Journal of Applied Computer Science Methods

Published by University of Social Sciences

ACSM

Volume 4 Number 1 2012

University of Social Sciences, IT Institute

INTERNATIONAL JOURNAL OF APPLIED COMPUTER SCIENCE METHODS (JACSM)
is a semi-annual periodical published by the University of Social Sciences
in Lodz, Poland.

PUBLISHING AND EDITORIAL OFFICE:
University of Social Sciences (SAN)
Information Technology Institute (ITI)
Sienkiewicza 9
90-113 Lodz
Tel.: +48 42 6646654
Fax.: +48 42 6366251
E-mail: acsm@swspiz.pl
URL: http://acsm.swspiz.pl

Print: Drukarnia Wojskowa w Łodzi Sp. z o.o., ul. Gdańska 130, tel.: +48 42 637 76 77

AIMS AND SCOPE:

The **International Journal of Applied Computer Science Methods is** a semi-annual, refereed periodical, publishes articles describing recent contributions in theory, practice and applications of computer science. The broad scope of the journal includes, but is not limited to, the following subject areas:

Knowledge Engineering and Information Management: *Knowledge Processing, Knowledge Representation, Data Mining, Machine Learning, Knowledge-based Systems, Knowledge Elicitation, Knowledge Acquisition, E-learning, Web-intelligence, Collective Intelligence, Language Processing, Approximate Reasoning, Information Archive and Processing, Distributed Information Systems.*

Intelligent Systems: *Intelligent Database Systems, Expert Systems, Decision Support Systems, Intelligent Agent Systems, Artificial Neural Networks, Fuzzy Sets and Systems, Evolutionary Methods and Systems, Hybrid Intelligent Systems, Cognitive Systems, Intelligent Systems and Internet, Complex Adaptive Systems.*

Image Understanding and Processing: *Computer Vision, Image Processing, Computer Graphics, Pattern Recognition, Virtual Reality, Multimedia Systems.*

Computer Modeling, Simulation and Soft Computing: *Applied Computer Modeling and Simulation, Intelligent Computing and Applications, Soft Computing Methods, Intelligent Data Analysis, Parallel Computing, Engineering Algorithms.*

Applied Computer Methods and Computer Technology: *Programming Technology, Database Systems, Computer Networks Technology, Human-computer Interface, Computer Hardware Engineering, Internet Technology, Biocybernetics.*

DISTRIBUTION:

Apart from the standard way of distribution (in the conventional paper format), on-line dissemination of the JACSM is possible for interested readers.

CONTENTS

ROBUST FLOW CONTROLLERS FOR A SINGLE VIRTUAL CIRCUIT IN DATA TRANSMISSION NETWORKS WITH LOSSY LINKS

Andrzej Bartoszewicz, Piotr Leśniewski

Department of Computer Science, University of Social Sciences,
9 Sienkiewicza St., 90-113 Łódź, Poland
(andrzej.bartoszewicz, piotr.lesniewski2)@gmail.com

Abstract

The paper concerns an application of regulation theory methods to modeling and effective control of connection-oriented data transmission networks. In particular the problem of congestion control in a single virtual circuit of such a network is considered and new discrete-time sliding mode data flow rate controllers are proposed. The controllers are designed in such a way that packet losses are explicitly accounted for. The closed-loop system stability and finite-time error convergence are proved. Moreover, a number of favorable properties of the proposed controllers are stated as theorems, formally proved and verified in a simulation example. It is demonstrated that the proposed controllers guarantee full utilization of the available bandwidth and eliminates the risk of bottleneck node buffer overflow. Application of time-varying sliding hyperplanes helps avoid excessive transmission rates at the beginning of the control process.

Key words: data transmission networks, congestion control, sliding-mode control, discrete-time systems

1 Introduction

The problem of congestion control in data transmission networks has recently become one of the most extensively studied research issues. Due to bandwidth variations, packet losses, round trip time uncertainty and users' constraints, the solution of the problem is not an easy task. On the other hand, the control theoretic approach [14, 19] to the congestion elimination offers many well developed tools and methods which can turn out to be very useful in the design of flow management strategies. Therefore, in this paper we introduce a discrete time model of a single virtual circuit in connection-oriented network and we apply sliding mode methodology [13–18, 20] to solve the congestion problem in the circuit.

The difficulty of the congestion control in modern data transmission networks is mainly caused by long propagation delays in the system. If congestion occurs at a specific node, information about this condition must be conveyed to all the sources transmitting data through that node, which involves feedback propagation delays. The congestion control in connection-oriented networks has recently been studied in several papers [1–12]. The control algorithms proposed in those papers employ a proportional plus derivative [10], stochastic [7], adaptive [9] and Smith predictor based control strategies [1], [2], [11], [12]. Recently a number of sliding mode congestion control algorithms have also been proposed [3–6]. However, not many results on congestion control in networks with lossy links are available. Therefore, this paper presents a sliding mode flow controller for a single connection which looses some packets during the transmission process. In other words, in this paper – on the contrary to the previously published results – we consider not only data losses caused by the bottleneck buffer link overflow, but also those which for other reasons happen on the transmission way from the source to the bottleneck link.

In the next section we introduce the state space model of the network, and then in section 3 we use this model to design a feasible sliding mode congestion control strategies.

2 Network Model

In this paper we consider a virtual circuit in a connection-oriented network which consists of a single data source, intermediate nodes and a destination. The block diagram of the circuit is shown in Figure 1. It is assumed that there is only one bottleneck node in the network. A controller which determines data transmission rate of the source is placed at the bottleneck node. The output signal of the controller (denoted by u) is sent back to the source, and reaches it after backward delay T_B. The source then sends the specified amount of data, which is passed from node to node until it reaches the bottleneck queue after forward delay T_F. We assume that somewhere along that line a known, fixed percentage of data packets are lost so that only αu (where $\alpha \in (0,1)$) data packets arrive at the bottleneck node. The round trip time RTT, i.e. the delay between generating a signal by the controller and the requested data arriving at the bottleneck queue, is a sum of the forward and backward propagation delays

$$RTT = T_B + T_F \tag{1}$$

Further in the paper, T represents the discretisation period, $x(kT)$ is the bottleneck queue length at time instant kT, and $x_d > 0$ is the demand value of $x(kT)$. It is assumed that before the start of data transmission, the buffer is empty, i.e.

$x(kT < 0) = 0$. We also assume that the round trip time is a multiple of the discretisation period, i.e. $RTT = m_{RTT}T$, where m_{RTT} is a positive integer.

