Accuracy:	
Measurement recording:	Average resource usage value per recording period of time.
Processing:	A metric value based on a recorded sequence of average resource usage values that cover the entire test period.
Example:	The resource usage trend for service "xx" is 0.02, i.e., no trend identified.

9.3.3 Availability metrics and related attributes

Availability metrics for software services express frequency rates of service request errors, frequency rates of correctly processed service requests, or probabilities of service request errors or correctly processed service requests. The measurement units for frequency rates are number of

events per kilo service request (KSR), or MSRs. The measurement units for probabilities are percentage values.

Availability metrics for hardware express estimated operational time for a device. The measurement units for hardware operability are hours.

Service rejection rates

Definition:	A performance metric for frequencies of rejected service requests. Service rejection rate is based on the number of rejected requests and the total number of requests. Service rejection rate is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): decimal value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated number of rejected requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of rejected requests and the number of processed requests.
Example:	Service rejection rate for service "xx" is 0.4/MSR.

Service rejection probability

Definition:	A performance metric for the probability of a service request to be rejected. Service rejection probability is based on the number of accepted requests and the total number of processed requests. Service rejection probability is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): percentage value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated number of rejected requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of rejected requests and the number of processed requests.
Example:	The service rejection probability for service "xx" is 0.0002%.

Service acceptance rates

Definition:	A performance metric for frequencies of accepted service requests. Service acceptance rate is based on the number of accepted requests and the total number of processed requests. Service acceptance rate is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): decimal value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated number of accepted requests and accumulated number of processed requests for the total test execution time.

Processing: Composite metric value based on the number of accepted requests and the number of processed requests.

Example: The service acceptance rate for service "xx" is 999 993.2 per MSR.

Service acceptance probability

Definition: A performance metric for the probability of a service request to be accepted. Service acceptance probability is based on the number of accepted requests and the total number of processed requests. Service acceptance probability is normalized per MSR.

Measurement unit: Counter value as frequency per mega service request (MSR): percentage value.

Resolution: 0.1 request/MSR.

Accuracy:

Measurement recording: Accumulated number of accepted requests and accumulated number of processed requests for the total test execution time.

Processing: Composite metric value based on the number of accepted requests and the total number of processed requests.

Example: The service acceptance probability for service "xx" is 99.9999%.

Mean time between failures (MTBF)

Definition: A performance metric for the mean time between failures of a hardware component or a hardware system. Mean time between failures is a statistical metric value expressed in number of hours.

Measurement unit: Elapsed time.

Resolution: Hours.

Accuracy: Depending on measured component.

Measurement recording: Depending on measured component.

Processing: A statistical value based on a number of observations depending on the measured component.

Example: The MTBF figure for disk "xx" is 220 000 hours.

9.3.4 Robustness metrics and related attributes

Robustness metrics express consequences in service production due to a specified condition or set of conditions. Robustness metrics can be calculated for service capacity, service responsiveness or service availability.

Service capacity impact

Definition: A performance metric for the impact on service capacity due to a specified condition or set of conditions expressed as a percentage value of total capacity decrease.

Measurement unit: Percentage value.

Resolution: 0.1%.

Accuracy:

Measurement recording: Accumulated per recording period of time and entire performance test time.

Processing: The capacity reduction is calculated as the sustained throughput capacity for the SUT under normal conditions, minus the sustained

	throughput capacity for the SUT when tested conditions apply. The service capacity impact is then calculated as the capacity reduction as a percentage of sustained throughput capacity for the SUT under normal conditions.
Example:	Service capacity impact on service "xx" is a reduction of 47.2%.

Service responsiveness impact

Definition:	A performance metric for the impact on service responsiveness due to a specified condition or set of conditions expressed as a percentage value for average response time increase.
Measurement unit:	Percentage value.
Resolution:	0.1%.
Accuracy:	
Measurement recording:	Accumulated per recording period of time and entire performance test time.
Processing:	Response time increase is calculated as the average response time for the SUT when tested conditions apply, minus the average response time for the SUT under normal conditions. The service responsiveness impact is then calculated as the response time increase as a percentage value of average response time for the SUT under normal conditions.
Example:	Service responsiveness impact on service "xx" is a response time increase of 215.0%.

Service availability impact

Definition:	A performance metric for the impact on service availability due to a specified condition or set of conditions expressed as a percentage value for service rejection increase.
Measurement unit:	Percentage value.
Resolution:	0.1%.
Accuracy:	
Measurement recording:	Accumulated per recording period of time and entire performance test time.
Processing:	Service rejection increase is calculated as the service rejection rate when tested conditions apply, minus the service rejection rate under normal conditions. The service availability impact is then derived from the service rejection increase as a percentage value of the service rejection rate under normal conditions.
Example:	Service availability rate impact on service "xx" is a service rejection increase of 32.4%.

9.3.5 Recovery metrics and related attributes

Recovery metrics express different aspects of recovery of a specified situation or set of situations. Robustness metrics can be calculated for detection of a situation, correction of the consequences of a situation and restart after completed correction. Correction could be anything from hardware repair or replacement to software or data recovery.

Detection time

Definition:	A performance metric for time to identify a specified condition or set of conditions.

Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	Depending on situation.
Measurement recording:	Measured time from the SUT is set to stated conditions until log messages are recorded.
Processing:	No processing requirements.
Example:	Average time to detect situation of type "xx" is 2 seconds.

Partial system restart time

Definition:	A performance metric for partial restart of system services after an outage.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	Depending on situation.
Measurement recording:	Measured time from when service becomes unavailable until service request traffic can be resumed.
Processing:	No processing requirements.
Example:	Average time to restart after situation of type "xx" is 25 seconds.

Total system restart time

Definition:	A performance metric for restart of a system after an outage that requires total restart.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	Depending on situation.
Measurement recording:	Measured time from when the SUT becomes unavailable until service request traffic can be resumed.
Processing:	No processing requirements.
Example:	Average time to do a total restart of the system is 325 seconds.

Application restart time

Definition:	A performance metric for time restart of a system after system software updates.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	Depending on situation.
Measurement recording:	Measured time from when the SUT update is ready until service request traffic can be resumed.
Processing:	No processing requirements.
Example:	Average time to do a total restart of the system after software update is 75 seconds.

Service take-over time

Definition:	A performance metric for restart of system services on a back-up system.
Measurement unit:	Elapsed time in seconds or milliseconds.

Resolution:	1 second or 1 millisecond.
Accuracy:	Depending on situation.
Measurement recording:	Measured time from when the SUT becomes unavailable until service request traffic can be resumed.
Processing:	No processing requirements.
Example:	Average time to resume operations on a backup system is 175 seconds.

9.3.6 Correctness metrics and related attributes

Correctness metrics express frequency rates of service request errors, frequency rates of correctly processed service requests, or probabilities of service request errors or correctly processed service requests. The measurement units for frequency rates are number of events per kilo service request (KSR), or MSR. The measurement units for probabilities are percentage values.

Service error rate

Definition:	A performance metric for frequencies of incorrectly processed service requests. Service error rate is based on the number of incorrectly processed requests and the total number of processed requests. Service error rate is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): decimal value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated number of incorrectly processed requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of incorrectly processed requests and the number of processed requests.
Example:	Service error rate for service "xx" is 0.4/MSR.

