TESTING OF SOFTWARE RESPONSE TIME IN VARIOUS COMPUTING ENVIROMENTS

LAURIS PLUME, VITALIJS OSADCUKS, ALDIS PECKA

Latvia University of Agriculture, Latvia
e-mail:plume.lauris@gmail.com

Abstract: *The article describes a study about real-time performance of conventional PC by testing its response time to external events on a serial port in various conditions. For now PC's are mainly used as the supervisory control and data acquisition units on the highest level of PLC system. The results of the study will clearly show the possibilities of addressing to a conventional PC more time-critical tasks e.g. controlling microclimate in industrial buildings, apartments or green-houses; controlling slow to moderately fast changing processes in bioreactors; temperature control for drying or material melting process etc. The goal of the tests is to determine the real-time performance of software running in runtime system environment on conventional home and office computing systems with common PC hardware and multi-tasking desktop operating systems. The study will be performed on PC using a variety of OS (Windows 2000, Windows XP, Windows7 x32) and two types of runtime systems: Oracle Java Runtime and Microsoft .NET Common Language Runtime. Loading of the PC's components by concurrent tasks also will be considered. Microcontroller unit (MCU) will be used to generate the signal to the serial port to which the PC should respond and also to accurately measure the computer's response time, i.e. the MCU is used as a frame sender and computer as a forwarder. The time is measured between moment when the frame is sent out and received back to the MCU. The measured time is stored on the MCU. The computer software pools the serial port buffer to check if there is any data and once the frame is received the software immediately transmits the frame back to the sender. As a result the actual response time of the software running on the specified hardware depending on computer load and operating system features is determined. At the end of the study statistical analysis will be presented.*

Keywords: real-time, Windows 2000, Windows XP, Windows7 x32, microcontroller, PC, serial port.

Introduction

The article describes a study about real-time performance of conventional PC by testing its response time to external events on a serial port in various conditions. For now PC's are mainly used as the supervisory control and data acquisition (SCADA) units on the highest level of PLC system. Due to complexity and human user oriented software PC's are not considered to be hard real-time devices in classical interpretation (Joseph, 2003).
The results of the study will clearly show the possibilities of addressing to a conventional PC more time-critical tasks e.g. controlling microclimate in industrial buildings, apartments or green-houses; controlling slow to moderately fast changing processes in bioreactors; temperature control for drying or material melting process etc. (Vilums, 2010)
Lately, in order to run user software, the virtual machines or runtime systems (Oracle Java, MS.NET Framework) are used more frequently. They have continually updated access to development tools and code libraries, they ensure software portability between different platforms, code modularity and reusability (Jönsson, 2004). Consequently, decision is made to test the PC response time to the virtual machines instead of using native code. Another reason that .NET and Java Runtime (Segovia,1999) environments are used, because it would be difficult to compare different OS in native code. It is decided to test three OS and two virtual machines using different test methods in this research. Additional factors are number of external events and frequency of these events as well as load of various PC components.
Research hypothesis is that the PC (Audette, 2004) can be reliably used for automatic control systems with response time 100 ms ... 1 s and longer. Goal of the study and conducting these tests is to determine computer and software response time in different environments and at diverse combinations of load of the computer components.

Materials and methods

Software and hardware:
Microsoft Windows operating systems where chosen as an environment to run the tests: Windows 2000, Windows XP and Windows 7. The study will be performed on x86 based PC. Two types of runtime systems will be used: Oracle Java Runtime and Microsoft .NET Common Language Runtime. Windows OS were used because of the popular application on home and office use and availability of various software tools.
Tests are carried out using DELL Latitude D830 portable computer with following hardware configuration: Intel® Core™2 Duo CPU T8300 @2.4 GHz, 2.00 GB RAM, 74.53 GB Standard disk drive, Mobile Intel® 965 Express Chipset Family. All necessary up-to date drives where installed on operating systems.

Microcontroller (MCU) development board Microchip Explorer16 with PIC24FJ 16-bit device running at 32 MHz is used as real time external device, from which data should be read and written. Software has been developed in C language using C30 student's version compiler and MPLab IDE environment. The purpose of software is to generate simulated "sensor" data, send the data out using serial port to PC, receive "command" data back from PC, measure the PC's response time and store results. Microcontroller generates and sends to PC "sensor" data frames with various periods. Each data frame contains the same data value "2220", which is recognized by PC software and responded as a "command" with the same value. Experimental data flow is shown in Fig. 1.