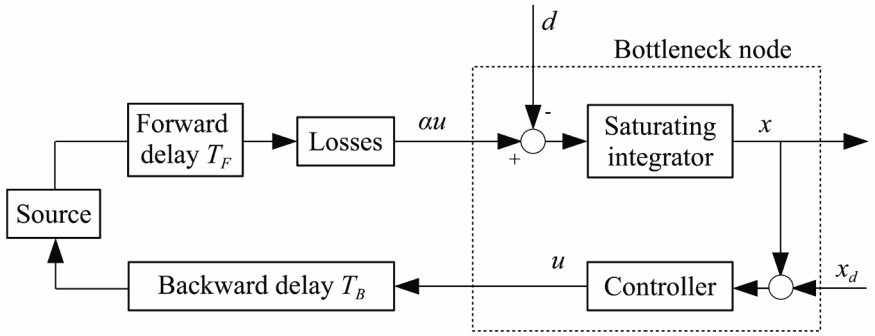

Figure 1. The network model

The controller output at time kT is denoted as $u(kT)$.The first data will reach the queue after RTT so for any time $kT \le RTT$ the queue length

$$x(kT) = 0 \tag{2}$$

The amount of data which may leave the bottleneck buffer is modeled as an *a priori* unknown bounded function of time $d(kT)$. The maximum value of $d(kT)$ is represented by d_{max}. The amount of data actually leaving the bottleneck node at time kT is denoted by $h(kT)$. For any $k \ge 0$

$$0 \le h(kT) \le d(kT) \le d_{max} \tag{3}$$

The queue length for $kT > RTT$ may be expressed as

$$x(kT) = \alpha \sum_{j=0}^{k-1} u(jT - RTT) - \sum_{j=0}^{k-1} h(jT) = \alpha \sum_{j=0}^{k-m_{RTT}-1} u(jT) - \sum_{j=0}^{k-1} h(jT) \tag{4}$$

and the network can be formulated in the state space in the following form

$$\begin{aligned} \boldsymbol{x}[(k+1)T] &= \boldsymbol{A}\boldsymbol{x}(kT) + \boldsymbol{b}u(kT) + \boldsymbol{o}h(kT) \\ y(kT) &= \boldsymbol{q}^T \boldsymbol{x}(kT) \end{aligned} \tag{5}$$

where $\boldsymbol{x}(kT) = [x_1(kT)\ x_2(kT)\ \ldots\ x_n(kT)]^T$ is the state vector, $y(kT) = x_1(kT)$ is the queue length, and $x_i(kT) = u[(k - n + i - 1)T]$ for any $i = 2, \ldots, n$. Furthermore, $\boldsymbol{A}$ is an n x n state matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 & \alpha & 0 & & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & & 0 \end{bmatrix} \tag{6}$$

$\boldsymbol{b}$, $\boldsymbol{o}$ and $\boldsymbol{q}$ denote $n \times 1$ vectors

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{o} = \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

and $n = m_{RTT} + 1$. The state space equation can also be rewritten as follows

$$\begin{cases} x_1\left[(k+1)T\right] = x_1(kT) + \alpha x_2(kT) - h(kT) \\ x_2\left[(k+1)T\right] = x_3(kT) \\ \vdots \\ x_{n-1}\left[(k+1)T\right] = x_n(kT) \\ x_n\left[(k+1)T\right] = u(kT) \end{cases} \tag{8}$$

with the output signal $y(kT) = x_1(kT)$. The desired state of the system is denoted by $\boldsymbol{x_d} = [x_{d1}\ \ x_{d2}\ \ \ldots\ \ x_{dn}]^T$. The first state variable x_{d1} is the demand queue length, and further in the paper it is represented by x_d. It can be noticed from equations (8) that for $h(kT) = 0$ all other components of the demand state vector are equal to zero.

3 Congestion Control Strategies

In this section the flow control problem for the described network is considered. In chapter 3.1 a chattering-free discrete-time sliding mode controller is designed that guarantees finite-time error convergence to zero. Important properties of the proposed control strategy are then formulated and proved. Since the strategy proposed in chapter 3.1 may lead to large values of control signal in the starting phase of the control process, in chapter 3.2 a time-varying sliding hyperplane is introduced that minimizes this effect. Then important properties of the modified control strategy are also formulated and proved.

3.1 Time-Invariant Sliding Hyperplane

For the sliding mode controller design purpose we neglect the disturbance $h(kT)$ and introduce a sliding hyperplane described by the following equation

$$s(kT) = \boldsymbol{c}^T \boldsymbol{e}(kT) = 0 \tag{9}$$

where vector $\boldsymbol{c}^T = [c_1 \;\; c_2 \;\; \ldots \;\; c_n]$ satisfies $\boldsymbol{c}^T\boldsymbol{b} \neq 0$. Error of the closed loop system is denoted by $\boldsymbol{e}(kT) = \boldsymbol{x}_d - \boldsymbol{x}(kT)$. Substituting (5) into $\boldsymbol{c}^T\boldsymbol{e}[(k+1)T] = 0$ we obtain the following control law

$$u(kT) = \left(\boldsymbol{c}^T\boldsymbol{b}\right)^{-1} \boldsymbol{c}^T \left[\boldsymbol{x}_d - \boldsymbol{A}\boldsymbol{x}(kT)\right] \tag{10}$$

When this control signal is used, the closed-loop system state matrix has the form $\boldsymbol{A}_c = [\boldsymbol{1} - \boldsymbol{b}(\boldsymbol{c}^T\boldsymbol{b})^{-1}\boldsymbol{c}^T]\boldsymbol{A}$. The characteristic polynomial of this matrix

$$det\left(z\boldsymbol{I}_n - \boldsymbol{A}_c\right) = z^n + \frac{c_{n-1} - c_n}{c_n} z^{n-1} + \ldots + \frac{\alpha c_1 - c_2}{c_n} z \tag{11}$$

which gives the condition $c_n \neq 0$. A discrete-time system is asymptotically stable if and only if all of its eigenvalues are located inside the unit circle. Furthermore, to ensure finite-time error convergence to zero the characteristic polynomial (11) has to satisfy

$$det\left(z\boldsymbol{I}_n - \boldsymbol{A}_c\right) = z^n \tag{12}$$

Comparing the coefficients of (11) and (12) we find the following form of vector $\boldsymbol{c}$

$$\boldsymbol{c}^T = \left[\, 1/\alpha \quad 1 \quad \cdots \quad 1 \,\right] c_n \tag{13}$$

Using (6), (7) and (13) we can rewrite (10) as follows

$$u(kT) = \frac{1}{\alpha}\left[x_d - x(kT)\right] - \sum_{i=2}^{n} x_i(kT) \tag{14}$$

From (8) we notice that all the state variables except x_1 are the delayed values of the control signal, i.e. for $i = 2, \ldots, n$

$$x_i(kT) = u\left[(k - n + i - 1)T\right] \tag{15}$$

Substituting (15) into (14) we get

$$u(kT)=\frac{1}{\alpha}\left[x_d-x(kT)\right]-\sum_{i=k-m_{RTT}}^{k-1}u(iT) \tag{16}$$

This completes the design of a flow control algorithm with a time-invariant sliding hyperplane.

Properties of the Proposed Strategy

In the previous section a time-invariant sliding hyperplane has been designed to guarantee stability and finite-time error convergence of the closed-loop system. The amount of data to be sent is given by (16). Consequently

$$u(0)=\frac{x_d}{\alpha} \tag{17}$$

Lemma 1: If the designed sliding mode controller is applied, then its output for any $k \geq 0$ satisfies

$$u(kT)=\frac{1}{\alpha}h\left[(k-1)T\right] \tag{18}$$

Proof: Substituting (4) into (16) we obtain

$$\begin{aligned} u(kT)&=\frac{1}{\alpha}\left[x_d-\alpha\sum_{j=0}^{k-m_{RTT}-1}u(jT)+\sum_{j=0}^{k-1}h(jT)\right]-\sum_{j=k-m_{RTT}}^{k-1}u(jT)\\ &=\frac{1}{\alpha}\left[x_d+\sum_{j=0}^{k-1}h(jT)\right]-\sum_{j=0}^{k-1}u(jT)\end{aligned} \tag{19}$$

By mathematical induction: first we check if (18) holds for $k = 1$

$$u(T)=\frac{1}{\alpha}\left[x_d+\sum_{j=0}^{0}h(jT)\right]-\sum_{j=0}^{0}u(jT)=\frac{1}{\alpha}\left[x_d+h(0)\right]-\frac{x_d}{\alpha}=\frac{1}{\alpha}h(0) \tag{20}$$

Now we assume that (18) holds for some $k = m$, where m is a positive integer, i.e.

$$u(mT)=\frac{1}{\alpha}h\left[(m-1)T\right] \tag{21}$$

Then using this assumption we can find from (19) that for $k = m + 1$

$$
\begin{aligned}
u\left[(m+1)T\right] &= \frac{1}{\alpha}\left[x_d + \sum_{j=0}^{m} h(jT)\right] - \sum_{j=0}^{m} u(jT) \\
&= \frac{1}{\alpha}\left[x_d + \sum_{j=0}^{m} h(jT)\right] - \frac{x_d}{\alpha} - \sum_{j=1}^{m} u(jT) \\
&= \frac{1}{\alpha}\sum_{j=0}^{m} h(jT) - \frac{1}{\alpha}\sum_{j=0}^{m-1} h(jT) = \frac{1}{\alpha} h(mT)
\end{aligned}
\tag{22}
$$

which means that if (18) holds for $k = m$, then it also holds for $k = m + 1$.