Service correctness rate

Definition:	A performance metric for frequencies of correctly processed service requests. Service correctness rate is based on the number of correctly processed requests and the total number of processed requests. Service correctness rate is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): decimal value.
Resolution:	0.1 request/MSR.
Accuracy:	
Measurement recording:	Accumulated number of correctly processed requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of correctly processed requests and the number of processed requests.
Example:	The service correctness rate for service "xx" is 999 998.5/MSR.

Service error probability

Definition:	A performance metric for the probability a service request is incorrectly processed. Service error probability is based on the number of incorrectly processed requests and the total number of processed requests. Service error probability is normalized per MSR.
Measurement unit:	Counter value as frequency per Mega service request (MSR): percentage value.
Resolution:	0.0001.
Accuracy:	
Measurement recording:	Accumulated number of incorrectly processed requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of incorrectly processed requests and the number of processed requests.
Example:	The service error probability for service "xx" is 0.0001%.

Service correctness probability

Definition:	A performance metric for the probability a service request is correctly processed. Service correctness probability is based on the number of correctly processed requests and the total number of processed requests. Service acceptance probability is normalized per MSR.
Measurement unit:	Counter value as frequency per mega service request (MSR): percentage value.
Resolution:	0.0001.
Accuracy:	
Measurement recording:	Accumulated number of correctly processed requests and accumulated number of processed requests for the total test execution time.
Processing:	Composite metric value based on the number of correctly processed requests and the number of processed requests.
Example:	The service correctness probability for service "xx" is 99.9999%.

9.4 Abstract efficiency metrics

Abstract efficiency metrics express measurements of service production dependencies on resources and service production utilization of resources. Efficiency is measured on:

- service level, i.e., the tested application; and
- platform level, i.e., for hardware and software supporting the tested application.

Abstract efficiency metrics have sub-categories for service resource usage, service resource linearity, service resource scalability, platform resource utilization, platform resource distribution and platform resource scalability.

9.4.1 Service resource usage metrics and related attributes

Service resource usage metrics express resource usage per processed service request or fixed amount of service requests of a kind. The measurement unit is in absolute figures or as a percentage value of available resources.

Service resource usage is usually calculated per processed service request or batches thereof, such as 1 000 service requests. Measured resources can be physical (hardware) or logical (software).

CPU usage per service request

Definition:	A performance metric for used CPU resources per processed service request of some kind.
Measurement unit:	CPU usage in time.
Resolution:	Milliseconds or microseconds depending on service type.
Accuracy:	Depending on service type.
Measurement recording:	Accumulated value for CPU usage and number of service requests.
Processing:	Composite metric value based on the accumulated CPU time and the total number of processed requests.
Example:	CPU usage per service request for service "xx" is 1.23 milliseconds.

Memory usage per service request

Definition:	A performance metric for used memory resources per processed service request of some kind.
Measurement unit:	Memory used.
Resolution:	Number of KB (kilobytes).
Accuracy:	Depending on service type.
Measurement recording:	Instantaneous value for allocated memory and number of concurrent service requests.
Processing:	Instantaneous value for allocated memory resources and the concurrent number of processed requests.
Example:	Memory usage per service request for service "xx" is 270 KB.

9.4.2 Service resource linearity metrics and related attributes

The service resource linearity characteristics are indicators of a system's ability to use a constant amount of resources for the production of a service regardless of the actual load level on the system.

Service resource linearity metrics express the probability of an identified trend in resource usage correlated with an increase in service request rate. The measurement unit for a trend is a probability value for the correctness of the trend, where 0% means no identifiable trend and 100% means a guaranteed or reliable trend.

CPU usage trend

Definition:	A performance metric for CPU usage trends when the number of service requests per time unit increase. The trend may indicate an increasing or a decreasing CPU usage. The CPU usage trend is presented as a probability figure in the range 0.0 to 1.0, where 0.0 indicates no trend and 1.0 a very strong trend.
Measurement unit:	Probability of trend: percentage value.
Resolution:	0.00001.
Accuracy:	
Measurement recording:	Average CPU usage value per recording period of time.
Processing:	A metric value based on a recorded sequence of average CPU usage values that cover the entire test period.
Example:	The probability of an identified CPU usage trend for service "xx" is 2.0%.

Memory usage trend

Definition:	A performance metric for memory usage trends when the number of service requests per time unit increase. The trend may indicate an increasing or a decreasing memory usage. The memory usage trend is presented as a probability figure in the range 0.0 to 1.0, where 0.0 indicates no trend and 1.0 a very strong trend.
Measurement unit:	Probability of trend: percentage value.
Resolution:	0.00001.
Accuracy:	
Measurement recording:	Average memory usage value per recording period of time.
Processing:	A metric value based on a recorded sequence of average memory usage values that cover the entire test period.
Example:	The probability of an identified memory usage trend for service "xx" is 2.0%.

9.4.3 Service resource scalability characteristics

The service resource scalability characteristics are indicators of a system's ability to utilize additional platform resources for increased production of a service.

Service resource scalability metrics express the relation between hardware resource increases and related service capacity increases. Service resource scalability metrics can be measured for a single type of hardware resource or a balanced mix of hardware resources.

The measurement units for scalability metrics are service capacity increases in absolute numbers. The measurement units for capacity increases can also be percentage values, however, any percentage value depends on the current service capacity level if a fixed quantity of resources is added.

Service capacity per additional processing unit

Definition:	Performance metric for the increase of service capacity of a kind by adding a processing unit, such as a server, a CPU more cores per CPU. The metric value is expressed as additional service requests processed per time unit. The scalability metric applies for services with active load.
Measurement unit:	Counter value as frequency per time unit: number of requests/second.
Resolution:	1 request/second.
Accuracy:	
Measurement recording:	Accumulated value per recording period of time and entire performance test time.
Processing:	Calculated as the sustained throughput capacity for the SUT with an additional processing unit, minus the sustained throughput capacity for the SUT with no additional processing unit.
Example:	Service capacity per additional processing unit for service "xx" is 850 requests/second.

Service capacity per additional memory unit

Definition:	Performance attributes for the increase of service capacity of a kind by adding a memory unit, such as a DIMM. This scalability metric applies to services with passive load.
Measurement unit:	Counter: number of requests.

Resolution: 1 request.

Accuracy:

Measurement recording: Accumulated value per recording period of time and entire performance test time.

Processing: Calculated as the in progress passive load capacity for the SUT with an additional processing unit, minus the in progress passive load capacity for the SUT with no additional memory unit.

Example: Service capacity per additional memory unit for service "xx" is 1 500 requests.

9.4.4 Platform resource utilization metrics and related attributes

Platform resource utilization metrics express the utilization level as percentage value of the resource total or as a quote between used resources such as the CPU and memory.

Platform resource utilization profile

Definition: Performance metric for the utilization of various resources at sustained throughput capacity of a service "xx" or a mix of services. To be comparable utilization of each resource is expressed as a percentage value of the total resource.

Measurement unit: Resource usage level: percentage value.

Resolution: 0.1.

Accuracy:

Measurement recording: Resource utilization percentage value at sustained throughput capacity for service "xx".