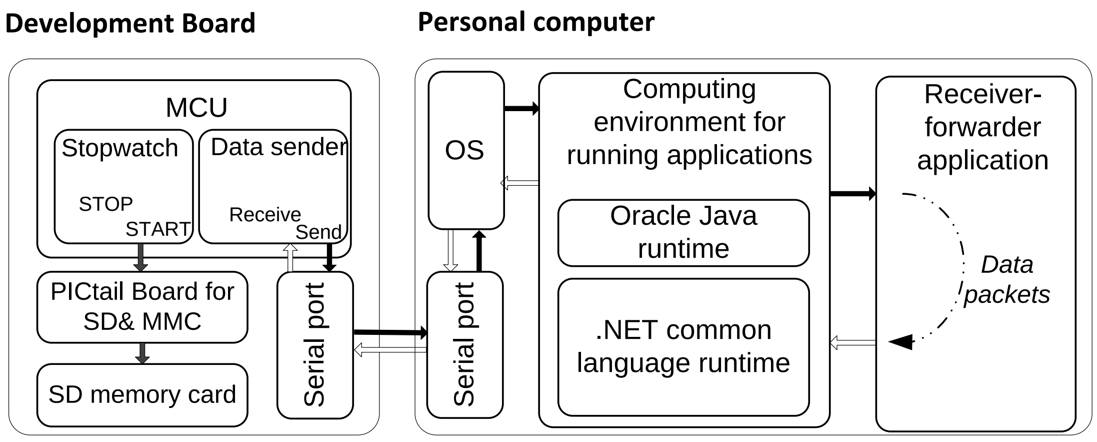


Fig.1. **Route of data frames travelling through components of PC and Development board:** ➡ shows the sent data frames; ⇨ shows forwarded data frames; ➡ measured result

The data frame is put in serial UART (Universal Asynchrony Receiver Transmitter) buffer and timer is started. The data frame is sent to the converter from the UART to RS-232 port, from which it is then sent to the PC.
The data is sent at a constant speed from the serial port of controller to serial port of PC. PC reads program port buffer if the data is in the buffer, it is recognized by the .NET or Java application and then sent back to the serial port.
When the controller receives the data, UART performs a hardware interrupt, the data is read from the UART buffer and placed in the controller memory buffer, it is then checked whether the received frame is with the value "2220" and the timer is stopped. Accuracy of the measurement is affected by MCU interrupt latency which is 4 CPU command cycles (about 0.25 us) and timer tick, which is 1 us. Approximate PC response times are more than 1 ms, consequently the measurement accuracy is better than 0.1 % of measured values.
Once data has been received, then the timer measures time data frame has spent traveling back and forth and the results are recorded in the SD memory card. Experimental set-up is shown in Fig. 1.

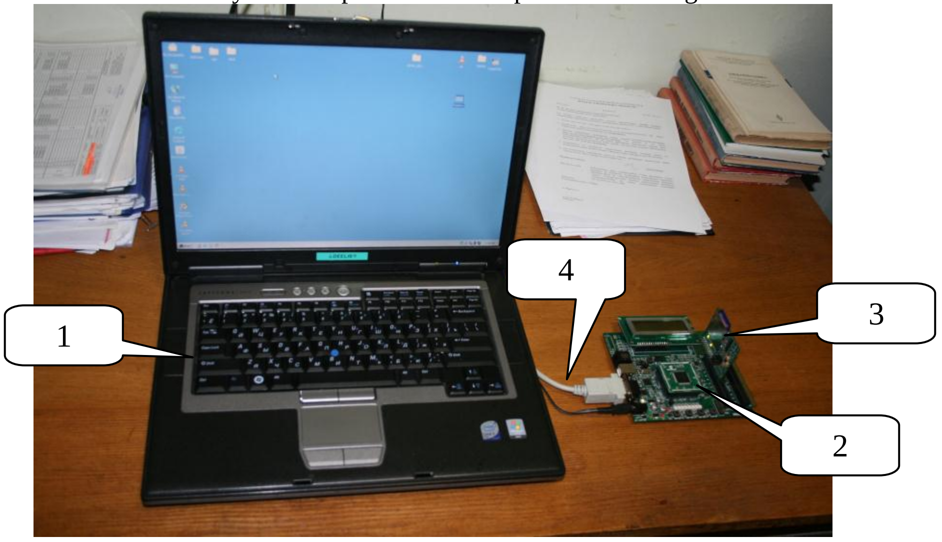


Fig. 2. **Experimental set-up:**
1 – PC used for tests; 2 – MCU development board; 3 – SD memory card; 4 – serial cable RS-232

The affected response time is studied under effect of four factors:

- OS (Windows 2000, Windows XP, Windows 7, 32 bit);

- runtime system (.NET, program written in C# and Java);
- 5 test methods;
- 4 PC load conditions.