Finally, taking into account (21) and (22) we can conclude that (18) indeed holds for any integer $k \geq 0$. This ends the proof.

Lemma 1 clearly shows that the output of the proposed controller is always nonnegative and bounded, i.e. for any $k \geq 1$

$$0 \leq u(kT) \leq \frac{1}{\alpha} d_{max} \tag{23}$$

Theorem 1: If the proposed strategy is used, then the queue length will never exceed its demand value, i.e. for any $k \geq 0$

$$x(kT) \leq x_d \tag{24}$$

Proof: From (2) for any $k < (m_{RTT} + 1)$ the queue length $x(kT) = 0$. Hence to prove the theorem we only need to check if (24) holds for $k \geq m_{RTT} + 1$. Using (18) we can rewrite (4) as

$$x(kT) = x_d - \sum_{j=k-m_{RTT}-1}^{k-1} h(jT) \leq x_d \tag{25}$$

This ends the proof.

From the first equation in set (8) we notice that if $x[(k+1)T]$ is greater than zero, then the available bandwidth $d(kT)$ is fully used. Theorem 2 gives the necessary condition to guarantee that the queue length is strictly positive.

Theorem 2: If the proposed strategy is used, and the demand queue length satisfies

$$x_d > (m_{RTT} + 1) d_{max} \tag{26}$$

then for any $k > m_{RTT}$ the queue length is always strictly positive.

Proof: From (3) we see that for any $k \geq 0$ the consumed bandwidth is always upper bounded $h(kT) \leq d_{max}$. Using (25) for $k > m_{RTT}$, we obtain

$$x(kT) = x_d - \sum_{j=k-m_{RTT}-1}^{k-1} h(jT) \geq x_d - (m_{RTT}+1)d_{max} > 0 \tag{27}$$

This ends the proof.

Theorem 2 shows that for any $k > m_{RTT}$ the queue length is strictly greater than zero, which implies that the available bandwidth is fully used for any $k \geq m_{RTT}$.

3.2 Time-Varying Sliding Hyperplane

A disadvantage of the control strategy proposed in chapter 3.1 is a large value of the control signal at the first time instant. Therefore, in this subsection we introduce a time-varying hyperplane that reduces this effect. The properties of this modified strategy are then formulated and proved.

We replace equation (9) describing the sliding hyperplane with the following one

$$s(kT) = c^T e(kT) + f(kT) = 0 \tag{28}$$

where $f(kT)$ is an *a priori* known function of time chosen to satisfy $s(0) = 0$ (the representative point at time instant $k = 0$ is positioned on the sliding hyperplane). This gives the following condition

$$f(0) = -c^T e(0) \tag{29}$$

Because the previously proposed controller exhibits very good dynamic performance after the starting phase of the regulation process, there should exist such a k_0 that $f(kT) = 0$ for any $k > k_0$. Furthermore function $f(kT)$ should be strictly monotonic in the time interval $[0, k_0]$ .With the use of such a function the sliding hyperplane moves monotonically towards the origin of the coordinate frame, intersects it after k_0, and remains fixed for any $k > k_0$.

We chose $f(kT)$ to be linear in the interval $[0, k_0]$. Thus it can be written as follows

$$f(kT) = \begin{cases} \dfrac{k-k_0}{k_0} \boldsymbol{c}^T \boldsymbol{e}(0) & \text{for} \quad k \leq k_0 \\ 0 & \text{for} \quad k > k_0 \end{cases} \tag{30}$$

Now substituting $\boldsymbol{e}(kT) = \boldsymbol{x_d} - \boldsymbol{x}(kT)$ into $s[(k+1)T] = 0$ we obtain

$$u(kT) = \left(\boldsymbol{c}^T \boldsymbol{b}\right)^{-1} \left\{\boldsymbol{c}^T \left[\boldsymbol{x}_d - \boldsymbol{A}\boldsymbol{x}(kT)\right] + f\left[(k+1)T\right]\right\} \tag{31}$$

where vector $\boldsymbol{c}$ is given by (13) in order to maintain the desirable properties of the previous control strategy for $k > k_0$.

Using (6), (7), (13) and (15) we rewrite (31) as follows

$$\begin{aligned} u(kT) &= \frac{1}{\alpha}\left[x_d - x_1(kT)\right] - \sum_{i=2}^{n} x_i(kT) + \frac{1}{c_n} f\left[(k+1)T\right] \\ &= \frac{1}{\alpha}\left[x_d - x_1(kT)\right] - \sum_{j=k-m_{RTT}}^{k-1} u(jT) + \frac{1}{c_n} f\left[(k+1)T\right] \end{aligned} \tag{32}$$

This completes the design of a flow control algorithm with the proposed time-varying sliding hyperplane.

Properties of the Proposed Strategy

In the previous subsection we modified the strategy proposed in chapter 3.1, introducing a time-varying hyperplane. The goal of this modification is to reduce the control signal in the starting phase of the data transmission process. In this section, we formulate and prove the properties of this altered algorithm. Lemma 2 shows that the control signal is nonnegative and upper bounded. Theorems 3 and 4 (analogous to Theorems 1 and 2) show that the queue length will not exceed its demand value and that after some initial time the queue length will always be strictly positive, which implies that the available bandwidth will be fully used.

Lemma 2: If the designed sliding mode controller is applied, then its output for any $k \geq 0$ satisfies

$$u(kT) = \frac{1}{\alpha} h\left[(k-1)T\right] + \frac{1}{c_n}\left\{f\left[(k+1)T\right] - f(kT)\right\} \tag{33}$$

Proof: Substituting (4) into (32)

$$u(kT) = \frac{1}{\alpha}\left[x_d + \sum_{j=0}^{k-1} h(jT)\right] - \sum_{j=0}^{k-1} u(jT) + \frac{1}{c_n} f\left[(k+1)T\right] \tag{34}$$

By mathematical induction, first we check if (33) holds for $k = 0$

$$
\begin{aligned}
u(0) &= \frac{1}{\alpha}\left[x_d + \sum_{j=0}^{-1} h(jT)\right] - \sum_{j=0}^{-1} u(jT) + \frac{1}{c_n} f(T) \\
&= \frac{1}{\alpha} x_d + \frac{1}{c_n} f(T) = \frac{1}{c_n}\left[f(T) - f(0)\right] = \frac{1}{\alpha} h(-T) + \frac{1}{c_n}\left[f(T) - f(0)\right]
\end{aligned}
\tag{35}
$$

Then we assume that (33) holds for $k = m$

$$
u(mT) = \frac{1}{\alpha} h\left[(m-1)T\right] + \frac{1}{c_n}\left\{f\left[(m+1)T\right] - f(mT)\right\} \tag{36}
$$