The resource utilization percentage value is recorded for a set of resources.

Processing: Convert resource usage level or resource usage quantity to a percentage value of the resource total.

Example: Utilization level per additional processing unit is 82% to 93%, depending on service.

CPU-to-memory usage ratio

Definition: A performance metric for resource usage ratio between the CPU and memory for a service or a service mix calculated as the quote of CPU usage level divided by memory usage level.

Measurement unit: Quote of CPU usage level (%) and memory usage level (%): decimal value.

Resolution: 0.00001.

Accuracy:

Measurement recording: Average CPU usage values and average memory usage values recorded during the entire test period.

Processing: A metric value based on a recorded sequence of average CPU usage values and average memory usage values that cover the entire test period.

Example: The CPU-to-memory memory ratio for service "xx" is 0.85.

Memory-to-CPU usage ratio

Definition:	A performance metric for resource usage ratio between the memory and CPU for a service or a service mix calculated as the quote of the memory usage level divided by the CPU usage level.
Measurement unit:	Quote of memory usage level (%) and CPU usage level (%): decimal value.
Resolution:	0.00001.
Accuracy:	
Measurement recording:	Average CPU usage values and average memory usage values recorded during the entire test period.
Processing:	A metric value based on a recorded sequence of average memory usage values and average CPU usage values that cover the entire test period.
Example:	The memory-to-CPU ratio for service "xx" is 0.90.

9.4.5 Platform resource distribution metrics and related attributes

The platform resource distribution characteristics are indicators of how fast and evenly platform resources are distributed to requesting services. The platform resource distribution can be measured for an individual service or a mix of services. The platform resource distribution characteristics discussed here are of two kinds.

- Demand-driven resource distribution characteristics are indicators of a system's ability to distribute available resources to demanding service requests. A performance metric for demand driven resource distribution is average queuing time-for requested resource; and
- Outage-driven resource distribution characteristics are indicators of a system's ability to redistribute available resources to demanding service requests after various outage situations. The objectives of outage driven resource distribution are to minimize the effects of various outage situations. A performance metric for outage driven resource distribution is average latency time for resource redistribution.

Average queuing time for requested resource

Definition:	A performance metric for the average elapsed time spent waiting for requested resources.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	
Measurement recording:	Average queue length to service and processing time in service.
Processing:	Average queuing time is calculated as (average queue length – 1) × processing time.
Example:	Average queuing time for resource "aa" for service "xx" is 3 milliseconds.

Average latency time for resource redistribution

Definition:	A performance metric for the average elapsed time spent waiting for requested resources.
Measurement unit:	Elapsed time in seconds or milliseconds.
Resolution:	1 second or 1 millisecond.
Accuracy:	

Measurement recording:	Average queue length to service and processing time in service.
Processing:	Average queuing time is calculated as (average queue length – 1) × processing time.
Example:	Average queuing time for resource "aa" for service "xx" is 3 milliseconds.

9.4.6 Platform resource scalability metrics and related attributes

The platform resource scalability characteristics are indicators of to what level additional system resources can be utilized for production of a service or a mix of services, i.e., resource utilization applied on additional resources. Platform resource scalability can be measured for an individual service or a mix of services.

The highest possible value for platform resource scalability is of course 100% of additional resources. However, there are some limitations to what is possible to reach set by the platform resource utilization measured before addition of resources. Few systems have a perfectly linear platform resource scalability. The effect of adding more resources of some kind is in reality limited by the capacity of other related resources. For example, the effect of adding more processing power and memory to a system is in reality limited by the transmission capacity between the CPU and memory.

Utilization level per additional processing unit

Definition:	Performance attributes for the predicted utilization of an added processing unit, such as a server, a CPU or more cores per CPU. This scalability metric applies to services with active load.
Measurement unit:	Resource utilization level: percentage value.
Resolution:	0.1.
Accuracy:	
Measurement recording:	Resource utilization percentage value at maximum throughput for service "xx". The percentage value is recorded for a set of frequently used services.
Processing:	Convert resource usage level or resource usage quantity to a percentage value of the resource total.
Example:	Utilization level per additional processing unit is 82.2% to 93.7%, depending on service.

10 Performance data processing

10.1 Steps in performance data processing

A major part of performance testing is the processing of all performance data collected during the performance test. Performance data are processed in the following sequence of steps:

1) collection and storage of raw performance data;
2) condensation and normalization of raw performance data;
3) performance data computations;
4) evaluation of performance data; and
5) presentation of performance data.

This is an abstract flow of performance data processing. The text does not address whether performance data are processed and presented during or after execution of a performance test, since this is regarded as a test tool implementation issue in this context.

10.2 Time series of performance data

Performance data as discussed in the following text represent sequences of measurement values (usually long sequences) recorded at different times during the measurement period. This is called a time series of measurement values (Figure 13).

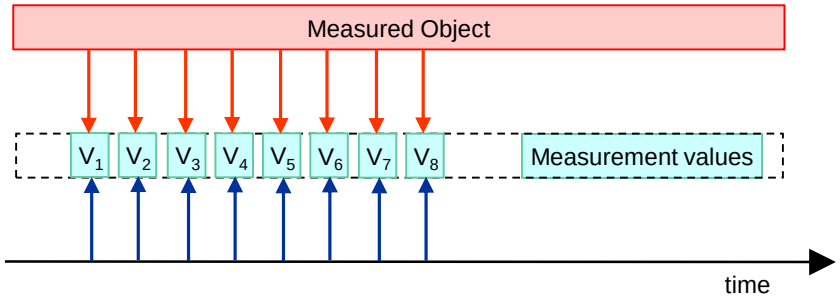

Figure 13 – A time series of measurement values

Management of time series of measurement data is critical to the usability of a performance test tool.

10.3 Collection and storage of raw performance data

Collection and storage of raw performance data is the first processing step. It is performed during execution of the performance test.

Raw performance data are performance data that are still in the native form in which they were collected and that have not yet been processed in any way. In reality, this means there are response-time measurements recorded for every response to a service request. Therefore, raw performance data occupy a lot of space and need to get condensed.

Additionally, raw performance data observations are hard to visualize in a graph. A plotting of millions of response time recordings usually looks like someone has spilled ink on a piece of paper. This is another reason for the processing of performance data in the following steps.

10.4 Condensation and normalization of raw performance data

Condensation and normalization of raw performance data is the second processing step. It can be performed during execution of the performance test or after the performance test has completed. This processing step is mandatory during the performance test execution, if a performance test-monitoring tool is used.

Condensation of performance data usually reduces the amount of data to a small fraction of the raw performance data.

Normalization of performance data is transformation to a common norm, for example, transactions per second or rejected requests per million service requests, etc.

10.5 Performance data computations

Performance data computation is the third processing step. All requested performance metrics requesting some kind of computation are processed in this step. The following shows three areas of performance data computations:

- trend analysis;
- comparisons of regressions tests;
- computations of composite performance metrics.

10.5.1 Trend analysis

Trend analysis of performance data is usually done for stability and availability tests. The purpose is to find traces of undesired behaviour that will cause severe disturbances in production if not handled at an early stage.

