e-Informatica Software Engineering Journal, Volume 7, Issue 1, 2013, pages: 53–64, DOI 10.5277/e-Inf130106

Time Domain Measurement Representation in Computer System Diagnostics and Performance Analysis

Stanisław Wideł*, Jarosław Flak*, Piotr Gaj*

*Faculty of Automatic Control, Electronic and Computer Science, Silesian University of Technology

stanislaw.widel@polsl.pl, jaroslaw.flak@polsl.pl, piotr.gaj@polsl.pl

Abstract

Time analysis is a common approach for testing and detecting methods for the performance analysis of computer systems. In the article it is shown, that measuring and identifying performances based on a benchmark is not sufficient for the proper analysis of the computer systems behavior. The response time of the process is often composed of the execution of many subprocesses or many paths of execution. Under this assumption, it is presented, that both convolution and deconvolution methods can be helpful in obtaining time distributions and modeling of complex processes. In such a modeling the analysis of measurement errors is very important and was taken into consideration. The example of using the methods in buffering process is also discussed.

1. Introduction

Performance analysis and diagnostics of system software is a difficult problem even for simple computer systems. In working systems many processes run simultaneously and influence each other, so time analysis is a complex task. The time analysis includes a number of different approaches. The most important ones are the worst-case time analysis, the performance analysis, and benchmark tests. Each of the methods has its specific features, advantages, and disadvantages. However, in the time analysis, a convolution operation, as a mathematical tool for analyzing the composition of processes, could be helpful.

1.1. Worst-case time analysis

The worst-case time analysis usually concerns industrial systems with strong time constraints (hard real-time system) [1]. Such systems require time determinism. The determinism is specified by values of the response time due to process requirements. From the perspective of the process, exceeding the maximum response time is unacceptable. Therefore, the primary parameter of evaluation of the system is the maximum response time. Time analysis is performed on the basis of pessimistic execution time of specific tasks in the system. These times are based on the maximum allowable time specification (time-out) for hardware and software and algorithms that determine the performance of a given system component.

This approach is sufficient to answer the question whether the system during the design phase meets time requirements imposed by the industrial process. However, it does not provide the information about a typical operation, in particular, about average processing time of the tasks. A statistical approach, which provides additional information on characteristics of a real working system, is taking into account time distributions of the tasks.

In the time analysis, the response times of the system are measured. As a result of a continuous observation of response times that occur at specific points of time, a discrete function of response times is obtained. By measuring the time for analysis, the area of analysis can be extended to analyze other values after recording the data containing the system response times. It can take into account not only the maximum values but also other values. The probability mass function of response times may be its representation. In this way the phenomena observed in the measurements can be interpreted statistically using the probability mass function.

However, the analysis and statistical interpretation of measurement results of the computer system response time turns out to be far more complex than the time analysis based on the expected time limits that result from time determinism.

1.2. Performance analysis

The performance analysis is a good method for finding errors in the process of creating and running the software, especially for systems operating in a continuous mode. The performance analysis usually involves the measurement of average values, whilst the time analysis is based on the analysis of maximum values. The following applications of the performance analysis of computer systems [2] can be distinguished:

- comparison of alternative solutions;
- checking the influence of new functionality on the system;
- tuning the system;
- monitoring the relative changes in system performance (acceleration, deceleration);
- detection of errors in creation and development of software;
- performance planning for solutions that do not exist yet.

Monitoring and the performance analysis is a fundamental detection test used by departments of quality control in the companies producing systems operating in a continuous mode. An important advance in these applications would be a possibility of transforming the methods of performance measuring of computer systems from a test of simple detection to the methods of diagnosis. The diagnostic tests can detect performance degradation. However, it is important not only to detect the degradation, but also to identify the component (subsystem), which caused the performance degradation. Diagnostic methods are intended to identify this element, such as locating the bottleneck in the system. Note also, that without the measurement process, it is not possible to enter any phase of validation or verification process in the performance evaluation study, presented in [3].

The performance analysis and time analysis are distinct areas of computing research. In the performance analysis the essential importance is attached to the probabilistic analysis and statistical models. In the time analysis of real-time systems the main problem is to identify time determinism in operation of the system. An open question is whether these areas have a common part. The method proposed in this article can be a common part of these two areas, with particular emphasis on the possibility of using measurements to diagnose the system. It may be an extension of measurement methods towards a diagnosis, in terms of determining the cause of changes in performance and searching for items that should be corrected.

1.3. Benchmarks

As mentioned before, in the time analysis of computer systems the worst-case analysis and analysis based on average values are normally used. Benchmarks are widely applied for comparing various properties of systems. On the other hand, the performance analysis usually measures average values. Such measurements are insufficient to identify many important characteristics of the system, for example the worst-case analysis, and do not provide much information which measurement data should provide. If there is degradation of performance, then benchmarks do not provide the answer to the question about the reason.

Based on the assumption that correct implementation works better and more efficiently than implementation with functional errors, the prototype of the new system can be quickly diagnosed. Using the performance tests [4] for detecting errors and verifying the correctness of implementation requires:

- development of criterion for quantitative evaluation of system performance;
- development of measurement methods and measurements on referencing systems;
- development of model of the system;
- determination of asymptotic limits;
- searching for states of system performance and methods of identification;
- searching for measures of performance, based on parameters easily accessible by measurement;
- studying the statistical features of real systems;
- development of measurement methods for non-stationary systems.

One of the common approaches [5] applicable to the testing and debugging of prototype applications is measuring the performance [6]. Instead of decomposition of the system and verification of the particular components, subsystems, or functions, the performance of the whole system is examined.