Using this assumption from (32) we can find, that for $k = m + 1$

$$
\begin{aligned}
u\left[(m+1)T\right] &= \frac{1}{\alpha}\left[x_d + \sum_{j=0}^{m} h(jT)\right] + \frac{f\left[(m+2)T\right]}{c_n} - \sum_{j=0}^{m} u(jT) \\
&= \frac{x_d}{\alpha} + \frac{1}{\alpha}\sum_{j=0}^{m} h(jT) + \frac{1}{c_n} f\left[(m+2)T\right] - u(0) \\
&- \sum_{j=1}^{m}\left\{\frac{1}{\alpha} h\left[(j-1)T\right] + \frac{1}{c_n}\left[f\left((j+1)T\right) - f(jT)\right]\right\} \\
&= \frac{x_d}{\alpha} + \frac{1}{\alpha}\sum_{j=0}^{m} h(jT) + \frac{1}{c_n} f\left[(m+2)T\right] - \frac{x_d}{\alpha}\frac{1}{k_0} \\
&- \frac{1}{\alpha}\sum_{j=0}^{m-1} h(jT) - \left\{\frac{1}{c_n} f\left[(m+1)T\right] - \frac{1}{c_n} f(T)\right\} \\
&= \frac{1}{\alpha} h(mT) + \frac{1}{c_n}\left\{f\left[(m+2)T\right] - f\left[(m+1)T\right]\right\}
\end{aligned}
\tag{37}
$$

which means that if (32) holds for $k = m$, then it also holds for $k = m + 1$.
Taking into account (35) and (37), we conclude that equation (33) actually holds for any $k \geq 0$. This ends the proof.

It is easy to notice, that because $\max\{[f((k+1)T) - f(kT)]/c_n\} = x_d/\alpha k_0$ and $h(kT) \leq d_{max}$ for any $k \geq 0$, then $u(kT) \leq (x_d/\alpha k_0) + d_{max}$ for any $k \geq 0$. Moreover, as $\min\{[f((k+1)T) - f(kT)]/c_n\} = 0$ and $h(kT) \geq 0$ for any $k \geq 0$ then $u(kT) \geq 0$ for any $k \geq 0$. This shows that the designed controller determines data transmission rate which is always nonnegative and upper-bounded. Furthermore, choosing $f(kT)$ to be linear in the interval $[0, k_0]$ we obtained a constant upper bound of the control signal, which is quite practical from application point of view.

Theorem 3: If the proposed controller is applied, then the queue length will never exceed its demand value, i.e. for any $k \geq 0$

$$x(kT) \leq x_d \tag{38}$$

Proof: Transforming (32) we obtain

$$\frac{1}{\alpha}\left[x_d - x(kT)\right] = u(kT) + \sum_{j=k-m_{RTT}}^{k-1} u(jT) - \frac{1}{c_n} f\left[(k+1)T\right] \tag{39}$$

From the second Lemma $u(kT) \geq 0$ for any $k \geq 0$, and from (30) $f[(k+1)T]/c_n \leq 0$ also for any $k \geq 0$. From this follows that the right hand side of (39) is nonnegative, which gives $x_d - x(kT) \geq 0$. This ends the proof.

Theorem 4: If the proposed control strategy is applied and the demand queue length satisfies inequality

$$x_d > (m_{RTT} + 1) d_{max} \tag{40}$$

then the queue length is strictly greater than zero for any $k > k_0 + m_{RTT}$.
Proof: Using Lemma 2 we can rewrite (4) as follows

$$\begin{aligned} x(kT) &= -\sum_{j=0}^{k-1} h(jT) + \alpha u(0) \\ &+ \alpha \sum_{j=1}^{k-m_{RTT}-1} \left\{ \frac{1}{\alpha} h\left[(j-1)T\right] + \frac{1}{c_n}\left[f\left((j+1)T\right) - f(jT)\right]\right\} \\ &= \frac{\alpha}{c_n}\left\{ f\left[(k-m_{RTT})T\right] - f(T)\right\} + \sum_{j=0}^{k-m_{RTT}-2} h(jT) - \sum_{j=0}^{k-1} h(jT) + \frac{1}{k_0} x_d \\ &= \frac{1}{k_0} x_d + \frac{\alpha}{c_n}\left\{ f\left[(k-m_{RTT})T\right] - f(T)\right\} - \sum_{j=k-m_{RTT}-1}^{k-1} h(jT) \end{aligned} \tag{41}$$

Then using (30), for any $k > k_0 + m_{RTT}$ from (41) we obtain

$$\begin{aligned} x(kT) &= \frac{1}{k_0} x_d + \frac{\alpha}{c_n}\left\{ f\left[(k-m_{RTT})T\right] - f(T)\right\} - \sum_{j=k-m_{RTT}-1}^{k-1} h(jT) \\ &= \frac{1}{k_0} x_d - \frac{1-k_0}{k_0} x_d - \sum_{j=k-m_{RTT}-1}^{k-1} h(jT) \geq x_d - (m_{RTT}+1) d_{max} > 0 \end{aligned} \tag{42}$$

This shows that the queue length is indeed strictly greater than zero for any $k > k_0 + m_{RTT}$.

4 Simulation Example

In order to verify the properties of both proposed strategies computer simulations of the network are performed. The discretisation period T is selected as 1 ms. The round trip time RTT is assumed to be 9 ms. Therefore $m_{RTT} = 9$, and $n = 10$. The maximum available bandwidth of the bottleneck node is $d_{max} = 80$ kb. According to Theorems 2 and 4, the demand queue length x_d in both control algorithms should be greater than 800 kb. Therefore x_d has been chosen as 810 kb. In the presented simulation example, coefficient $\alpha = 0.97$, which means that 97% of the data sent by the source arrive at the bottleneck node. For the time-varying hyperplane, parameter k_0 was chosen equal to 7.

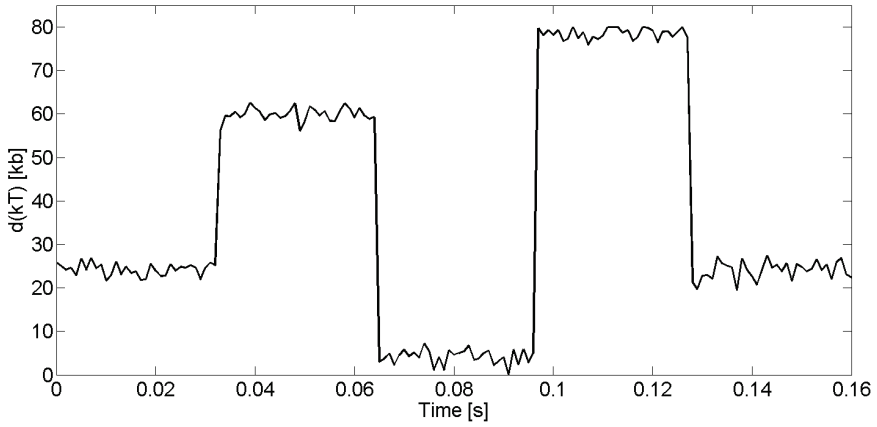

Figure 2. Available bandwidth

The available bandwidth is shown in Figure 2. It changes rapidly between small and large values, which reflects the most adverse possible conditions, that could exist in the network. Figure 3 shows the output signal of controller (16). It can be seen from this figure that the control signal is always strictly positive and upper bounded. Then, Figure 4 shows the bottleneck link queue length for the same control strategy. We can observe, that the queue length never exceeds its demand value, and is strictly positive for any $k > m_{RTT}$. This implies that the proposed strategy eliminates the risk of buffer overflow and ensures full bandwidth utilization.