10.5.2 Comparisons of regression tests

Comparisons of regression tests are usually done to verify improvements in performance of a service.

10.5.3 Computations of composite performance metrics

Computations of composite performance metrics is processing of performance metrics based on multiple sources of recorded performance data. One example of composite performance metrics is resource usage per processed request of a service.

10.6 Evaluation of performance data

Evaluation of performance data is the fourth processing step. In this step, measured performance metrics are rated according to a set of rules expressing stated or desired performance goals. Evaluation of performance data can also be performed on the output from comparison of regression tests.

An evaluation of performance data results in some kind of verdict of the tested system's performance.

10.7 Presentation of performance data

Presentation of performance data is the fifth processing step. Presentation of performance data will convert processed and evaluated performance data into easily understood presentations, such as diagrams, tabular formats, etc.

11 General performance test concepts

11.1 Performance tests

Performance tests collect performance data that indicates the behaviour of a system under test under specified conditions in a controlled environment.

The goal of a performance test can be to find capacity limits of a system, or testing a system's ability to deliver services regardless of time, or many other performance characteristics. Performance measurements may cover an almost infinite number of performance characteristics of a system, but are for practical reasons limited to a carefully selected set of performance characteristics in most cases.

The conditions for captured performance data are caused by performance test tools generating artificial load on the tested system in the form of realistic service requests from simulated users.

11.2 Performance tests and system life cycle phases

Performance testing applies to all life cycle phases of a system from the first design steps of a system throughout real production of services. There are two main groups of performance tests:

1) pre-deployment performance tests
2) post-deployment performance tests

11.2.1 Pre-deployment performance test applications

Performance cannot be added to a system after it is designed and implemented. A system has to be designed and built for good performance to achieve stated performance goals. A general principle is therefore to start work on performance issues as early as possible during system design and development.

Pre-deployment performance testing takes place during system design and development and includes tests of:

1) intended performance goals
2) system design
3) system implementation
4) system integration

Performance tests of intended performance goals

Performance tests of intended performance goals are done to set realistic performance goals, i.e., to test if intended performance goals are possible to reach and if not, how they should be changed. The purpose is to transform intended performance goals to stated performance goals.

Performance tests of system design

Performance tests of system design are done to verify or modify stated performance objectives. The test results are also input to performance goals for implementation of individual elements of the system.

Performance tests of system implementation

Performance tests of system implementation are done to maintain control over stated performance objectives in developed code. The focus of performance tests of system implementation is powerfulness and efficiency of developed code.

Performance tests of reliability characteristics are of little value during this phase, since the developed code is not stable.

Performance tests of system integration

Performance tests of system integration are done to verify that measured performance of a system is maintained, when the system gets integrated with other related systems. Performance test objectives during system integration cover all specified performance characteristics of the system.

11.2.2 Post-deployment performance test applications

Post-deployment performance tests are measurements of a deliverable or a delivered system. Post-deployment performance tests include:

1) benchmarking or system evaluation
2) performance tests of system delivery
3) performance tests of service production

Benchmarking or system evaluation

Benchmarking is performance tests of a system based on a suite of standardized performance tests.

The main purpose of a performance benchmark is to produce a metric that can be rated and compared with the metric values produced by other systems using the same benchmark.

Performance tests of system delivery

Performance tests of system delivery are done to verify that stated performance requirements for a delivered system are at hand, when the system is integrated with other systems on the installation site. Performance test objectives in system delivery cover all specified performance characteristics of the system.

Performance tests of service production (performance monitoring)

Performance tests of service production (also referred to as performance monitoring) are done to verify that produced services are in accordance with stated quality requirements (quality of services).

Performance monitoring of service production can be:

- reactive; or
- proactive.

Reactive performance monitoring

Reactive performance monitoring aims at detecting and acting on situations after they have happened. Actions are based on situations identified from analysis of log files of different kinds from the system and service production.

Proactive performance monitoring

Proactive performance monitoring aims at detecting and acting on identified situations or trends that might evolve into severe disturbances or a disaster in service production before the situations get critical.

11.3 Performance test objectives

Performance test objectives describe the reasons for doing performance tests. Performance test objectives can be confirmative or explorative. A performance test execution can be both confirmative and explorative.

11.3.1 Confirmative performance tests

Confirmative performance tests are done to verify stated performance objectives.

A confirmative performance test case can be to verify required throughput capacity for a specific service or mix of services. Other cases of confirmative performance tests are regression testing.

Examples of confirmative performance tests follow.

- Has the tested system processing capacity for the specified load?
- Does the system respond fast enough under specified load conditions?
- Has the tested system processing capacity for the specified load?
- Does the system handle requested services continuously?
- Does the system deliver correct results under heavy load?

11.3.2 Explorative performance tests

Explorative performance tests are done to get an understanding of the behaviour of a system under specified conditions, or to find the performance limitations.

An example of explorative performance tests could be to find the maximum throughput capacity for a specific service or mix of services under specified conditions such as maximum CPU load.

Examples of explorative performance tests follow.

- Does the system respond fast enough under specified load conditions?

- Has the system processing limitations due lack of certain resources?
- Is the system's responsiveness continuously predictable?
- Can the system's processing capacity be expanded with more hardware?
- Has the system processing bottlenecks limiting the capacity?
- Is the system configured to fully utilize the hardware platform?
- Can the system manage various production-critical situations such as DoS attacks?
- How long does it take to recover from a partial or a full restart?

11.4 Performance objectives and performance requirements

"Performance objective" is a term for desired performance goals that a system should meet. Performance objectives should at least cover stated performance requirements, if specified. Performance objectives can be defined as absolute performance figures or as performance figures relative to stated performance requirements or the measured performance of other systems. Relative performance figures are usually percentage values, such as 30% higher capacity than brand "XYZ" or 20% reduction of response time for service "ABC".

"Performance requirements" is a term for performance figures a system is expected to meet to be approved. Performance objectives and performance requirements can be stated for a range of objects from a whole system, or a subsystem down to individual services.

11.5 Performance measurement conditions

Performance measurement conditions specify the circumstances under which performance data can be recorded during a performance test. Performance measurement conditions can be external, internal or both.

11.5.1 External measurement conditions

External measurement conditions describe what a tested system (SUT) should be exposed to during a performance test. External measurement conditions include types of service requests, volumes of service requests (traffic rates), duration of service request volumes and volumes of simulated entities or users requesting services.

11.5.2 Internal measurement conditions

Internal measurement conditions describe expected situations inside a tested system during a performance test, such as resource usage levels of CPU, memory, etc.

11.5.3 Example

If, for example, an explorative performance test case is to find the maximum system throughput of a specific service at 80% CPU load there are two test conditions, one internal and one external:

1) external test condition: system load from service requests of the specified type
2) internal test condition: a measured CPU load of 80%

11.6 Performance targets

Performance targets describe expected performance goals in a validating performance test.

11.7 Performance measurements standards

Performance measurement standards are generally accepted specifications for how to measure and evaluate some kind of performance on a standardized system or an architectural standard for a system.

11.8 Some performance test characteristics

11.8.1 Test coverage

Performance tests cover the application services with potential performance implications, i.e., the most frequent, the most critical services or the most demanding services of an application.