2. Benchmark-type tests

Benchmarks usually measure performance for a particular workload. Let us analyze how the system behaves for such particular load. Symbols in the description of benchmarks presented here are used in the descriptions of queuing systems [7]. Let systems x, y, z have the same structure and consist of the following resources: processor *cpu*, disk *hdd* and network card *net*. Suppose that for the systems tests, workload references η_1, η_2, η_3 have been developed for the maximum workload of the individual elements of the system, i. e. *cpu*, *hdd* and *net* respectively. The measurement results indicate a measure of performance, which is assigned to the system by a benchmark test. During benchmark testing, when testing a single element of the system, other elements are also involved. For example, during the test of the drive *hdd*, the CPU, DMA and memory operations also take part.

Workloads η_1, η_2 and η_3 have the following properties. For fractional ϵ we choose the workload η_1, for maximum use of the resource *cpu*

$$\eta_1 \rightarrow U^{(cpu)} = 1 - \epsilon \tag{1}$$

where $U^{(cpu)}$ is resource utilization. Level of resource utilization is in the range [0-1]. Then the following relationship holds between the response time R of the system and the response time $R^{(cpu)}$ of the *cpu*.

$$R \geq R^{(cpu)} \tag{2}$$

In the test η_1 the resource with the maximum workload is the *cpu*. The resource *cpu* is the bottleneck then, so the system cannot have a better response time R than the response time of the most loaded resource. So similarly for the test η_2, where the most loaded resource is *hdd*.

$$\eta_2 \rightarrow U^{(hdd)} = 1 - \epsilon \tag{3}$$

then

$$R \geq R^{(hdd)} \tag{4}$$

For η_3 test

$$\eta_3 \rightarrow U^{(net)} = 1 - \epsilon \tag{5}$$

and then

$$R \geq R^{(net)} \tag{6}$$

Using a model of the system in which tasks performed by the system are served by resources, the average system response time $\bar{R}$, according to the Little law [8], [9], [10], [11], is

$$\bar{R} = \frac{\bar{N}}{X} - \bar{Z} \tag{7}$$

where $\bar{N}$ is the average number of clients in the system, X is the throughput, that is the number of tasks performed per time unit, Z is the average time (interval) between tasks generated for the system. Throughput is understood as defined in queuing theory and is the ratio of processed tasks during the observation time T. Using operational research approach we can determine the current value of X by counting the number of processed tasks in the system during the observation time T. Assuming that tasks are executed in series (not in parallel) and $D^{(i)}$ is working time of the resource i, working time D of all resources is

$$D = \sum_i (D^{(i)}) \tag{8}$$

It is typically assumed for the benchmark tests that realization runs for one client only and there is no interval between tasks. So $Z = 0$ and $N = 1$. Thus, if there is only one task, this task does not wait, so the time of execution of the task in all resources is the response time of the system. So, if $N = 1$ then $R = D$. For each of the systems x, y and z the same rule applies, so

$$N = 1 \rightarrow R^{(i)} = D^{(i)} = B^{(i)}V^{(i)} \tag{9}$$

where $B^{(i)}$ is the execution time of the task in the resource i, $V^{(i)}$ is the number of visits (execution) of the task in the resource i. The working time of the resource consists of several visits of the task. In benchmark tests the value $B^{(i)}$ is usually large because typical tasks are long. Thus the number of visits $V^{(i)}$ in the resource is reduced.

Assume, that for the workload η_1 and the same number of visits V_{η_1} in the systems x, y and z, the system x has the shortest working time of the *cpu* resource. Thus the system x has the best *cpu* unit.

$$\min(D_x^{(cpu)}, D_y^{(cpu)}, D_z^{(cpu)}) = D_x^{(cpu)} \tag{10}$$

Similarly, assume that for the workload η_2 and the same number of visits V_{η_2}

$$\min(D_x^{(hdd)}, D_y^{(hdd)}, D_z^{(hdd)}) = D_y^{(hdd)} \tag{11}$$

For the workload η_3 and the same number of visits V_{η_3}

$$\min(D_x^{(net)}, D_y^{(net)}, D_z^{(net)}) = D_z^{(net)} \tag{12}$$

Little law [12] and assumptions $Z = 0$ i $N = 1$, usually adopted for benchmarks, show that

$$\bar{R} = \frac{1}{X} \tag{13}$$

The following conclusions may be drawn from the above considerations. For the workload η_1 the system x has always the shortest response time R of all systems x, y and z.

$$R_x = \min(R_i), i \in \{x, y, z\} \tag{14}$$

The throughput X_x of the system x is

$$X_x = max(X_i), i \in \{x, y, z\} \tag{15}$$

For the workload η_2:

$$R_y = min(R_i), i \in \{x, y, z\} \text{ and } X_y = max(X_i), i \in \{x, y, z\} \tag{16}$$

Similarly, for the workload η_3:

$$R_z = min(R_i), i \in \{x, y, z\} \text{ and } X_z = max(X_i), i \in \{x, y, z\} \tag{17}$$

The system with the best *cpu* always is the best in the test η_1, in which *cpu* is the most loaded resource.

The obtained results of the benchmark allow to determine the arrangement and relationship between systems x, y and z. For various benchmarks the measurements as response time R and throughput X are obtained. If the benchmark is based on R, then the system is better if it receives lesser value of the test result. If the benchmark is based on X, then the system is better if a greater value follows from the test.