Test methods:

MCU is used to generate the signal to the serial port to which the PC should respond and also to accurately measure the computer's response time, i.e. the MCU is used as data frame sender and computer as a replier. The time is measured between moment when the frame is sent out and received back to the MCU. The measured time is stored on the MCU. The computer software monitors the serial port buffer to check if there is any data and once the data frame is received the software immediately transmits the same data back to the sender. In both runtime systems built in software event mechanisms are used to respond to the MCU data. As a result the actual response time of the software running on the specified hardware depending on computer load and operating system features is determined.

Methods used for the tests are shown below (Tab. 1). The number of frames and delay time is diverse in order to simulate real environment. There will be five different tests carried out on each combination of OS, runtime environment and load of specific PC component. Each test has different number of data frames and/or delay time. Delay in seconds is a time difference between previous PC response received by MCU and next data frame sent out from MCU to PC.

Configuration for serial COM port is:

- Baud rate: 115200;
- Data bits: 8;
- Handshake: None;
- Stop bits : One;
- Parity: None.
- Input buffer size: default 4096 for .NET and Java
- Output buffer size: default 2048 for .NET and Java

Taking into consideration the baud rate, start and stop bits, the time needed for sending and receiving 4 byte data frame is $2 \cdot 4 \cdot 10 \cdot 115200^{-1}$ s = 0.7 ms. The software of PLC development board is programmed in such a way that if PC does not respond in 60 seconds, the software sends out next data frame.

Table 1

Test methods

No.	Number of frames	Delay, *s*
1	30	0
2	30	1
3	15	10
4	5	60
5	2	120

Load simulation for different components of PC:

Originally it is decided to use software that would simulate load for all computer components. In order to ensure that tests are being conducted under equal conditions, it is necessary to use a single developer's software program for testing all the various OS. The other option would be to use various developer programs with the same computer component load simulation algorithms. Due to the fact that all the computer components load for both of these options are not available, we have selected options shown in the table below (see tab.2).

The reason for this methodological choice is the universality of these applications and functions. File archiving is possible with the WinZip program, memory load is simulated by Obrut software and file copying is supported on all Windows OS – Windows 2000, Windows XP, and Windows 7 x 32. Obrut creates high memory and/or processor loads and thus simulate a slow or overloaded computer. Obrut can simulate varying levels of single or dual CPU usage intensity at three levels of regularity, and both virtual and physical memory loads.

Table 2

Method of load for separate components of PC

No.	Component of PC	Method of load	Application/OS funcion
1	RAM	Load simulation	Obrut
2	CPU	Archive of video file	WinZip
3	HDD	Process of copying 3 5 GB files to different locations and partitions of HDD	copy/paste

Computer has several components, which under load could affect the response time. For real situation the load can be different, the fallowing tests are going to be made (tab.3).

Table 3

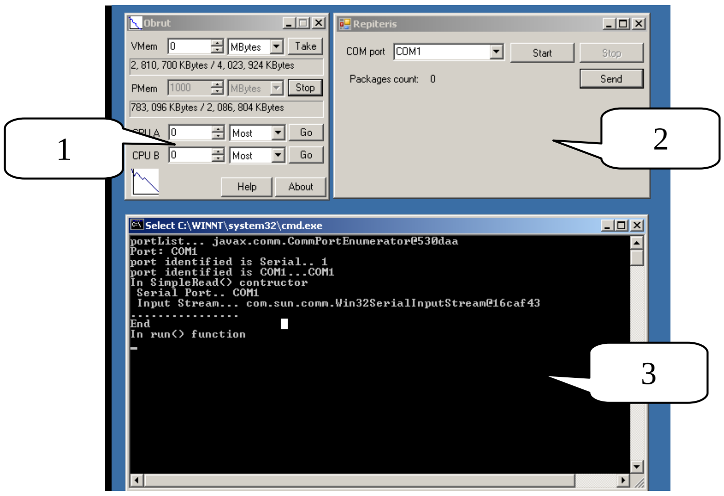
Load for components of PC – for testing all OS

Nr.	**Combination**
1	RAM
2	CPU
3	HDD
4	No load for computer components

The experimental results are analysed graphically and statistically using 3 and 4 factor univariate analysis of variance (ANOVA). SPSS20 statistical software is used for statistical analysis.