Performance tests normally measure the performance of system services that have passed functional tests (positive test cases). Measuring performance of non-working system requests (negative test cases) is regarded as beside the point. In some explorative cases, however, performance tests can be positive test cases driven to the point of non-working service requests by load level increase beyond what the tested system can handle.

11.8.2 Test purposes

A performance test is either explorative with the purpose to find performance limits of a tested system, or confirmative with the purpose to verify that one or more stated performance requirements are met.

11.8.3 Test cases

Performance tests of a system require a smaller number of test cases than functional tests.

11.8.4 Test concurrency

Performance tests block the test environment, i.e., no other activities can be performed on a tested system during an ongoing performance test.

11.8.5 Test resources

Performance tests are usually demanding on hardware resources and consequently expensive to set up. The main reason for this is that performance tests are exclusive, i.e., no other activities are allowed on a system under test for performance tests. In particular, performance tests of availability and stability are demanding, since the test time might last up to several weeks during which no other activities can be performed on the system under test.

11.8.6 Test execution and test case

A single performance test collects performance data that cover many performance test cases.

11.8.7 Test execution time

The execution time of a single performance test varies from some minutes up to several weeks depending on type.

11.8.8 Recorded test data

A performance test records massive amounts of measurement data that need to be managed wisely.

11.8.9 Test data evaluation and test results

A performance test result is rarely binary (passed/not passed). On the contrary, performance measurement results are complicated to understand and require experienced and careful handling to give an understandable and useful verdict.

12 Performance test environment

12.1 Test environment concepts

The performance test environment contains hardware and software components required to run performance tests.

When operational for performance tests, the performance test environment is called a test bed or a Test site (Figure 14).

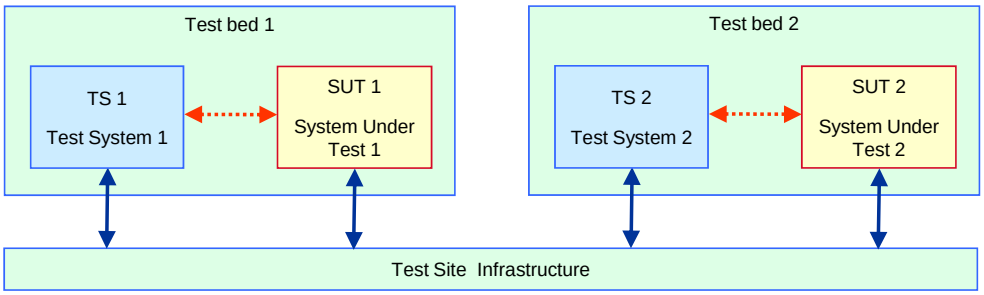

Figure 14 – A general view of a test site with test beds, test systems and systems under test

12.1.1 Test bed concepts

A test bed contains hardware and software components that:

1) constitute the test system (TS);

2) constitute the system under test (SUT); and

3) connects the test system to the system under test.

The system under test and the test system are usually installed on physically separated equipment.

A test bed contains the physical interface between the performance test tools and the system under test, i.e., network components such as switches, routers and other components corresponding to layer 1 of the OSI model. The test bed supports the logical interface between the performance test tools (TS) and the system under test (SUT).

The logical interface can be an application programming interface (API) or a communication protocol interface corresponding to layer 7 of the OSI model. Both interfaces are in most cases IP-based communication services, but other interfaces such as SS7 can be requested.

12.1.2 Test site concepts

A performance test requires exclusive access to the system under test. Consequently, any concurrent performance test is done on a separate SUT. Large development projects usually use several performance test beds to enable all required performance tests to be done within the given time limits of the performance test project.

A test site is a test location with equipment that:

1) enables two or more test beds to be configured and work concurrently; and

2) allows equipment to be reassigned between the supported test beds, i.e., each test bed can be equipped differently from performance test to performance test.

12.2 System under test concepts

12.2.1 System under test components

A system under test or SUT is the set of hardware and software components constituting the tested system in a performance measurement (Figure 15). A system under test is composed of two parts:

1) tested components (TC)

2) supporting components (SC)

The reason for the decomposition is that a system under test will report different performance figures depending on the set of supporting components it is tested on. This applies to all systems not dedicated for a specific platform.

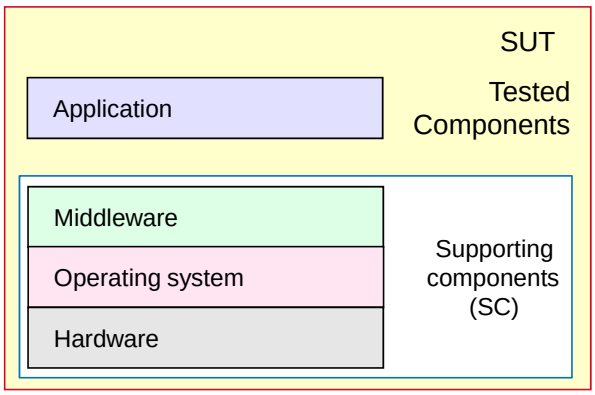

Figure 15 – Components of an SUT

Tested components

The tested components are, in the context of a distributed system, the requested services on a system under test.

Supporting components

The supporting components are all hardware and software components required to enable performance tests of the tested components. Typical supporting components are:

1) middleware software, such as database software or application platform software, etc.
2) operating system software
3) hardware, such as servers, disk systems, load balancing equipment, etc.

The supporting components are regarded as tested components when the system under test is able to run on one specific set of supporting components only. In those cases, there is only one set of measured performance results.

The supporting components are regarded as a test condition, when the system under test is able to run on multiple sets of supporting components. In such cases, measured performance results refer to the used set of supporting components.

12.2.2 Borders of a system under test

A system under test has two types of borders interfacing the test system components (Figure 16):

1) front-end borders
2) back-end borders

Front-end borders

The Front-end borders are the intersections between the system under test and the test system's service requesting tools. The Front-end borders contain the interfaces for incoming service requests.

Back-end borders

A system under test may provide services where requests are passed on to an external unit sometimes called a service responding device. In order to test such services the test system contains service responding tools simulating such units. The back-end borders contain the interfaces between the test system and the system under test for outgoing service requests.

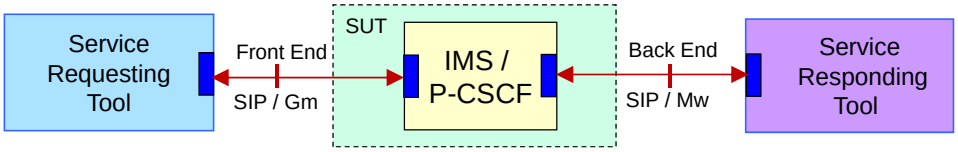

Figure 16 – Example of an SUT with front-end border SIP/Gm (left side) and back-end border SIP/Mw (right side)

12.2.3 System under test replacements

In a distributed system services are usually available in a client-server relation. The party requesting a service is called a client and the party providing the requested service is called a server. A server can in turn act as a client requesting services. These services can be internal services or external services shared with other application systems.

Such internal or external services can in some cases be replaced by service simulation tools providing identical services to the tested system (see also clause 12.3.3).

