J. Basic. Appl. Sci. Res., 1(7)559-563, 2011

ISSN 2090-424X
Journal of Basic and Applied Scientific Research
www.textroad.com

Measurement and Specification Framework for System Performance Requirements

Farid AlZboun

Ajloun National University, Collage of Administration and information Technology

ABSTRACT

In the project management and cost estimation projects, The functional and non-functional requirements are defined for benchmarking and estimation purposes, Unfortunately, The non-functional requirements are defined lately in the testing and evaluation phases and not early in the feasibility study. This paper presents an early method of measuring the performance of a computer system. The performance is a non functional requirement, and so there is no yet a standard on how we can measure the performance of a system. We will identify the components of performance and the way to measure each of these components in order to get a measure of performance as a result of this measurement method.

KEY WORDS: Performance Requirements, Measurement Framework, Measurement Method, Non functional requirements, Derived and Base measurements.

INTRODUCTION

Performance is defined the speed at which a software operates in a computer, either theoretically for example, using a formula for calculating MTOPS (millions of theoretical instructions per second) or by counting operations or instructions performed, for example, MIPS (millions of instructions per second). It is a pervasive quality of software systems; everything affects it, from the software itself to all underlying layers, such as operating system, middleware, hardware, communication networks, [1].

Performance measurement: We measure to learn what is happening, in order to identify problems, make corrections, and demonstrate successes. In general Performance measurement is about measuring results. It is the process of seeking real-time answers to the question: "Are we getting what we expect at an acceptable cost?" [2].

Software Performance measurement can be used in many fields and for many reasons:

- It can provide valuable information to support effective system design.
- Provides information needed to intelligently make tradeoff decisions.[3]
- Can be used to compare alternative hardware configurations and, therefore, can aid in determining the final hardware specifications for a future system.
- Can aid in specifying the design of data structures and application software.
- Can be used to resolve operational considerations. Measuring the performance effects of certain capabilities
- Can provide data on which to base decisions such as whether or not system capabilities should be provided or not.

This paper is organized as follows. Section 2 presents the steps for designing a measurement method. Section 3 presents an overview of the performance components. Section 4 presents the measurement method for the 'performance' as non-functional requirements. Section 5 presents an evaluation of conclusion is presented in section 6.

1. Measurement Method

Measurement objectives: We are creating this method of measurement in order to:

- Quantify the performance of software so we can determine if specific software is performing as it should or as much as it is supposed to.

***Corresponding Author:** Dr. Farid AlZboun, Ajloun National University, Collage of Administration and information Technology
Email: alzboun@yahoo.com

- To forecast the impact of changes newly done in a system, such as a reconfiguration of the hardware or an improvement in the frequently executed software modules.
- Provide a general sense of what results are expected from the measured software. And to help in making other decisions depending on the result of this measurement. Constraints: There are some constraints that we should deal with in measuring the performance of a software system such as:
- The capabilities of the computer on which we do the test or the calculations (size of memory, capacity of CPU) and the operating system is a central factor to the performance of software.
- The speed of assembly and compilation plus the execution of the resultant output code are also of great importance.[4]
- Packages, and utilities are also a part of the computer system, and their performance will affect the total systems performance.

Characterization of the concept to measure: What is the concept to be measured? We are measuring the software performance which can be decomposed into many components, each of which can also be measured.

2. Performance Components

There are many definitions available for identifying the term "software performance" but the problem is that each is looking to the performance from a specific viewpoint (Ex: performance of a software in the networking field, production field and each of these fields introduces a new component or attribute to performance like: availability, effectiveness, fault tolerance which are considered as a part of performance of any network software.

In this document we will use the categorization adopted in the ISO/IEC 14756: 1998(E) which characterized the software performance by two main attributes: Execution time and throughput [5].

Execution time referred to as "response time": The time it takes an instruction or a task to be executed from its submission to completion. It makes up the last half of the instruction cycle. [6]

The execution time of a task depends on many factors which can be classified into two groups: intrinsic and extrinsic. [7]

- The intrinsic factors control the flow within a task. Example intrinsic factors include precedence relations and condition parameters that determine loops and branches within a task. [7]
- The extrinsic factors, on the other hand, control interactions among tasks. Examples of extrinsic factors include resource contention, communication, and synchronization delays.[7]

Throughput: System Throughput is defined as the quantity of jobs completed by the overall system per unit of time. In other words, it's the amount of work that a computer can do in a given time period or the amount of output that be produced by a system or component in a given period of time. [8]

Historically, throughput has been a measure of the comparative effectiveness of large commercial computers that run many programs concurrently. An early throughput measure was the number of batch jobs completed in a day. [9]

Entities that should be measured to determine performance of software are instructions or tasks executed by that software.