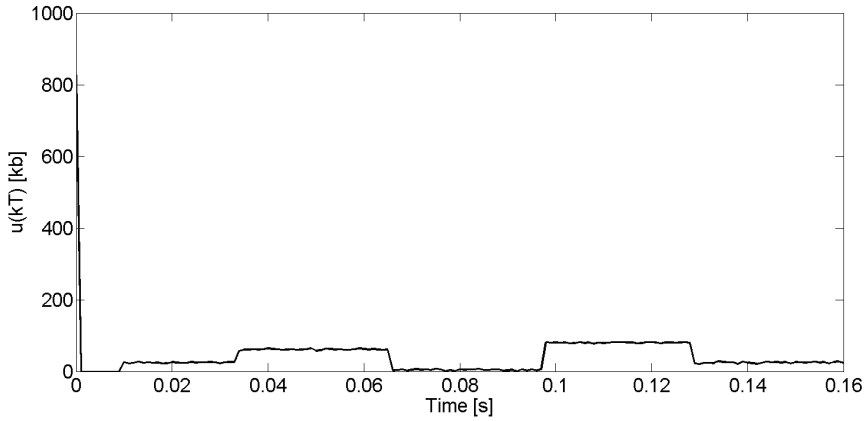

Figure 3. Output signal of the controller with the time-invariant sliding hyperplane

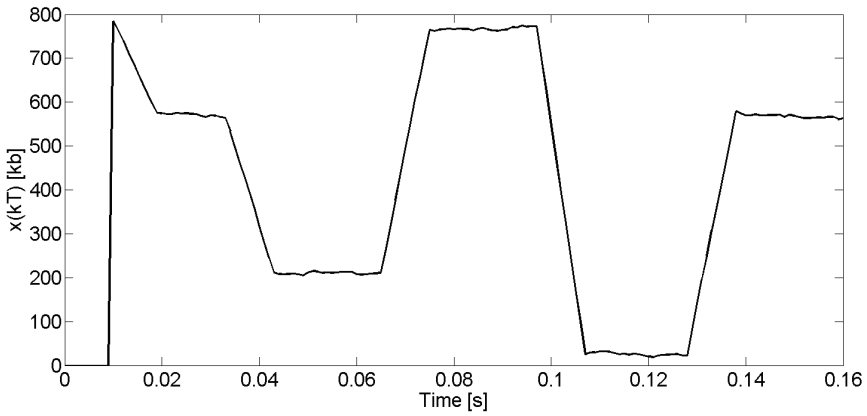

Figure 4. Queue length with the application of the controller with the time-invariant sliding hyperplane

Figures 5 and 6 show the respective simulation results for the network controlled according to strategy (32). Comparing figures 3 and 5 we notice that the introduction of a time-varying sliding hyperplane significantly reduces the maximum value of the control signal in the starting phase of the control process. Furthermore, as can be seen from figure 6, all the advantages of the previous controller with the time-invariant sliding hyperplane are maintained.

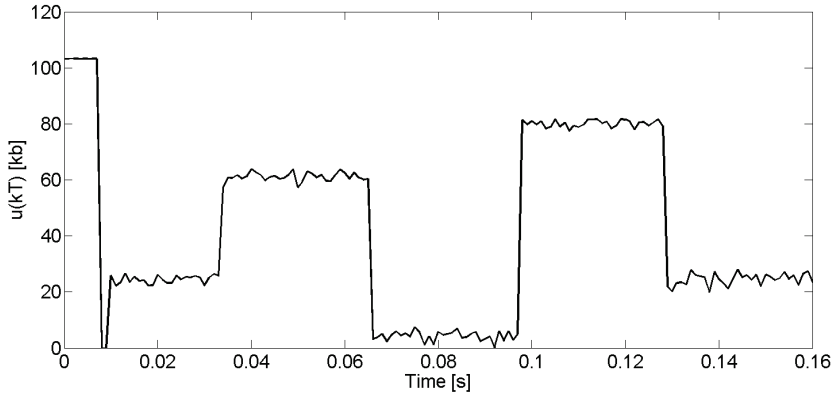

Figure 5. Output signal of the controller with the time-varying sliding hyperplane

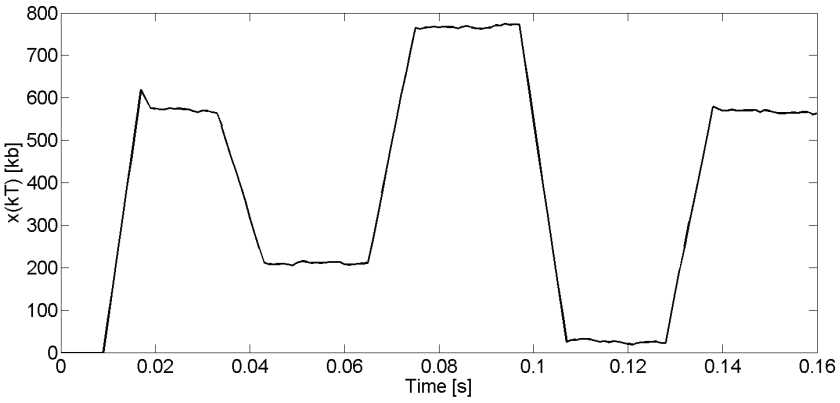

Figure 6. Queue length with the application of the controller with the time-varying sliding hyperplane

5 Conclusions

In this paper two sliding mode control strategies for a single virtual connection in a network with lossy links have been presented. The first strategy, which uses a time-invariant sliding hyperplane, has been designed to ensure closed-loop system stability and finite time error convergence. Then it has been modified by introducing a time-varying sliding hyperplane in order to reduce the maximum value of the control signal in the starting phase of data transmission. Flow rates generated by both strategies are proved to be always non-negative and upper bounded. Moreover, both control algorithms eliminate the risk of buffer overflow and for each of the algorithms conditions that guarantee full bottleneck link bandwidth consumption have been derived.

Acknowledgments

This work has been performed in the framework of a project "Optimal sliding mode control of time delay systems" financed by the National Science Centre of Poland – decision number DEC 2011/01/B/ST7/02582.

References

1. A. Bartoszewicz and T. Molik, 2004, "*ABR traffic control over multi-source single-bottleneck ATM networks*," Journal of Applied Mathematics and Computer Science., vol. 14, no. 1, pp. 43-51.
2. A. Bartoszewicz, 2006, "*Nonlinear flow control strategies for connection-oriented communication networks*," IEE Proceedings – Control Theory and Applications, vol. 153, no. 1, pp. 21-28.
3. A. Bartoszewicz, J. Żuk, 2009, "Discrete-time sliding mode flow controller for multi-source connection-oriented communication networks," Journal of Vibration and Control, vol. 15, no. 11, pp. 1745-1760.
4. P. Ignaciuk, A. Bartoszewicz, 2008, "Linear quadratic optimal discrete time sliding mode controller for connection oriented communication networks," IEEE Transactions on Industrial Electronics, vol. 55, no. 11, pp. 4013–4021.
5. P. Ignaciuk, A. Bartoszewicz, 2009, "*Linear quadratic optimal sliding mode flow control for connection-oriented communication networks*," International Journal of Robust and Nonlinear Control, vol. 19, no. 4, pp. 442–461.
6. P. Ignaciuk, A. Bartoszewicz, 2011, "*Discrete-time sliding-mode congestion control in multi-source communication networks with time-varying delay*," IEEE Transactions on Control Systems Technology, vol. 19, no. 4, pp. 852–867.
7. O. C. Imer, S. Compans, T. Basar, R. Srikant, 2001, "*Available bit rate congestion control in ATM networks*," IEEE Control Systems Magazine, vol. 21, no. 1, pp. 38-56,.
8. R. Jain, 1996, "Congestion control and traffic management in ATM networks: recent advances and a survey," Computer Networks ISDN Syst., vol. 28, no. 13, pp. 1723-1738.
9. K. P. Laberteaux, Ch. E. Rohrs, P. J. Antsaklis, 2002, "*A practical controller for explicit rate congestion control*," IEEE Transactions on Automatic Control, vol. 47, no. 6, pp. 960-978.
10. I. Lengliz, F. Kamoun, 2000, "*A rate-based flow control method for ABR service in ATM networks*," Computer Networks, vol. 34, no. 1, pp. 129-138.
11. S. Mascolo, 1999, "Congestion control in high-speed communication networks using the Smith principle," Automatica, vol. 35, no. 12, pp. 1921-1935.