Though the results obtained in this analysis are simple, they show that benchmark tests reflect only simple dependencies occurring in the system. In order to stimulate the system in real-world conditions, for example in test η_j, the number of clients must be more than one. We can also expect that the interval between tasks will be non-zero. Then the relations between $V^{(i)}$ and $B^{(i)}$ are changed, so

$$N > 1, Z > 0, V_{x,y,z}^{(i)} \neq V_{x,y,z}^{(\eta)}, B_{x,y,z}^{(i)} \neq B_{x,y,z}^{(\eta)} \tag{18}$$

In such case the formula (9) does not apply and the formula (7) is valid instead of (13). The conclusion is that benchmark tests do not reflect the complexity of executing tasks in the real system. They also cannot be used as a measurement method in the validation and projection phase [3].

More information on the system operation can be achieved from the deeper analysis of response times of the processes. The response time of the process is the composition of the response times of its subprocesses. For such an analysis the operations of convolution and deconvolution

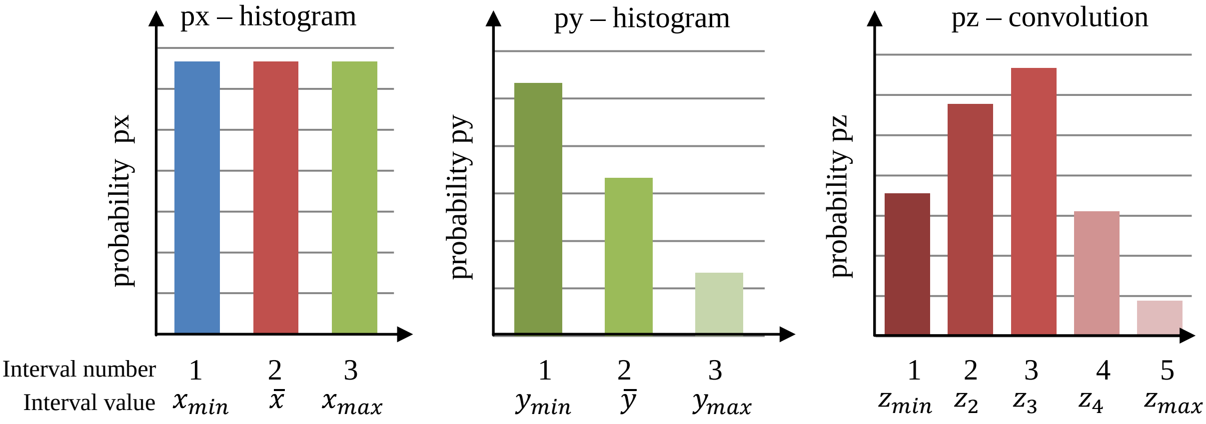

Figure 1. Histogram z calculated on base histograms of processes x and y for three ranges of values: minimum, mean, and maximum

performed on histograms of response times are helpful.

3. The convolution method in the time analysis of systems

Though convolution is well known method, its application in analyzing response time characteristics of particular components of complex system is a relatively novel approach [13], [14], [15], [16].

In the article it is particularly shown, that the convolution is relatively easy method to use, but the deconvolution is very sensitive when using for time characteristics measured in different times in independent components.

The presented method of time analysis of a compound computer system is based on convolution of two functions representing time-specific behavior of two subsystems that are parts of that system [17]. Convolution is considered one of the most important operations in the field of digital signal processing. Assume the existence of two independent subsystems x and y. A composition of both is a system z, so

$$z = x + y \tag{19}$$

The equation (19) represents also time dependences. It means that response time of system z is the sum of response times of subsystems x and y. Under this assumption it can be consequently stated, that the probability mass function pz of the response time of system z is equal to the convolution of probability mass function px and py of response times of subsystems x and y.

While executing the measurements for time analysis purposes, it is possible to record the average and minimum values, beside the maximum ones. The simplest case for introducing the method is when three ranges of values exist: minimum, mean, and maximum. The maximum value is collected during the worst-case analysis of a real-time system. The average and minimum values can be used in performance analysis. In addition to evaluation of the ranges, there is a need to know how often each value occurs. In the following discussion it is assumed that x and y are independent random variables, such as:

$$x \in \{x_{min}, \bar{x}, x_{max}\}; \quad y \in \{y_{min}, \bar{y}, y_{max}\} \tag{20}$$

For each value of the random variable, the probability of taking a given value can be determined. Thus, the probabilities that a variable represents the minimum, mean, and maximum value are known. The division of x and y into three values with the same intervals is used to derive the equations (21 - 23). The example of the convolution of two histograms is presented in Fig. 1. The consecutive probabilities of the resulting histogram pz for convolution shown in Fig. 1 could be calculated by simple formulas:

$$\begin{aligned} pz_{min} &= px_{min}py_{min}; \\ pz_2 &= px_{min}py_2 + py_{min}px_2 \end{aligned} \tag{21}$$

$$pz_3 = px_{min}py_{max} + px_2py_2 + px_{max}py_{min}; \quad (22)$$

$$pz_4 = px_2py_{max} + py_2px_{max}$$

$$pz_{max} = px_{max}py_{max} \quad (23)$$

The convolution method can be the useful tool for calculating time probability distributions. Thus, more information could be obtained than only minimum, average and maximum values. Referring to the definition of convolution, e.g., in [18], pz is a convolution of px and py.

$$pz = px * py = \int_0^\infty px(\tau)py(t-\tau)d\tau \quad (24)$$

The generalization of probability distribution pz for discrete values takes the following form:

$$pz_i = \sum_{j=0}^{i} px_j py_{i-j} \quad (25)$$

In the following considerations, the discrete counterparts of the continuous functions of probability distribution are used.