Tasks categories: Tasks can be categorized depending on many aspects such as: priority, complexity and type.

- Priority: Some tasks can be given a higher level of priority than others. So the will be executed first.[10]
- Complexity: Some tasks can be more complex than others. So execution time will differ depending on this complexity.
- Type: Tasks can be categorized into two types. Tasks that has relations ships with other tasks or tasks that are independent of other tasks.

In this procedure of measurement we do not take into consideration tasks categories such as complexity and the other characteristics of tasks. Because each of these characteristics will affect the results obtained during the calculation of performance at throughput and performance at execution time. That means calculation is made after categorizing tasks depending on complexity and priority.

What is an instruction? It is a basic command. The term instruction is often used to describe the most rudimentary programming commands. For example, a computer's instruction set is the list of all the basic commands in the computer's machine language**.**

- To measure an instruction we need to know:
- The clock cycles of the computer on which this instruction will be executed.
- Numerical Assignment Rules

3. Design of Measurement Method

Four steps are recommended to carry out the design of a measurement method:

Step1: Determination of the Measurement Objectives: Before designing a software measure, it is important to answer the

following questions, it will give a strong influence on the design of the measurement method.
Step 2: Characterization of the concept to be measured**:** In order to enable the measurable concept to be defined into a measurable construct and from it the measurement method to be built, the concept to be measured must be clearly defined,
Step 3: Design or Selection of the Meta Data: The set of characteristics selected to represent software or a software piece, together with the set of their relationships, constitute the meta-model of the software to which the proposed measurement method will be applied. This meta-model should be described generically: it should not be specific to particular software.
Step 4: Numerical Assignment Rules: consists in defining the measurable concept and an empirical relational set, to complete the design of a measurement method, a numerical relational set and a homomorphism between these two relational sets must be defined,

3.1 Measurement Procedure

As we discussed earlier software performance has two attributes throughput and execution time. And measuring performance at each of these attributes will enable us to calculate overall performance of the system.

To measure performance of the whole system we need to calculate the mean value of performance at the execution time of the whole system and the mean value of performance at throughput of that system. Next the sum of the result is divided by 2. The final result obtained will be a measurement of the system performance. It will be measured in MIPS (millions of instructions per second). We will discuss later the procedure or different steps that should be done in order to calculate those means.

But in this case a new question will arise for example if we get the value of 40 MIPS as a final result. What does 40 MIPS means? Does 100 MIPS mean the best score we can get as a performance measurement? Why it is not 200 or 100 MIPS?

To avoid such problems and to make the results meaningful and clearer we decided to give the result as percentage where 100% is the ultimate performance of a system.

How to measure each of the components: Measuring execution time: Execution time of an instruction or a specific task as defined earlier: the time needed by software to take request and get results.

Execution time =
Request Timing – Result Timing (1)

Performance at the execution time: To calculate performance at execution time of certain task or instruction, we calculate the ratio of time stated in the specification document and the real time it takes software to execute a task:

Execute Task =
Specific Doc. Time / Real Time. (2)

But how much the result of that ratio is significant or to what will we refer to compare our result? There is currently, no standard to specify that this obtained result will be considered as good or bad. To avoid this question we decided to give the result of the execution time as a percentage which is more meaningful and more comprehensive to readers. So performance at the execution time will be measured as follows:

Performance at Execution Time =

(Specific Doc. Time / Real Time)* 100 (3)

Example: Time stated in specification document for executing a series of instructions or a specific task "M" = 0.4s and the Real time taken by software to execute task "M" = 0.9s. So performance at the Execution time of task "M" is: (0.4/0.9)*100= 44.44%.

Measuring Throughput: As defined earlier it's the amount of work a computer system can do or accomplish in a specific period of time [11]. For example: In data transmission, throughput is the amount of data moved successfully from one place to another in a given time period [9]. The measurement of throughput can be done by measuring data units per unit of time, such as bits, bytes, blocks, cells, frames, or packets per second [12].

Throughput =

data units / time units (4)

The Prformance at throughput: Performance at the Throughput is the output relative to input; the amount passing through a system from input to output (especially of a computer program over a period of time) [13].

For example: in Ethernet, the interframe gap is 12 bytes, and the maximum frame size 1518 bytes (maximum 1500 byte payload + 8 byte preamble + 14 byte header + 4 Byte trailer). An additional minimum interframe gap corresponding

to 12 byte is inserted after each frame. This corresponds to a maximum channel utilization of 1518/ (1518+12) *100% = 99.2%, or a maximum throughput of 99.2 Mbit / s in a 100 Mbit / s Ethernet connection [13].