12. S. Mascolo, 2000, "*Smith's principle for congestion control in high-speed data networks*," IEEE Transactions on Automatic Control, vol.45, no. 2, pp. 358-364.
13. C. Milosavljević, B. Peruničić-Draženović, B. Veselić, D. Mitić, 2006, "*Sampled data quasi-sliding mode control strategies*," IEEE International Conference on Industrial Technology", pp. 2640-2645.
14. J. Slotine, W. Li, 1991, Applied Nonlinear Control. Prentice–Hall, Englewood Cliffs, NJ.
15. S. Tokat, I. Eksin, M. Guzelkaya, M. Soylemez, 2003, "*Design of a sliding mode controller with a nonlinear time-varying sliding surface*". Transactions of the Institute of Measurement and Control vol. 25, pp. 145-162.
16. V. Utkin, 1977, "*Variable structure systems with sliding modes,*" IEEE Transactions on Automatic Control, vol. 22, pp. 212-222.
17. V. Utkin, S. Drakunow, 1989, "*On discrete-time sliding mode control*," IFAC Conference on Nonlinear Control, pp. 484-489.
18. V. Utkin, 1992, Sliding Modes in Control and Optimization, Springer-Verlag, Berlin.
19. M. Vidyasagar, 1993, Nonlinear Systems Analysis, Prentice-Hall International, Englewood Cliffs.
20. K. Young, V. Utkin, Ü. Özgüner, 1999, "*A control engineer's guide to sliding-mode control*," IEEE Transactions on Control Systems Technology, vol. 7, pp. 328-342.

PREREQUISITES FOR EFFECTIVE REQUIREMENTS MANAGEMENT

Konrad Grzanek

IT Institute, Academy of Management, Łódź, Poland
kgrzanek@swspiz.pl, kongra@gmail.com

Abstract

Despite an undeniable progress in the whole software creation process, software development is still more art than science. The requirements analysis is a highly critical step in the software life-cycle. Requirement managements errors are the most common errors in the software projects. The proper and effective requirements management saves the overall project costs. The key motivation behind this work was opening a way of finding approaches to managing the requirements appearing in such large software projects as compilers for various programming languages. This paper is an introduction to a full presentation of requirements management solution in which the requirements and implementation information is placed directly in the source code. We concentrate on describing a context in which the requirements management process takes place, trying to present the most interesting existing solutions, indicating the problems and opening a discussion on what ways to follow in the future scientific research.

Key words: requirements engineering, requirements abstraction, functional programming

1 The State of the Art in Requirements Management

According to the commercial surveys conduced back in the ’90s an average US software project overran its budgeted time by 190%, it’s budgeted costs by 222%, and delivered only 60% of the planned functionality. Only 16% of projects were delivered at the estimated time and cost, and 31% of projects were canceled before delivery, with larger companies performing much worse than smaller ones (source: [1]). Martyn Thomas also mentions:

> „A UK survey, published in the 2001 Annual Review of the British Computer Society showed a similar picture. Of more than 500 development projects, only three met the survey's criteria for success. In 2002, the annual cost of poor quality software to the US economy was estimated at $60B."

Despite an undeniable progress in the whole software creation process, software development is still more art than science (after [3]). Most researchers point out the following causes of software process failures [3]:

- Poor requirements management. We forge ahead with development lacking user input and without a clear understanding of the problem we are attempting to solve.
- Poor change management. Changes to requirements and other development products are inevitable; yet we rarely track them or understand their impact.
- Poor quality control. We have poor measures for system quality, little knowledge of processes that affect quality, and no feedback to modify the process after witnessing the effects of a particular development strategy.
- Little control of schedules and costs. Accurate planning is the exception while unrealistic expectations are the norm.

It is a fact universally acknowledged in the software engineering world that requirements analysis is a highly critical step in the software life-cycle [2]. The lack of the ability to specify, control and manage the software project requirements causes the loss of control over the overall system behavior, it's design and quality [3]. The proper and effective requirements management saves the overall project costs due to the following reasons (as stated in [3]):

- Requirement errors typically cost well over 10 times more to repair than other errors.
- Requirement errors typically comprise over 40% of all errors in a software project.
- Small reductions in the number of requirement errors pay big dividends in avoided rework costs and schedule delays.

Moreover, the requirement managements errors are the most common errors in the software projects. No wonder the issue is seen as one of the fundamental issues in the field both by scientific researches as well as organizations like Software Engineering Institute (SEI) with their Capability Maturity Model. In CMM the requirements management is one of the first steps to achieving process maturity and the key area that must be addressed to move from Level 1 to Level 2 [3].

We define a requirement as a capability or feature needed by a user to solve a problem or achieve an objective. There are two major kinds of re-

quirements, the functional and the non-functional ones (abbreviated NFR). Some management approaches try to treat these two categories uniformly, but often they are treated separately due to their apparent differences in nature (e. g. [4]) - the functional requirements specify each function that a system must be capable of performing, whereas the NFRs specify how the system is going to be implemented to achieve it's goals and what it's quality attributes will be.

The requirements engineering as defined in [5] and [6] is:

> „the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

This discipline inevitably breaks the borders of multiple system views, because – as stated in [6] - software cannot function in isolation from the environment in which it operates and in which it is embedded. So in fact the requirements engineering may be treated as a branch of systems engineering. The paper [6] also states that requirements may and should undergo formal treatment, i. e. the formal description and reasoning. Present article addresses the problem of the formal describing and managing the requirements in a persistent way.

Works by Zave and Jackson expose similar conclusions related to the multidisciplinary character of requirements engineering. In [7] the authors put a particular emphasis on the impact of the environment onto the requirements engineering process. They state that the descriptions of the requirements should be in fact the environment's descriptions. Another problem they try to address are the implementation bias while defining the requirements (especially in the early stages/high abstraction layers) and the role of knowledge management in the whole process. Some design and implementation achievements presented in the present paper loosely refer to the knowledge management area.

Hofmann and Lehner [8] underline the unquestionable importance of possessing a deep domain knowledge on increasing the probability of software project success. Consequently the requirements engineering is the key factor here, together with the experts' knowledge as well as the stakeholders' competence. According to this the requirements engineering is a multidisciplinary and highly competence-demanding field.

This multidisciplinary character refers also to the possible applications for the requirements management. The discipline is by no means limited to the software domain. The paper [9] describes a requirements management framework that enables health information custodians (HIC) to document and track compliance with privacy legislation as the legislation and hospital business

processes evolve. An interesting graphical notation called the User Requirements Notation (URN) is given there together with it's major complementary notations, namely the Goal-oriented Requirements Language and Use Case Maps.