4. A case of compound process

Let us show an example of application of convolution for analyzing a simple process of writing data. Suppose the system uses the data buffering mechanism shown in Fig. 2. Buffering is implemented using two possible paths $x^{(1)}$ and $x^{(2)}$. The first one writes data to the buffer. This path is realized with probability p_1. The second path writes data to the buffer with t_0 delay, due to waiting for the access to the buffer. This path is realized with probability $1-p_1$. These paths are modeled by two statistical processes with the exponential response time, shifted relative to each other at t_0 and probabilities: p_1 and $1-p_1$ (Fig. 3a and Fig. 3b).

The two events are complementary. The distribution of the sum of these events (26) is shown in Fig. 3c. The write operation y is represented by the distribution py of response times, shown in Fig. 4b. It could be written as:

$$px = p_1px^{(1)} + (1-p_1)px^{(2)} \quad (26)$$

where: $px^{(1)}$ – probability mass function of execution time $x^{(1)}$ of buffering with delay, $px^{(2)}$ – probability mass function of execution time $x^{(2)}$ of buffering without delay, px – probability mass function as result of compound execution time $x^{(1)}$ and $x^{(2)}$.

Thus

$$pz = px * py \quad (27)$$

where: py – probability mass function of time execution of write operation, pz – probability mass function of write operation with buffering.

The distribution of the sum of events (Fig. 4a) in convolution with the py distribution (Fig. 4b) gives the resulting distribution (Fig. 4c). The resulting distribution can be observed by measuring the system z response time for write operations. Such a specific characteristic (dual peek) can be detected in practice [19]. So, the observed distribution can be analyzed in a better way than while obtaining only minimum, average and maximum values. The characteristics is explained as a convolution of component processes [20].

Changing the time of waiting for the access to the buffer represented by parameter t_0 and changing the probability p_1 of this waiting, change the shape of the resulting distribution. Resulting distributions for $p_1 = 0.5$ and various t_0 are presented in Fig. 5.

Depending on the parameters of the model, different characteristics of system response time distributions can be obtained. The distributions can be in the form of one peek or even separated dual peeks in some cases. So the measured practical results can vary depending on the behavior of the process. Distributions for $t_0 = 55$ and various p_1 are presented in Fig. 6.

The advantage of the method based on convolution over other methods is the opportunity to observe the entire time probability distribution instead of selected values. Another advantage is the ability of simulation of behavior of the system for the case when some component or its time characteristic has to be changed. Then, by substituting the time distribution only for this component and then making the convolution with distributions of other components, the time distribution for the whole system can be calculated.

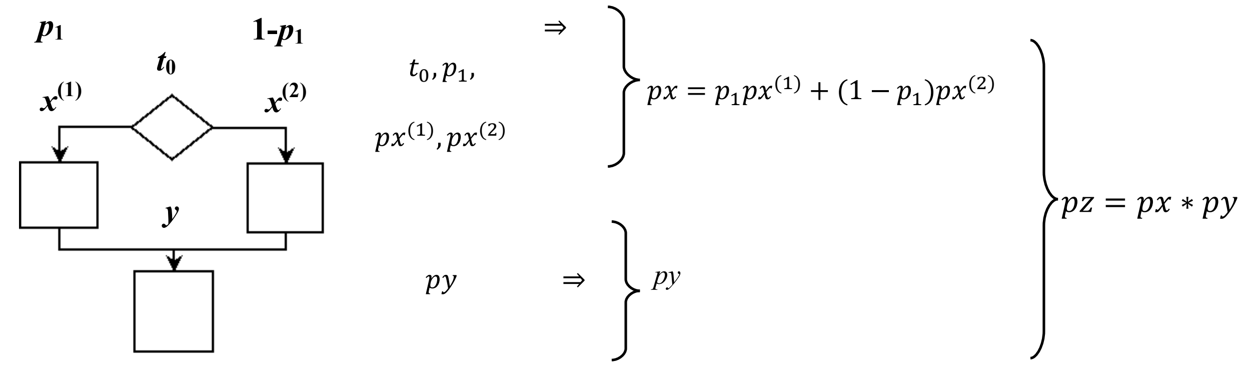

Figure 2. The example of system for writing with buffering

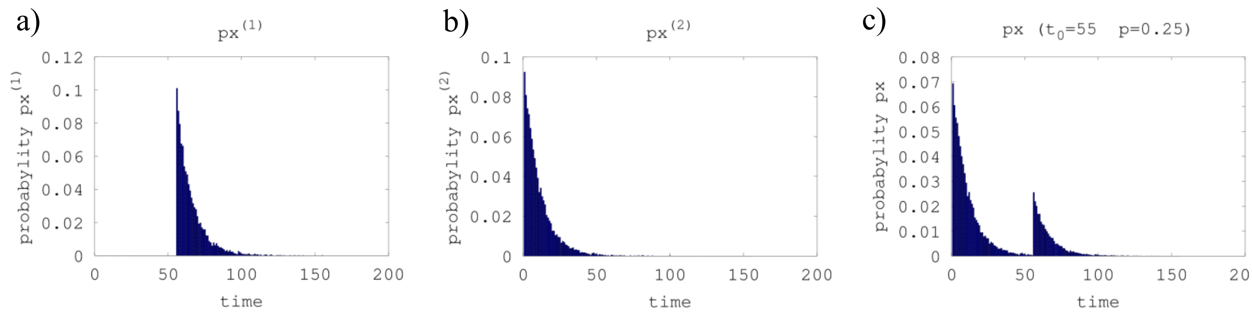

Figure 3. Distributions of response time for an example of a write system. $t_0 = 55$, $p_1 = 0.25$