Results and discussion

Research is made online on a software program that would simulate the load on components of PC. The list of these software programs is wide enough.
To simulate load on RAM Obrut software is chosen. The reason why it is chosen is simple – this software meets specified requirements for the tests – it supports all operating systems that are included in the tests – Windows 2000, Windows XP and Windows 7 x32. Data compressing program WinZip is used to load CPU component. WinZip is supported on all the Windows operating systems so it is a good choice to use it.
Windows is currently the most popular OS used daily both at home and at work. That is the reason for decision to test different Windows OS first. The next step would be to test the Mac OS and Linux OS. It may be also good idea to test different OS used on tablet PC's and smartphones. There could be problem on finding appropriate software program that would be useful to test separate components on all these PC devices. All of the software programs are likely to have their own unique algorithm to load components of PC. Just in case if the algorithms are the same in some of the software programs, it would be impossible to verify that assumption because most developers are reluctant to reveal their algorithms. It is therefore necessary to use a single software program that is developed by the same software developer. Of course appropriate research on such software would have to be made anyway as there is a little doubt that the software would not use the same algorithm irrespective of the OS.
Initially we made tests where all the components where under load simultaneously, i.e., CPU, RAM and HDD. This test failed with the methods we used. The reason why it failed is because the moment when RAM is fully loaded, the WinZip program which is archiving a file, will stop fully loading CPU with mathematical calculations.



```
portList... javax.comm.CommPortEnumerator@530daa
Port: COM1
port identified is Serial.. 1
port identified is COM1...COM1
In SimpleRead() contructor
 Serial Port.. COM1
 Input Stream... com.sun.comm.Win32SerialInputStream@16caf43
................
End
In run() function
_
```

Fig.3. **Screenshot of the Windows 2000 environment running the tests:**
1 – Obrut software that simulates physical memory (RAM) load; 2 – Sofware program developed in C# and running .NET Runtime environment, that reads COM port, receives and sends back data packages to PLC; 3 – Sofware program developed in Java and running Java Oracle Runtime environment, that reads COM port, receives and sends back data packages to MCU

Three measurement repetitions are performed for each 4-factor combination (OS – 3 cases, runtime system – 2 cases, test method – 5 cases, load condition – 4 cases). In each of 5 test methods respective numbers of frames (see Tab. 1) are sent to PC and average, maximum and minimum response times are calculated. It resulted in 1080 measurement values total. The results for average grouped by runtime environment and PC load conditions are shown in Fig. 4–7.

The results show that average response times for all tests are less than 6 ms except RAM load and Windows XP – CPU load – .NET environment combination. Slightly worse results for both environments are under HDD load condition and high request frequency with no pauses between requests (first test method). The worst response times are for RAM load and requests to the PC at low frequencies in tests methods 4 and 5. It should be pointed out that CPU load condition did not affect the response times significantly with the exception of windows XP operating system.