Requirements representation is very important with respect to the potential algorithmic processing. For example [11] gives a broad and systematic review of existing literature works that transform textually represented requirements into analysis model. The major reason is rooted in model transformation being one of the basic principles of Model Driven Architecture. According to the paper building a software system consists of a sequence of transformations, starting from requirements and ending with implementation. The problem of textual representations and processing of requirements will also be addressed further in our works.

In [10] there are mentioned the relations between requirements and high-level testing methodology called Abstract Testing. The article states that „[...] often a one-to-one correspondence between abstract test cases (resp. verification scenarios) and requirements can be achieved, which links abstract testing much more closely to the requirements and facilitates construction and maintenance of abstract test cases". The interesting feature of this methodology as a whole is an existence of a close relationship of the verification scenarios (and so – indirectly – the requirements) to the source code by the fact of an automatic checking the scenarios against the sources performed by a source model checker.

The requirements modeling with a combination of SysML and UML are described in [12]. The work is of a special significance because it presents the problem in the context of real-time systems specification. A classification of user requirements is also proposed there.

2 The Idea of Source Code as the Requirements Database

Now we should take a look at the existing approaches and tools automating the requirements management. According to [16] most requirements management tools perform the same core functions:

- They allow the system developer to import large documents from a variety of standard word processing formats.
- These documents can be split up into separately managed document elements.
- The document elements are subject to a rigorous change and version control regime.
- Relations can be established between document elements and attributes can be associated with the document elements and often the relations.

- A variety of document views can be generated using both attributes and relations, generally specific traceability views such as traceability matrices.
- Document templates can be set up and used to create new composite documents.
- Scripting or query languages provide support for the retrieval of information and the development of project specific views.
- Simple checks to ensure structural integrity of documents may be performed.

An apparent importance of attribution and labeling shows up. In fact, the labels are the core of our planned approach. The tool we start working on will also possess an effective querying mechanisms.

When referring to the architecture of requirements management tools [16] states that these tools have much in common: „They are generally based on a document repository, which may either be hosted on top of an industry standard database (relational or object-oriented) or a specifically crafted file store [...]. Most tools provide some simple control for multi-party editing of documents, the granularity of this control is dependent upon the underlying repository. At the front end, the tools generally appear similar to standard document processors. From a user interface standpoint, they provide a number of tools to support work with large hierarchical documents including the ability to work seamlessly in different document views."

[16] notices a very important weakness of these tools; they are in general process-free. The obvious yet not fully realized yet solution is „to integrate requirements management tools with a work-flow or process engine. Despite this being an obvious answer it is not very straightforward to achieve." One possible solution would make a step towards integrating programmers' personal information management with requirements management and their basic professional activity: working on the source code of implemented systems.

More features of an ideal requirements management system are presented in [17]. They extend the list of previously mentioned desired features with such elements as:

- Using effective information models
- Supporting various views of the same data
- Handling formal change requests for the requirements
- Keeping the history of requirements change
- Allowing base-lining
- Effective tool integration

and many more.

The key motivation behind this work was finding a way to manage the requirements appearing in such large software projects as compilers for various

programming languages. The Java 6 Language Specification is over 600-pages document containing lots of facts about the Java language run-time and compiler. Our goal is to deal with all those facts in an organized way. There are the following conclusions related to our situation:

1. There are huge amounts of facts in such a system, expected number reaches thousands of facts.
2. The facts are distributed, spread around many chapters of the source document. It closely resembles real-life situations that may appear in software projects of different nature.
3. Some information may be ambiguous and their transformation into the requires an active support from the programmer/designer.
4. The modules of the system described by these facts will be implemented by single developers who must have a clear view of what is to be done.
5. The requirements management system should not only help the programmer to organize these large volumes of information, but should also give him some help during the process of organizing facts.

We decided to use a unique approach of integrating the requirements management with source code. This approach is inspired by a ***homo-iconicity*** of the languages from the Lisp family of programming languages. Our solution is an embedded domain-specific language based on Clojure [18]. This DSL wins the following for the analysts, designers and programmers:

- Editing source code is a primary activity every programmer undertakes on every work-day. Putting the act of reading/writing the requirements into source code increases the comfort of this – sometimes boring – activity.
- It also affects the designers and other people not involved directly in the implementation phase, because it opens an effective channel of communication between – for instance – a system analyst and a coder; the analyst writes a requirement directly in a compilation unit, the programmer reads it and perform further steps to gain the required functionality.
- The presence of requirements in compilation units allows to interweave the them (their definitions formally speaking) with source code snippets being their direct implementations or implementation parts. This point is especially important because an act of locating requirements in pure (not instrumented with requirements or requirement-related tags) source code is a tedious and hard to solve problem. Further works on this can be found in [13, 14][1].
- A compilation unit keeping some requirements may be tracked and managed by a source management and revision control system, such as Git [19]. An immediate consequence is the ability to manage the requirements

1 Very interesting works related to real-time systems programming, source-code verification and requirements management in Ada programming language were presented in [15].

versions, because a requirement change is a change in the compilation unit. All version control system's goodies, including the possible encryption and the overall robustness of a distributed versioning system are there to be used.

3 Summary

The key motivation behind this work was opening a way of finding approaches to managing the requirements appearing in such large software projects as compilers for various programming languages. This paper may be treated as an introduction to a full presentation of requirements management solution in which the requirements and implementation information is placed directly in the source code. We concentrated on describing a context in which the requirements management process takes place, trying to present the most interesting existing solutions, indicating the problems and opening a discussion on what ways to follow in the future scientific research.

The main motivation for the conceptual creation and implementation of an innovative requirements management system is the urge to control the complexity (especially the non-accidental one) of large software projects, such as the implementation of a static analyzer of a formally described programming language or large modeling environments, such as the environments in which some biological structures and behaviors could be modeled (e. g. human immune system). The results of applying the requirements management system in future will be presented in future papers.

References

1. Thomas M., 2003, *The Modest Software Engineer*, Proceedings ISADS 2003, IEEE Press, pp. 169-174
2. Dardenne A., van Lamsweerde A., Fickas S., 1993, *Goal-directed Requirements Acquisition*, Science of Computer Programming, Vol. 20, pp. 3-50
3. Davis A. M., Leffingwell D. A., 1995, *Using Requirements Management to Delivery of Higher Quality Applications*, Rational Software Corporation
4. Ebert Ch., 1997, *Dealing with nonfunctional requirements in large software systems*, Annals of Software Engineering 3 (1997), pp. 367-395
5. Zave P., 1997, *Classification of Research Efforts in Requirements Engineering*, ACM Computing Surveys, 29(4), pp. 315-321
6. Nuseibeh B., Easterbrook S., 2000, *Requirements engineering: a roadmap*, ICSE '00 Proceedings of the Conference on The Future of Software Engineering, pp. 35-46