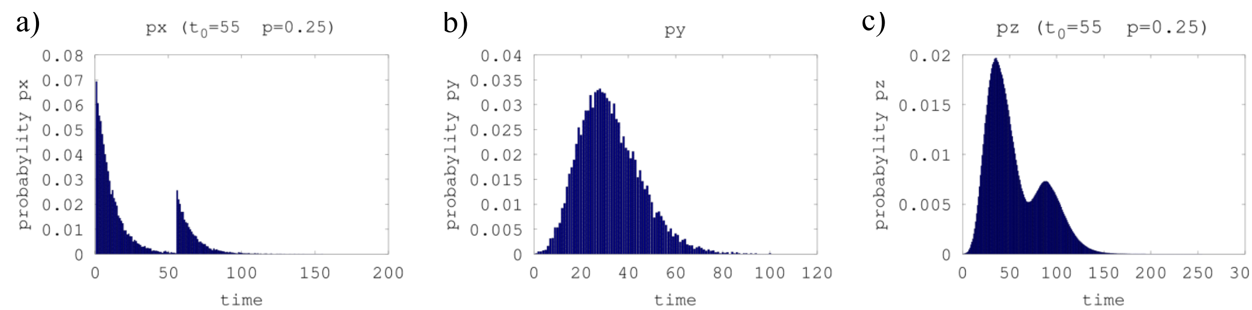

Figure 4. Resulting distribution pz as convolution of distribution px ('if' of events $x^{(1)}$ and $x^{(2)}$) with the distribution py

5. Analysis of measurement errors

In order to produce a good method of system analysis there is a necessity to consider that errors may occur in measurements. It may be caused by either inaccuracy of measurement or impossibility to measure all subsystems in the same time, so the measurements may be collected in different times. The result is the time inconsistency errors of consecutive measurement processes. Analysis of influence of measurement errors is important particularly in the operation of deconvolution. This process is not simple and clear because of sensitivity of deconvolution method. Convolution has the features as separation, commutativity, and associativity. The proofs are in [17].

The goal of deconvolution is to calculate time probability distribution for component y knowing time probability distributions of system z and

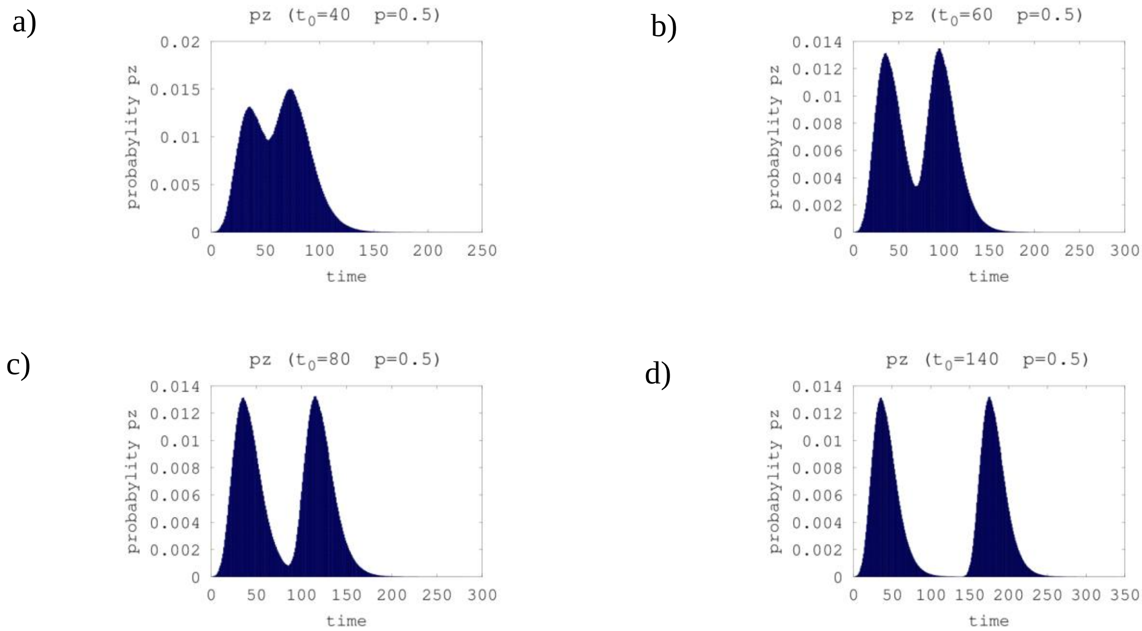

Figure 5. Distribution pz for $p_1 = 0,5$ a) $t_0 = 40$ b) $t_0 = 60$ c) $t_0 = 80$ d) $t_0 = 140$