Performance at throughput =
(Max data units / max data unit + delays)*100% (5)

3.2 Derived Measure

How to measure the overall performance of a system? We have several steps to get the final result of performance measure of a system:

- We have to identify instructions to be executed by software
- Categorize the instructions: depending on complexity and priority (not applied in this case).
- Calculate performance at execution time for each instruction or task
- calculate performance at throughput for each instruction or task
- for each category we calculate mean value of performance at execution time:

∑ Duration of tasks / number of tasks (6)

- for each category we calculate mean value of performance at throughput time

∑ Units of data / units of time (7)

- We calculate mean off all performance at execution time:

∑ means of each category/ Nb of categories (8)

- We calculate mean off all performance at throughput time:

∑ means of each category/ Nb of categories (9)

- add both values in step 7 and 8 and then divide it by 2
- The result will be a measurement of the system performance. To clarify this finale result we identify a scale 0 to 100 a degree of 50 means 50% of performance.
- To make the result more meaningful to readers and will satisfy managers who asks their employees all the times about numbers as a result of measuring a software.

3.3 Measurement Example:

In this example we will measure the performance requirements of software by using the following assumed specification, assume we identified 8 instructions to be executed by that software. We categorized them into three different categories see table 1. We have to calculate the performance at execution time and at throughput for each of these instructions as discussed earlier.
Performance at execution time of each category:

Table 1: Performance at execution time of each category

Performance Category1	Performance Category2	Performance Category3
90	75	55
70	69	94
88		91
Mean value: 82.7	Mean value: 72	Mean value: 80
Mean value of performance at execution time all categories: 78.3		

Performance at throughput of each category:

Table 2: Performance at throughput of each category

Performance category 1	Performance category 2	Performance category 3
80	60	90
75	77	86
89		95
Mean value: 81.3	**Mean value: 68.5**	**Mean value: 90.3**
Mean value of performance at throughput of all categories:80		

The final step is to add these 2 values of mean performance of all categories: (78.3 + 80) / 2 = 79.15.

Conclusion and Evaluation of the procedure of measurement

The proposed measurement of the software performance in this paper is identified separately the all functionality allocated to software performance based on architectural systems for software, whether it has yet to be built or it has already been delivered.

As we mentioned earlier that during the calculation of performance at throughput and performance at execution time we do not take into consideration that tasks might have different priorities or different complexities and sizes. So the calculation is made after categorizing tasks depending on complexity and priority.

While developing this method of measurement we tried as possible as we can to be based on Existing standards, ISO or IEEE, UML. We try to enlarge consensus on this design of this method of measurement, also we try to specify the measurement units.

This paper introduced a new design measure for the performance as a non-functional requirement independently of the software type or languages or hardware architecture.

Finally, as measuring performance of software is not standardized yet we try to give a more consensus on the way or the method of measuring software performance, because that will help in making better decisions concerning the addition of new hardware components or even software components to a system and it will give more information about the behavior of the new system.

REFERENCES

1. The Future of Software Performance Engineering Murray Woodside, Greg Franks, Dorina C. Petriu
2. http://www.tbs-sct.gc.ca/eval/pubs/pubs-since-1996/icguide-icguide-e.asp#c
3. Joann Lockett January 1978 Proceedings of the software quality assurance workshop on Functional and performance issues
4. H. Lucas Jr, “Performance evaluation and monitoring”
5. ISO/IEC 14756:1988(E), Information technology-Measurement and rating of performance of computer-based software systems, International Organization for Standardization Geneva, 1988.
6. Christof Ebert, Reiner Dumke, Manfred Bundschuh, Andreas Schmietendorf,’ Best Practices in software measurement’, Springer, 2005.
7. IEEE TRANSACTIONS ON COMPUTERS, VOL. 45, NO. 5, MAY 1996 execution Time analysis communicating tasks in distributed systems Jong Kim, Member, /€E€, and Kang G. Shin, Fellow, /€E€
8. http://www.webopedia.com/TERM/t/throughput.htm
9. http://searchciomidmarket.techtarget.com/sDefinition/0,,sid183_gci213140,00.html#
10. US Patent 6795797 - Method and apparatus for measuring CPU task occupancy rate in a real-time system
11. http://en.wikipedia.org/wiki/Throughput
12. http://www.yourdictionary.com/throughput
13. http://www.answers.com/topic/throughput cat=technology&nr=1