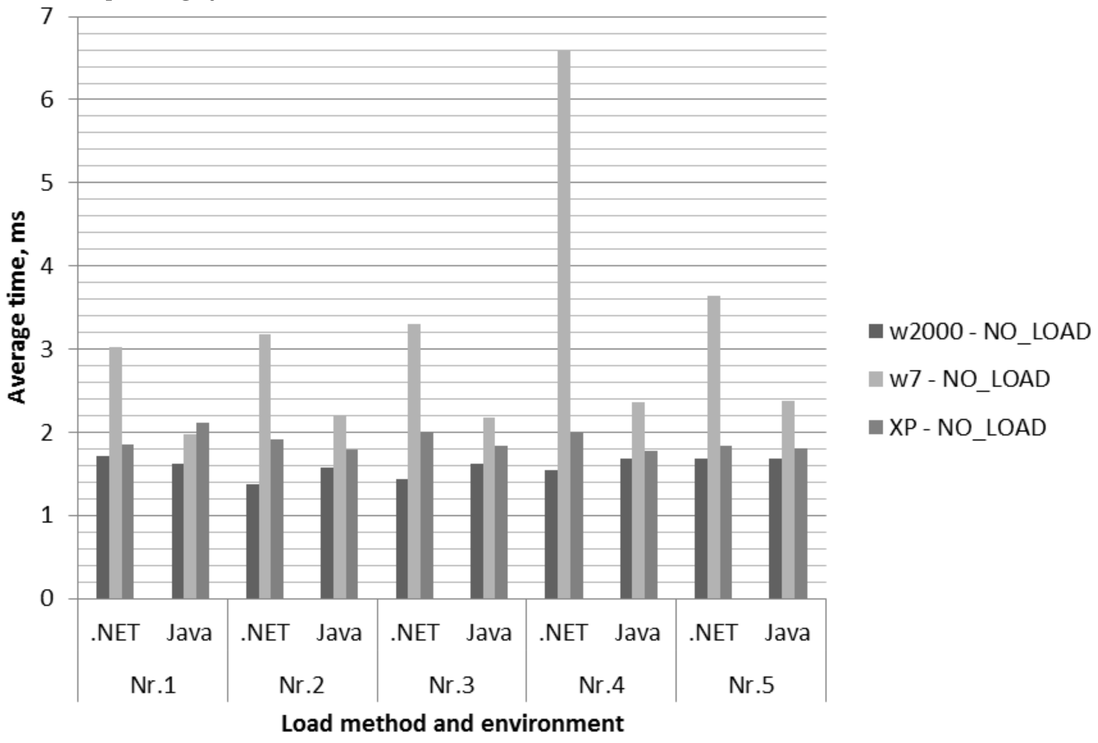


Fig. 4. **Tests for all OS, environments and load methods with no load on PC components**