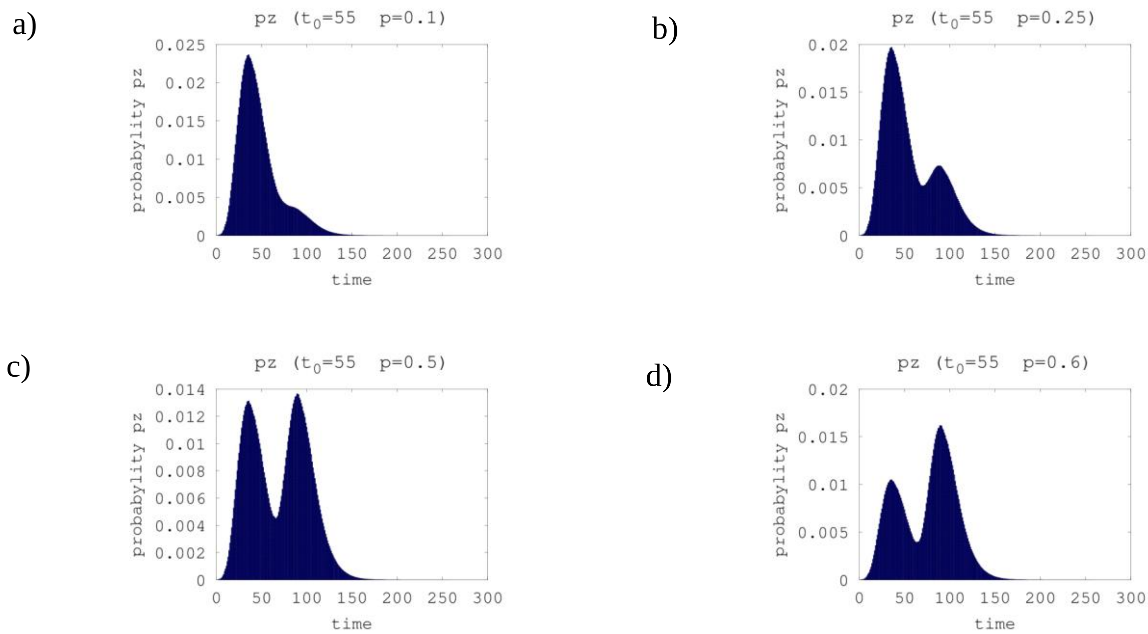

Figure 6. Distribution of pz for $t_0 = 0.5$ a) $p_1 = 0.1$ b) $p_1 = 0.25$ c) $p_1 = 0.5$ d) $p_1 = 0.6$

component x. If the form of an error in measurements z_m and x_m is known, then the result of equation (29) is identical to the solution of equation (28), under the condition that x and y are not distorted.

$$y = z - x \tag{28}$$

$$y = z_m - x_m \tag{29}$$

While measuring, a real response time, the z_m value is measured instead of z. It contains measurement errors e_x and e_y both for x and y, so

$$z_m = x + e_x + y + e_y \tag{30}$$

The total measurements error e consists of errors related to constituent processes (31).

$$e = e_x + e_y \tag{31}$$

Based on (30) and considering commutativity feature, the measured value z_m is:

$$z_m = x + y + e \tag{32}$$

To determine the value of y the two separated measurements have to be performed. During the one measurement the value x_m is collected (33):

$$x_m = x + e_x \tag{33}$$

During another measurement the value zm is collected (34):

$$z_m^{(error)} = x^{(error)} + y \tag{34}$$

There are errors acquired during the measurement of z_m. The $x^{(error)}$ is a response time from sub-system x during z_m measurement. These errors are immeasurable so instead of x_m the measured value is marked as $x^{(error)}$ in the case. It is also assumed that the values are similar (35).

$$x^{(error)} \approx x_m \tag{35}$$

The differences in the above values result from the facts that in the measurement of x_m the error e_y does not exist and also consecutive measurements of x are taken in different times. Thus, the assumption of the method while obtaining y is using x_m instead of $x^{(error)}$ in (34). From (34) the two mechanisms are considered. The first mechanism of occurring errors and the error analysis is done below in the form of (36).

$$y_{zm}^{(error)} = z_m^{(error)} - x \tag{36}$$

Formula (36) describes a situation in which the measurement error is introduced, because during the real system observation (z) the subsystem x behaves like x_m. Some environmental disturbances interact with the system x which do not exist during the observation of the isolated subsystem x. This disturbed subsystem is denoted as x_m. Unfortunately, the disturbances are not directly measurable. For the mean value the equations (37-40) represent the error introduced during the measurement.

$$\bar{y}_{zm}^{(error)} = \bar{z}_m^{(error)} - \bar{x} \tag{37}$$

$$\bar{y}_{zm}^{(error)} = \bar{x}_m + \bar{y} - \bar{x} \tag{38}$$

$$\bar{y}_{zm}^{(error)} = \bar{x} + e + \bar{y} - \bar{x} \tag{39}$$

$$\bar{y}_{zm}^{(error)} = \bar{y} + e \tag{40}$$

The result from (40) shows that measured mean value from y is charged with error e. If the calculations are not executed for the mean but for the probability mass functions p_{zm} and px, then the result is as below:

$$py_{zm}^{(error)} = pz_m^{(error)} \overset{/}{_{deconv}} px \tag{41}$$

$$py_{zm}^{(error)} = (px_m * py) \overset{/}{_{deconv}} px \tag{42}$$

$$py_{zm}^{(error)} = (px * pe * py) \overset{/}{_{deconv}} px \tag{43}$$

The equation (41) shows how the wanted py distribution can be obtained. The distribution found in such a way is laden by error pe convolved with the true distribution py (44):

$$py_{zm}^{(error)} = py * pe \tag{44}$$

The second mechanism of occurring errors during the measurements, other than shown in equation (37), is considered below:

$$y_{xm}^{(error)} = z - x_m \tag{45}$$

For the above formula (45) and during observation of the real system z the data is not affected by measurement errors derived from the subsystem x. Unfortunately, the interferences occur during the observation of the subsystem x and produce x_m. For the mean value the equations (46-48) represent error (45), introduced during the measurement.