12.3 Test system concepts

12.3.1 Performance test tools

A complete performance test tool is a set of hardware and software components that can handle the following tasks:

1) expose the SUT to a set of (load) conditions, under which performance measurement data are captured
2) transform captured performance measurement data into desired performance metrics about the SUT
3) monitor and display captured and processed performance measurement data online during an ongoing test
4) evaluate performance test results
5) present evaluated performance test results

The first task is handled by service handling tools, service simulation tools and performance data recording tools.

The second task is handled by performance data processing tools.

The third task is handled by performance test monitoring tools.

The fourth task is handled by performance evaluation tools.

The fifth task is handled by performance presentation tools.

12.3.2 Service handling tools

Service handling tools are the interfaces to the SUT for system services specified in the performance test cases.

Service handling tools interact with the SUT in two ways (Figure 17):

1) as service requesting tools
2) as service responding tools

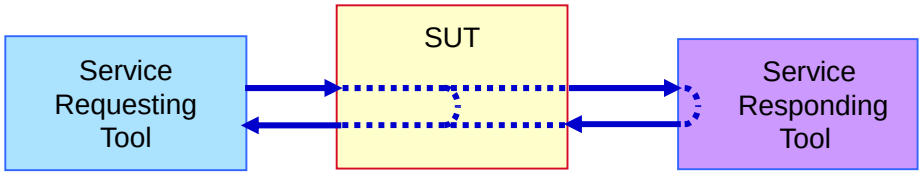

Figure 17 – Example of a test bed with service requesting and service responding tools

Service requesting tools

Service requesting tools, usually called load generators, send service requests to the SUT according to the test specifications. When the SUT has an API interface, the service requesting tool simulates an application requesting services over the application programming interface. When the SUT has a protocol interface, the service requesting tool simulates a device or a system requesting services over the protocol. Regardless of the SUT interface, the service requests are in performance tests referred to as client requests for services from the SUT.

Service responding tools

There are system services that connect a requesting client to one or more counterparts (usually called terminating devices) outside the tested system. Terminating devices are usually simulated by test tools receiving and responding to requests in the test bed. Such services normally require a peer-to-peer protocol, such as SIP or Diameter, where communicating devices are able to concurrently act as clients initiating server requests and servers responding to service requests.

Performance test tools interfacing peer-to-peer protocols are able to send service requests to the SUT and receive requests from the SUT concurrently.

12.3.3 Service simulation tools

The SUT contains all components required to resolve requested services, whenever a service is tested. SUT components with well defined services and interfaces can be replaced by service simulation tools.

There are several purposes with service simulation tools such as:

1) reduction of costs for a test bed. Service simulation tools are usually much cheaper than replaced units

2) shorten the time to build a test bed. Service simulation tools are usually less complex and easier to install

3) reducing the complexity to build a test bed. Service simulation tools are usually less complex to use.

Example: Registration of an IMS user is handled by two components in the IMS architecture, the S-CSCF and the HSS. When testing the capacity of an S-CSCF to handle registration requests, a real HSS can be replaced by a service simulation tool acting as an HSS when accessed by the S-CSCF (Figure 18).

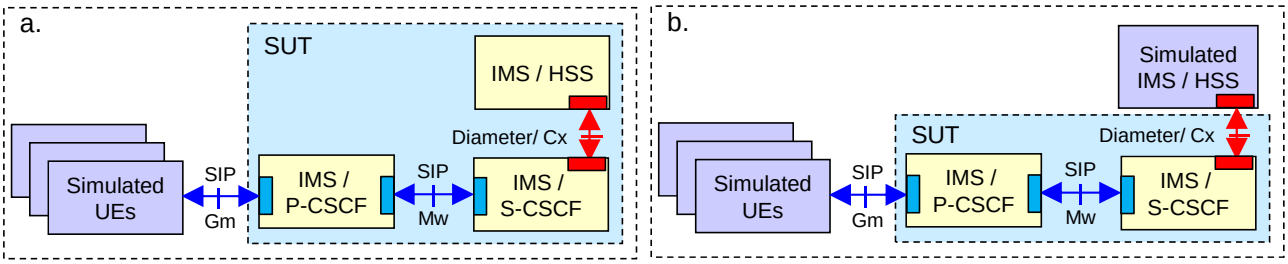

Figure 18 – Example of an SUT, with (a) a real HSS and (b) a simulated HSS

Service simulation tools also enable new possibilities to measure performance. By replacing an HSS by a test tool simulating the HSS services we can measure the time spent on processing a registration request in an S-CSCF, since the time spent processing a registration request in an HSS is controlled by the test tool.

12.3.4 Performance data recording tools

A main function of performance test tools is to capture and save performance data. Performance data can be captured externally and internally with respect to the SUT (see Figure 19).

External performance recording tools

External performance data are measurements of how the SUT responds to requests from the test system's service requesting tools. External performance data are captured by the test system's service requesting tools and the test system's service responding tools (if any), and recorded by the test system's data recorder tools.

Internal performance recording tools (probes)

Internal performance data are measurements of how the SUT handles service requests from the service requesting tools internally. Internal performance data are captured by probes running inside the SUT and recorded by the test system's data recorder tools. The probes are managed by the test system.

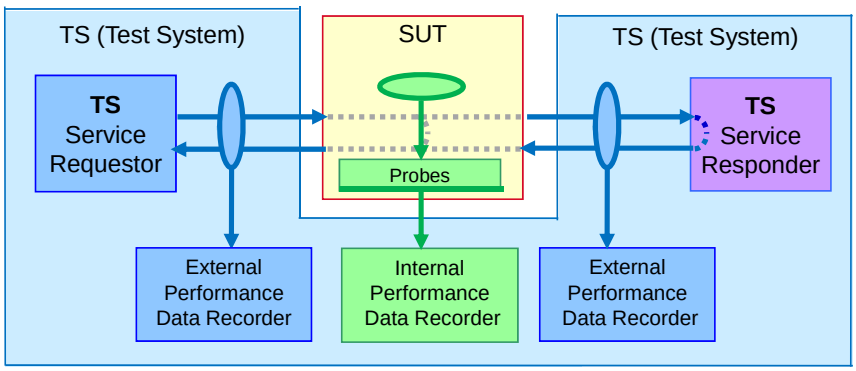

Figure 19 – External and internal performance recording tools

12.3.5 Performance test monitoring tools

Performance test monitoring tools (Figure 20) enable captured measurement data to be processed and viewed in real time or semi-real time during execution of a performance test.