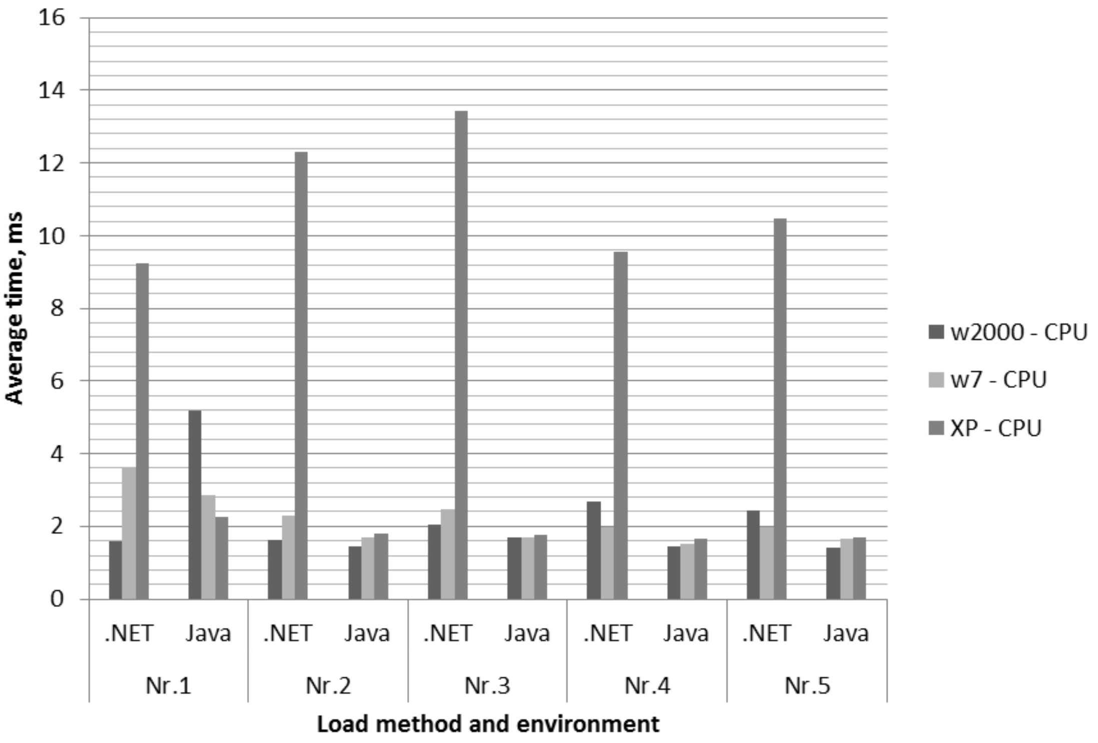


Fig. 5. **Tests for all OS, environments and load methods with load on CPU**

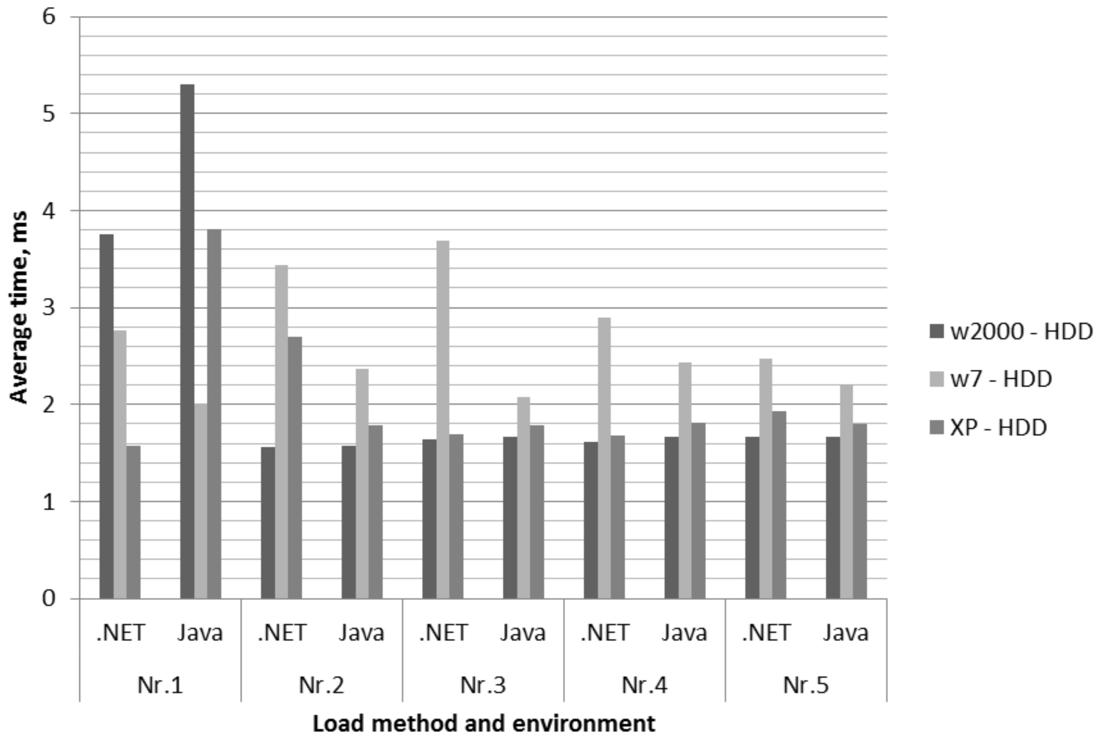


Fig. 6. **Tests for all OS, environments and load methods with load on HDD**

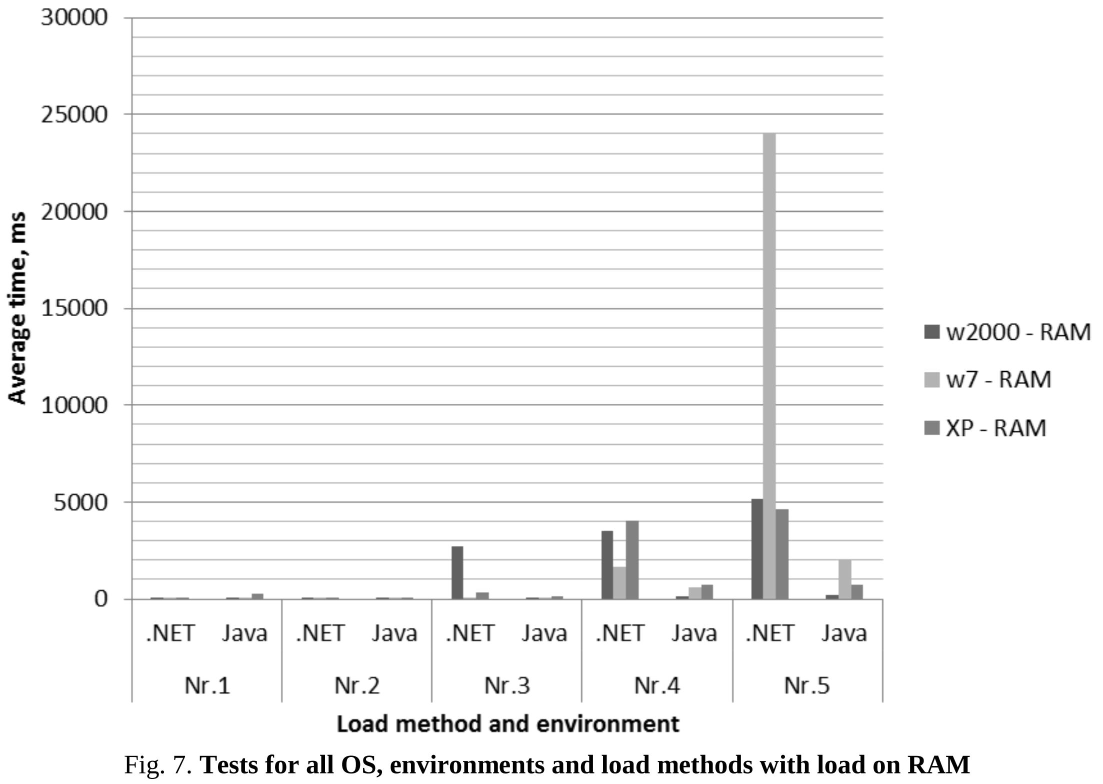


Fig. 7. **Tests for all OS, environments and load methods with load on RAM**

Charts of minimum and maximum values with respect to factors in general visually repeat average results therefore were not included in the paper. Maximum response times in conducted tests of more than 30 s were also observed under RAM load and Java application even failed to respond in several cases. Minimum response times for first 3 test methods differed insignificantly, but when responding on low frequency requests minimum response times reached 5 s.

Statistical analysis of variances generally approves conclusions from graphical analysis. In statistical difference between response times was assumed to be significant at the α 0.05 level.

Analysis with 4 independent factors shows that there is significant statistical difference for all factors except OS, so 3-way analysis was performed separately for both runtime environments. Pairwise comparison of average response times for loads for both runtime environments (3-factors each) and for all experiments (4-factors) reveals also that that only RAM load condition differs significantly at the probability 95 %. In all other cases p-value was more than 0.98. Pairwise comparison for different OS reveals that average values for .NET, Java (3-facftor) and all 4-factor analysis for Windows 7 significantly differ form other operating systems and are slightly worse in HDD, RAM and no load tests, but with no difference form Windows 2000 in CPU load test.

Conclusions