$$\bar{y}_{xm}^{(error)} = \bar{z} - \bar{x}_m \tag{46}$$

$$\bar{y}_{xm}^{(error)} = \bar{x} + \bar{y} - \bar{x} - e \tag{47}$$

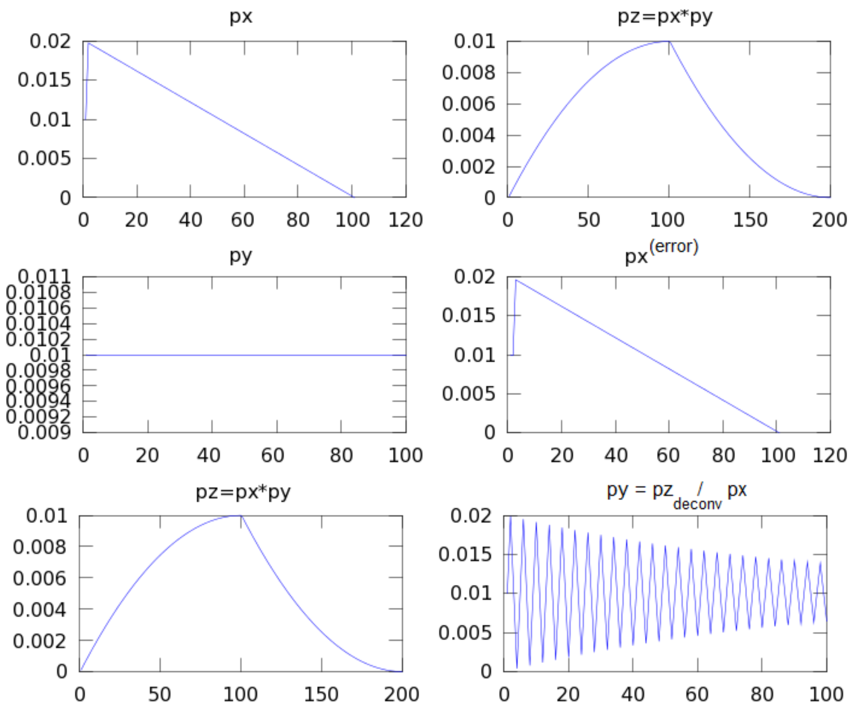

Figure 7. Example of the error influence: px, py, and pz on the left, pz, $px^{(error)}$ and deconvolved py on the right

$$\bar{y}^{(error)}_{xm} = \bar{y} - e \tag{48}$$

The equations (45) and (36) differ only in arithmetic operation in the equations (40) and (48). From the arithmetic point of view the mean value calculations (37-40, 46-48) are fully feasible for the any set of measurement data. Unfortunately, the situation changes radically for the application of deconvolution. The usage of inconsistent data i.e., collected from the consecutive measurement series taken at different times is not possible in a mathematically obvious way. The reason is high error sensitivity of the deconvolution operation. In order to perform the system decomposition (28), the data collected during the different ranges in the time of system activity can be used only with full awareness of the influence of the described error. Fig. 7 illustrates the phenomenon of deconvolution sensitivity. The example is based on the convolution of signals px and py and then the deconvolution of received pz and disturbed px marked as $px^{(error)}$. $px^{(error)}$ is slightly changed (near the value 2) in comparison to the original px. It can be seen, that small disturbance (error) of time probability distribution of component x causes very large disturbance in resulting probability distribution after deconvolution. For a probability mass functions pz and px_m, while using deconvolution, it is impossible to take into account the type of error (45), as shown in equations (49, 50).

$$py^{(error)}_{xm} = pz \overset{/}{_{deconv}} px_m \tag{49}$$

$$py^{(error)}_{xm} = (px * py) \overset{/}{_{deconv}} (px?+?pe) \tag{50}$$

Equation (50) by the usage of the operator '?+?' presents the mathematical problem which results in non-resistance of the methods of deconvolution to the error type (45). If pe is not convolved with px, then deconvolution made on the basis of the polynomial division method produces a large computational instability.

The presented considerations (37-50) show two different phenomena of errors. For the mean value the equations (37-40) compared to (46-48) differ only in the value of the error sign. Considering only the average values, the distinction of the mechanisms of error generation is unnecessary. However, it differs in the case of deconvolution of probability mass function. In the case (43-44) the considerations are fully mathematically correct.

The error values are added (convolved) (44) to the value of probability mass function $py_{zm}^{(error)}$. More precisely: added errors are the values of probability mass function pe. Unfortunately, using the deconvolution method is mathematically incorrect for the case (49-50). This is because the correctness of the physical implementation of measurement is not met. In the measuring method assumes that the measurements of the systems x and z are taken at different times. Thus the circumstances where the time signals are measured in physical conditions, in which the convolution of probability mass functions of signals exist, do not occur. It can be only supposed that for the separated measurements $z_1..z_l$ and $x_1..x_k$, and for the stable running conditions, the probability mass function px was the same in both measurements. Unfortunately, if px is changing during the measurement $z_1..z_l$ in relation to the measurement $x_1..x_k$, then the assumptions are not met and deconvolution method fails. In the calculation of the mean value (46-48) this problem does not occur.

6. Conclusions

While collecting measurement data, i.e., the response times, one does not receive only one state of the system, but a composition. Complex systems are non-linear, their analysis cannot be based on the simple method of measurement and analysis. The considerations conducted here show that measurements of the execution time of processes have much more complex statistical description than for example a simple description coming from benchmark tests. A statistical description delivers a dynamic character in contrary to a fixed value obtained in a given moment of system activity. The model of a queuing system, with appropriately selected assumptions, and ignoring insignificant details may, but may not, simulate basic rules of the system to be modeled.