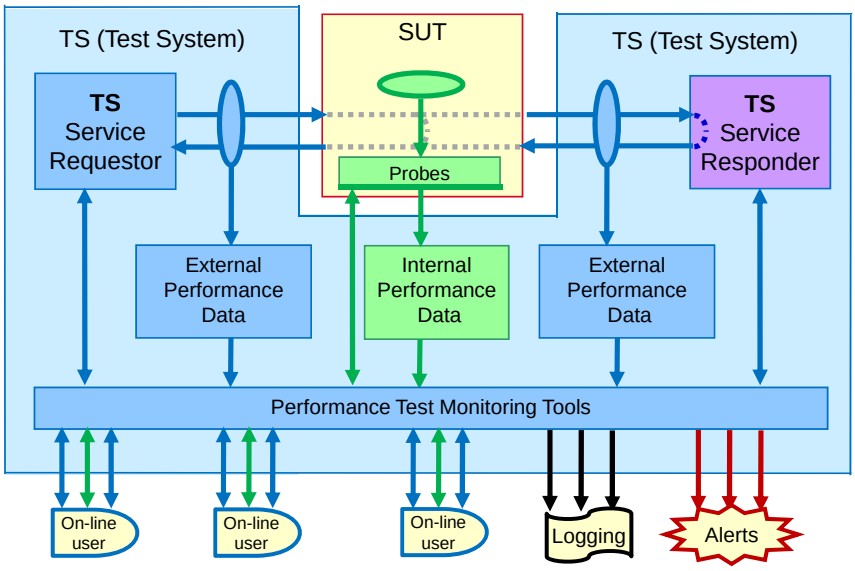

Figure 20 – Performance test monitoring tools

The purpose of a performance test monitoring tool is to make the information about the progress of an ongoing test instant, and consequently to improve the control of tests with long execution time.

For example, if a performance test of stability and availability characteristics is planned to run one week but for some reason fails after three hours, it is a waste of time to let the test continue the remaining 165 hours.

Performance test monitoring tools can usually also be set to trigger on specified conditions. Alerts about such situations are sent to other monitoring systems for further processing. Monitoring tools can also in many cases send alerts as SMS messages to remote units.

12.3.6 Performance data processing tools

Performance data processing tools transform measurement data into metric values describing requested performance characteristics of a system.

12.3.7 Performance evaluation tools

Performance evaluation tools rate processed metric values according to a set of rules. The purpose is to automatically produce a verdict about measured performance of the SUT. For reliability metrics, performance evaluation tools will process captured performance data to identify trends or irregular behaviour that could endanger the service production. For efficiency metrics, both trend spotting and rule-based checks applies.

12.3.8 Performance presentation tools

Performance presentation tools transform measured performance into graphs and other presentation formats. The purpose is to improve and enhance interpretation of measured performance.

13 Performance test specifications

13.1 Elements of performance test specifications

Specifications on a performance test include the following elements (Figure 21):

1) test objectives
2) test conditions
3) test configurations

4) test data specifications

5) test evaluation specifications

Performance test specifications are translated into performance test configurations, i.e., configurations of test system, system under test and test bed infrastructure to enable collection of performance data.

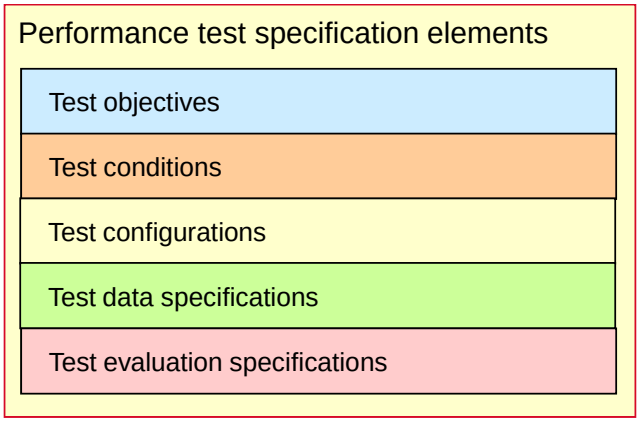

Figure 21 – Elements of performance test specifications

13.2 Test objectives

The test objectives of a performance test state the purposes of the test, i.e., what will be achieved by running the test.

13.3 Test conditions

The test conditions of a performance test include:

1) test specification prerequisites

2) test execution preconditions

3) operational measurement conditions

4) test execution post-conditions

13.3.1 Test specification prerequisites

Test specification prerequisites refer to output from other performance tests that is required as input to a performance test specification. A consequence is that such performance tests have to be executed before specifying the intended performance test.

Performance tests of stability and availability characteristics is an example of test cases that require the output from other performance tests as input to the test specifications. A performance test of stability and availability characteristics usually runs for days or even weeks at 80% of a system's measured maximum throughput level. A prerequisite to specify such a test is information about what is 80% of a system's measured throughput level. This information is obtained in a performance test of the system's sustained throughput capacity.

13.3.2 Test execution preconditions

Certain test-bed and SUT conditions are expected to have been met before starting the performance test. These conditions are referred to as test preconditions. The preconditions also apply after the initial test steps (the warm-up phase when simulated users get activated and the system is prepared to receive service requests) have completed. Preconditions are usually stated for the performance test bed as well as the tested application.

Test bed preconditions

Examples of test bed preconditions are:

1) exclusive access to the performance test bed
2) enough physical resources for execution of a performance test case, such as disk space
3) correct configuration of the test bed components, such as test tool equipment connected to all open SUT interfaces

Application preconditions

Examples of application preconditions are:

1) ensure that all simulated entities are in the correct state to run the test, such as all simulated entities, are successfully registered and accepted by the system (not always required)
2) ensure that common application resources, such as databases, contain expected data (not always required)
3) ensure that required load is applied on the SUT
4) ensure that the SUT can process requested application services

13.3.3 Operational measurement conditions

Operational measurement conditions state conditions that apply when measurement data are captured during an ongoing performance test. Operational measurement conditions may also state under what circumstances a performance test can be stopped. There are two types of measurement conditions:

1) requested measurement conditions
2) actual measurement conditions

Requested measurement conditions

Requested measurement conditions are stated requirements on the SUT and test bed (test tool) conditions for capturing performance data. Requested measurement conditions are a part of the performance test specification. Requested measurement conditions can be external or internal.

External measurement conditions

External measurement conditions describe what a tested system (SUT) is to be exposed to during a performance test. External conditions include types of service requests, volumes of service requests (traffic rates), duration of service request volumes and volumes of simulated entities or users requesting services.

Internal measurement conditions

Internal measurement conditions describe expected situations inside a tested system during a performance test, such as resource usage levels of CPU, memory, etc.

Actual measurement conditions

Actual measurement conditions are recorded conditions that applied when performance data were captured. The purpose of actual measurement conditions is to validate recorded performance data.

Actual measurement conditions include metrics such as load deviations – the differences between intended load and actual load during a performance test.

13.3.4 Test execution post-conditions

A set of post-conditions expected to be met after execution of the performance test has completed and before the performance test is regarded as completed. Examples of post-condition activities are:

1) all system resources reserved during test execution are released, such as all sessions, subscriptions or other resources related to pending services are returned
2) central resources on the test bed are reset, such as used databases

The purpose of post-conditions is to bring the test environment to a well defined initial state for the next test.

13.4 Test configurations

Performance test specifications are a set of specifications that apply to the test system and the system under test and enable execution of the performance test. Performance test configurations include:

- workload specifications;
- test bed specifications;
- data collection specifications.

A performance test configuration covers, in most cases, more than one performance test case.

13.4.1 Workload specifications

Workload specifications describe what a system under test is expected to handle or process during a performance test. Workload is described in more detail in clause 14.

13.4.2 Test bed specifications

The test bed specifications describe the test bed configuration of equipment for different services in a performance test.