It can be concluded that main factors that affect response times of a conventional PC (Intel® Core™2 Duo CPU T8300 @2.4 GHz, 2.00 GB RAM, 74.53 GB Standard disk drive, Mobile Intel® 965 Express Chipset Family) for Java and .NET runtime environments and all tested operating systems were: load on RAM and necessity to respond on requests at relatively large time intervals: 60 – 120 s between requests.
Average response times for all tests except RAM load and Windows XP – CPU load – .NET environment combination are less than 6 ms. Maximum response times in conducted tests of more than 30 s were also observed under RAM load and Java application even failed to respond in several cases. Minimum response times for first 3 test methods differed insignificantly, but when responding on low frequency requests minimum response times reached 5 s.
It should be pointed out that only response time has been measured in this research. There are no tests made to determine reliability (system crash, from 1 to 2 min freeze) in long-term operation. These as well as more factor combinations are planned as further research.

References

Audette, M.A., Hayward, V., Astley, O., Doyon, M., McCallister, G.A., Chinzei, K., 2004. A PC-based system architecture for real-time finite element-based tool-specific surgical simulation, 5 pp.
Joseph, M., Goswami, A., 2003. Formal description of realtime systems: a review, 9 pp.
Jönsson, A., Johan, W., Broman, G., 2004. A virtual machine concept for real-time simulation of machine tool dynamics, 6 pp.
Segovia, A., Díazb, A., Garduño, M., 1999. Integrating native code into a Java application for controlling a mobile robot via a PC’s serial port, 8 pp.
Vilums, S., 2010. Bioprocess realtime fault detection and diagnosis based on model and expertise, 8 pp.