The system usually works as many processes executed through many paths. The convolution method could be a useful tool for the analysis of behavior of complex processes and their time analysis, in a statistical meaning. Response times of the system, for many operations, sometimes give characteristic forms of probability mass functions that can be explained as the convolution of response times of subprocesses. Using the presented method, the entire time probability distributions of the system can be obtained. The method could also be used to simulate changes in any component of the system by simply modifying its probability mass function and then calculating the convolution. However, using modeling and convolution for diagnostics of a system requires experience. The analysis method depends heavily on the recorded measurement values. In such a modeling also measurement errors must be considered, particularly in deconvolution process. The deconvolution method may be also considered as a tool for such an analysis i.e., for obtaining time distribution of the component of the system for which the measurement cannot be collected. However, as shown above, deconvolution is a sensitive method and difficult for a practical application.

References

[1] G. Pratl, D. Dietrich, G. P. Hancke, and W. T. Penzhorn, "A new model for autonomous, networked control systems," *IEEE Transactions on Industrial Informatics*, Vol. 3, No. 1, feb. 2007, pp. 21 –32.

[2] D. J. Lilja, *Measuring Computer Performance: A Practitioner's Guide.* Cambridge University Press, 2005. [Online]. http://books.google.pl/books?id=R8RLniX5DNQC

[3] E. D. Lazowska, *Quantitative system performance: computer system analysis using queueing network models.* Prentice-Hall, 1984. [Online]. http://books.google.pl/books?id=NNZQAAAAMAAJ

[4] D. A. Menascé, V. A. F. Almeida, L. W. Dowdy, and L. Dowdy, *Performance by design: computer capacity planning by example.* Prentice Hall PTR, New Jersey, USA, 2004.

[5] G. Bjedov, "Using open source tools for performance testing," *Google London Test Automation Conference (LTAC) Google Tech Talks*, September 2006. [Online]. http://video.google.com

[6] R. Blum, *Network performance open source toolkit: using Netperf, tcptrace, NIST Net, and SSFNet.* Wiley Pub., 2003. [Online]. http:

//books.google.pl/books?id=ECt5ycQ9D7YC

[7] T. Czachórski, *Modele kolejkowe systemów komputerowych*, ser. Skrypty Uczelniane - Politechnika Śląska. Wydawnictwo Politechniki Śląskiej, 1994. [Online]. http://books.google.pl/books?id=50KkPgAACAAJ

[8] J. D. C. Little, "A proof for the queuing formula: $\mathsf{L} = \lambda\mathsf{W}$," *Operations Research*, Vol. 9, No. 3, May/June 1961, pp. 383–387.

[9] S. Stidham, Jr., "A last word on $\mathsf{L} = \lambda\mathsf{W}$," *Operations Research*, Vol. 22, No. 2, March/April 1974, pp. 417–421.

[10] F. Beutler, "Mean sojourn times in Markov queueing networks: Little's formula revisited," *Information Theory, IEEE Transactions on*, Vol. 29, No. 2, mar 1983, pp. 233 – 241.

[11] P. W. Glynn and W. Whitt, "Extensions of the queueing relations $\mathsf{L} = \lambda\mathsf{W}$ and $\mathsf{H} = \lambda\mathsf{G}$," *Operations Research*, Vol. 37, No. 4, July/August 1989, pp. 634–644.

[12] L. Lipsky, *Queueing Theory A Linear Algebraic Approach.* Springer New York, 2009. [Online]. http://dx.doi.org/10.1007/978-0-387-49706-8_3

[13] J. P. Lehoczky, "Real-time queueing theory," in *Real-Time Systems Symposium, 1996., 17th IEEE*, dec 1996, pp. 186 –195.

[14] J. L. Diaz, D. F. Garcia, K. Kim, C.-G. Lee, L. Lo Bello, J. M. Lopez, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, 2002, pp. 289 – 300.

[15] L. Abeni, N. Manica, and L. Palopoli, "Efficient and robust probabilistic guarantees for real-time tasks," *Journal of Systems and Software*, Vol. 85, No. 5, 2012, pp. 1147 – 1156. [Online]. http://www.sciencedirect.com/science/article/pii/S0164121211003232

[16] M. Santos, B. Lisper, G. Lima, and V. Lima, "Sequential composition of execution time distributions by convolution," *Proc. 4th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS 2011)*, November 2011, pp. 30–37.

[17] J. Yeh, *Real analysis: theory of measure and integration.* World Scientific, Danvers, USA, 2006. [Online]. http://dx.doi.org/10.1007/978-0-387-49706-8_3

[18] A. Mattuck, *Introduction to analysis.* Prentice Hall, 1999. [Online]. http://books.google.pl/books?id=N0FkQgAACAAJ

[19] S. Wideł, J. Flak, and P. Gaj, "Interpretation of dual peak time signal measured in network systems," in *Computer Networks*, ser. Communications in Computer and Information Science, A. Kwiecień, P. Gaj, and P. Stera, Eds. Springer Berlin Heidelberg, 2010, Vol. 79, pp. 141–152. [Online]. http://dx.doi.org/10.1007/978-3-642-13861-4_14

[20] ——, "Analysis of time measurements in network systems using decomposition on subprocesses," in *Computer Networks*, ser. Communications in Computer and Information Science, A. Kwiecień, P. Gaj, and P. Stera, Eds. Springer Berlin Heidelberg, 2011, Vol. 160, pp. 70–79. [Online]. http://dx.doi.org/10.1007/978-3-642-21771-5_9