The test bed specifications for interfaces between the test tools and the SUT describe:

- IP addresses and listening ports of the SUT for different services;
- used network layer protocols such as IPv4 and/or IPv6;
- used transport layer protocols such as TCP, UDP, SCTP, etc.;
- used application layer protocols such as HTTP, SOAP, SIP, Radius, Diameter, DHCP, etc.;
- security settings such as IPsec or HTTPS;
- time-out settings for service requests.

Other test bed specifications describe:

- the hardware configurations of servers in the SUT;
- the number of servers and load balancing equipment in the SUT;
- the test bed equipment interconnecting the SUT and test tools;
- requested versions of all software involved in a performance test.

The purpose of having test bed specifications is to recommend a test environment that will allow a performance test to be repeated identically at any point in time.

13.4.3 Data collection specifications

Data collection specifications contain specifications of performance data that should be captured.

Data collection specifications include:

1) specifications of internal performance data collection

2) specifications of external performance data collection

3) specifications of performance recording details

Specifications of internal performance data collection

Internal performance data collection specifications include:

- specifications of selected performance variables, such as CPU usage, memory usage or queues;
- recording location of selected performance variables, such as per server or for a specified process group;
- the frequency of recording internal performance data inside the SUT.

Specifications of external performance data collection

External performance data collection specifications include:

- specifications of selected performance variables for requested services and related responses;
- specifications of selected performance variables for actual measurement conditions.

Specifications of performance recording details

The performance recording details include configuration parameters such as:

- the resolution of recorded performance data, such as seconds, milliseconds or microseconds for response time;
- the frequency of saving captured performance data on disk, i.e., sample time for recording performance data.

13.5 Test data specifications

Test data specifications are a set of specifications to enable authentication of simulated users, authorization of service requests, customization of service requests and evaluation of test results.

Test data specifications contain three parts:

- test data for service requests;
- test data for SUT operability;
- test data for performance evaluation.

13.5.1 Test data for service requests

Test data for service requests is a set of parameter values for every simulated user that is used to make every service request from every simulated user unique.

13.5.2 Test data for SUT operability

Test data for SUT operability is a set of unique parameter values for every simulated user that corresponds to stored information about the users in the SUT's databases, such as identities, phone numbers, account numbers, etc. The test data for SUT operability is required to authenticate and authorize service requests from simulated users during performance tests, i.e., a prerequisite for SUT operability.

13.5.3 Test data for performance evaluation

Test data for evaluation of performance measurement is a set of evaluation rules and expected measurement values. Test data for performance evaluation of regression tests contain the performance measurement results from some previous executions of the performance test together with the evaluation criteria.

13.6 Test evaluation specifications

Test evaluation specifications are a set of rules and settings for evaluation and rating of performance test results.

Test evaluation specifications will, for instance, specify ranges for a performance measurement result when an appropriate verdict such as excellent, good, acceptable, poor, bad, etc., should be applied.

14 Workload concepts

Workload is a term for what a system under test is expected to handle or process totally during a performance test. A workload has three components:

- workload content;
- workload volume;
- workload time distribution.

A workload can also define what load an SUT is exposed to at a given point of time during the test.

14.1 Workload set or traffic set

A performance test may contain several workload specifications, where each workload specifies a group of simulated users exposing the SUT to a specific set of service requests with a specified intensity. A set of workload configurations is also called a traffic set.

14.2 Workload content

The workload content describes the services that will be requested from the SUT during a performance test. The workload content contains two parts:

1) a user session scenario or a requested service profile
2) a set of service scenarios

14.2.1 User session scenarios

A user session scenario describes the workload content as seen from the requesting side (the TS). A user session scenario contains a list of services requested during a user session and in which order the services are requested, i.e., the flow of service requests in a user session. The flow also contains alternative paths depending on the outcome of a service request. A user session specification usually has some exits for emergency situations too.

14.2.2 Requested service profile

A requested service profile describes the workload content as seen from the receiving side (the SUT). The requested service profile contains a list of requested services. Each service request has a specification of how frequently it should be generated, expressed as a percentage of all requests. The sum of percentage values for all service requests is 100.

14.2.3 Service scenarios

Service scenarios are specifications of individual service requests and include topics such as:

- how requests are sent to the SUT;
- how response messages from the SUT are validated;
- how errors reported from the test bed, such as timeout or disconnects, are handled;
- how the outcome of a service request is reported back to the user session;
- building a service request with requested content and formatted with user specific information.

14.3 Workload volume

The workload volume is a description of the amount of services (described in the workload content) that is requested during a performance test. Workload volume and workload time distribution are also referred to as load characteristics for a performance test.

The workload volume is specified differently depending on the type of load generation (see clause 14.4, load concepts).

For 'user session driven load', workload volume is defined as a set of user session load steps, where each load step specifies a number of concurrently simulated users and duration. For 'traffic rate driven load', workload volume is defined as a set of traffic rate load steps, where each load step specifies a traffic rate and duration.

14.4 Load concepts

Depending on the design of a test tool there are two concepts for generating load:

1) user session driven load;

2) traffic rate driven load.

14.4.1 User session driven load

User session driven load is based on traffic generated by a number of simulated users, where the rate of service requests from each user is controlled by think-time delays between consecutive service requests.

The total load on the system is in this case determined by the number of concurrently active user sessions. In order to increase the system load, more simulated user sessions have to start.

The number of concurrently active user sessions during a performance test is controlled in a load script. In some cases load can be manually controlled during a test.

14.4.2 Traffic rate driven load

Traffic rate driven load is based on traffic controlled by a central load control function in the test tool keeping track of when a user session is instructed to send a request. The traffic rate specification is independent of the number of simulated entities.

The load control function executes a load script telling what traffic rate should be applied in every moment. Each specified traffic rate in the load script has a duration time. The total performance test duration is set by the sum of all specified duration times. Transition time between two traffic rates is instantaneous.

14.5 Workload time distribution

The workload time distribution is a description of how the amount of requested services (the workload volume) is distributed over time during a performance test.

14.5.1 Load profiles

A load profile is the set of load conditions defined in a load script.

14.5.2 Load patterns

The load on the SUT can follow several types of patterns such as:

- constant load;
- peak load;
- saw-tooth load;
- statistical load;

- stepwise increased load.

Constant load

This is a load pattern where the SUT is exposed to a fixed rate of service requests per time unit. Constant load is commonly used in performance tests of stability and availability characteristics.

Peak load

This is a load pattern where the SUT is exposed to a repeated sequence of short periods of very high load (peaks) followed by longer periods of low load. Peak load is commonly used in performance tests of robustness characteristics.

Saw-tooth load

This is a load pattern where the SUT is repeatedly exposed to a sequence of increasing and decreasing load. Saw-tooth load is commonly used in performance tests of robustness characteristics.

Statistical load

This is a load pattern where the SUT is exposed to load according to a statistical model for arrival of service requests, such as a Poisson, F distribution or Erlang. Statistical load is commonly used in simulations of service production in system delivery tests.

Stepwise increased load

This is a load pattern where the SUT is exposed to a sequence of fixed load increases. Stepwise increased load is commonly used in performance tests of capacity characteristics. Stepwise increased load is sometimes called staged load.

Bibliography

[ISO/IEC 9646-1] ISO/IEC 9646-1:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2013