7. Zave P., Jackson M., *Four dark corners of requirements engineering*,1997, ACM Transactions on Software Engineering and Methodology (TOSEM) Volume 6 Issue 1, Jan. 1997, pp. 1-30
8. Hofmann H.F., Lehner F., 2001, *Requirements Engineering as a Success Factor in Software Projects*, IEEE Software Jul/Aug 2001, pp. 58-66
9. Ghanavati S., Amyot D., Peyton L., 2007, *A Requirements Management Framework for Privacy Compliance*, Proc. of the 10th Workshop on Requirements Engineering (WER'07), pp. 149-159
10. Merz F., Sinz C., Post H., Gorges T., Kropf T., 2010, *Abstract Testing: Connecting Source Code Verification with Requirements*, 2010 Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 89-96
11. Yue T., Briand L.C., Labiche Y., 2011, *A systematic review of transformation approaches between user requirements and analysis models*, Requirements Engineering Volume 16 Issue 2, June 2011, pp. 75-99
12. Santos Soares M., Vrancken J., Verbraeck A., 2011, *User requirements modeling and analysis of software-intensive systems*, The Journal of Systems and Software 84 (2011), pp. 328-339
13. Eisenbarth T., Koschke R., Simon D., 2003, *Locating Features in Source Code*, IEEE Transactions on Software Engineering, pp. 210-224
14. Eaddy M., Aho A.V., Antoniol G., Gueheneuc Y.G., 2008, *CERBERUS: Tracing Requirements to Source Code Using Information Retrieval, Dynamic Analysis, and Program Analysis*, ICPC 2008. The 16th IEEE International Conference on Program Comprehension, pp. 53-62
15. Ruiz J.F., Comar C., Moy Y., 2012, *Source Code as the Key Artifact in Requirement-Based Development: The Case of Ada 2012*, Ada-Europe 2012, Stockholm
16. Finkelstein A., Emmerich W., 2000, *The future of requirements management tools*, In: Quirchmayr, G and Wagner, R and Wimmer, M, (eds.) Information Systems in Public Administration and Law. Oesterreichische Computer Gesellschaft (Austrian Computer Society)
17. Hoffmann M., Kuhn N., Weber M., 2004, *Requirements for requirements management tools*, In Proceedings of the IEEE International Requirements Engineering Conference (RE'04), pp. 301-308
18. *The Clojure Language Website*, 2012, http://clojure.org
19. *Git, Website* 2012, http://git-scm.com/

CREATING VIRTUAL ENVIRONMENT FOR EDUCATIONAL PURPOSES OF SCHOOLS AND UNIVERSITIES

Bartłomiej Kacprzak, Piotr Goetzen, Alina Marchlewska

IT Institute, Academy of Management, Lodz, Poland
bercik1337@gmail.com, (goetzen, amarchlewska)@swspiz.pl

Abstract

The objective of this paper is to present how to prepare and setup the virtual machine environment on PC to improve speed, quality and overall feeling of education process with reducing costs of hardware and software. This project is a response to problem arising in most educational agencies and training centers. It is a result of continuously growing amount of data to process in today's world. What is more, the range of subjects to teach widens everyday because of market needing versatile specialist on every level. As a result, the students nowadays have to get to know variety of systems, software and ideas in short period of time.

Key words: virtualization, database, VMware, ESXi, Linux, education, reducing costs, saving time

1 Introduction

As the years pass by, computer technologies and IT services are common to people. IT market is in a need of good specialists to provide needed service. Educational institutions try to cope with that need implementing constantly growing range of specializations according to market valuation. This requires wide range of hardware (different types of computers: screen sizes, CPU and GPU power etc,), software (application for networking, graphic editing, audio editing, video editing, system administration, software development) and operating system configuration. All tasks performed by students are usually located in different rooms because all mentioned above tasks require different types of system access. If, for example, soon-to-be system administrator reconfigures NICs[1] revoking access to Internet access on this station, this would impact all other students using this station. Schools and universities are trying

[1] Network Interface Card

to fix this problem by acquiring more workstations with specialized software (e.g. DeepFreeze) witch results of rising costs.

Today's world is facing flood of information. Systems process more and more data because of growing population and ease of access to Internet. Increasing number of customers forces service providers to add more servers and storage, build new facilities and lease greater Internet connection. All that results in greater electricity bills, more space to equip and higher maintenance costs. Is it really the only path? Thankfully to virtualization - no. This technology can reduce hardware needed for specific job. Because of system consolidation, servers utilize most of system resources which leads to less hardware needs[3].

So what actually is virtualization? To put it as simple as possible - it's mechanism that separates software (for instance whole operating system) from hardware. Thanks to that "encapsulation" it is possible to run multiple OS's at this same time in one physical machine.

The very concept of virtualization is almost half century old. First working virtual machine was created in early 60’s. Even back at that time system were fully functional when ran at the same time. Since that day, personal computers gained great amount of computing power. Thanks to that, anyone can run Virtual Machine (VM) on his PC. Over past few years significant growth of virtualization market can be seen. Reason for that mostly is previously mentioned computing power. On software side of things first steps were made by MIT students. They created system called CTSS (Compatible Time-Sharing System) the ancestor of today's z/VM. PC's virtualization market was pushed forward by VMware company. Started in 1998, their products really popularized virtualization on small and medium sized corporations as well as home oriented solutions[4].

2 Education

Different types of schools nowadays have different types of path for their students with a lot of topics to go through[4,5,6]. Educational system is not coherent. This can be easily observed by looking at computer rooms where computer systems are dedicated usually for one task only. This results in wasting system resources that could be used in better manner. This section of paper will describe differences in educational programs and as a result of that – different specifications of systems required for those tasks.

2.1 Primary schools

During primary schools main topics of IT education are:
- Basics usage of computer
- Personalization of operating system
- File and folder operations
- Text editor usage [6]

2.2 High school and middle school

High schools and later on middle school implemented education of basic programming languages. Those were most commonly PASCAL and BASIC. Each student workstation had to be equipped with programming environment (code editor, compiler, etc). Also basics of HTML are quite popular during this time of education [7]

2.3 Universities

One of the least steps of education tree. There, IT knowledge is widened on multiple specializations. Most common are:
- Computer architecture
- Programming
- Computer networking
- Operating system administration
- Computer graphics [9]

3 DESCRIBING THE PROBLEM

3.1 Cases

As mentioned before, differences in educational programs project on need for specific hardware. The following cases will show some of examples of these issues.

Case one:
For purpose of teaching Linux administration specific tasks, teachers require numerous workstation and server configurations for teaching purposes. Those machines have to be equipped with wide verity of peripherals. For some tasks diskless station with 32MB of RAM will be enough. For others multicore server with gigabytes of RAM is needed.

Case two:
In the need of graphic and video editing, students have to use very powerful graphic stations, preferably with multicore CPU, reasonable or even multiprocessor GPU and the display of significant size for better editing.

Case three:
Programmers during their work spend most of time on reading documentation, analyzing existing code. That requires minimum system resources. Huge CPU and IO utilization is required only to compile the program.

3.2 Preparations and tests

Several test were done. The main aim of tests is to measure the time needed to copy the template of the system onto different locations:

- Test 1 Replication of one copy of the template
- Test 2 Copying the template between the physical disks
- Test 3 Double replication test
- Test 4 Parallel replication
- Test 5 Remote storage template copy
- Test 6 Final benchmark

For purpose of this paper, testing environment was created. It took the following steps to complete:
a. Hypervisor installation[1]
b. Hypervisor configuration[3]
c. Creating teacher's system
d. Creating student's template
e. Replication tests
f. Automation of replication process

VMware ESXi 4.0.0 build 208167 hypervisor software was chosen for this project. The following hardware was used:
- CPU Intel Core 2 Quad Q8200 2,33GHz x 4
- Memory 6GB DDR2
- NIC Intel PRO 1000MT
- HDD Hitachi 250GB SATA2

After completing tasks mentioned in steps A up to D (those steps won't be discussed in this paper) system is ready for replication tests.

In the presented example, we prepared the student's system that takes up 5GB of space on hard drive. When using build in ESXi file transfer tool time needed for data transfer on 100 Mbps network is represented in table 1

Table 1. Calculations

5GB = 5*1024MB = 5120MB
5120MB + 20% = 6144MB
6144MB / 7MB/s = 877s

Assuming that speed of network link will reach 7MB/s it will take 877 seconds to transfer file to or from host with average encryption overhead of 20% [8]

Using build in tool for replicating images is easy to use and ready out of the box, however it has some major disadvantages:

- In order to copy multiple instances from computer to ESXi server, all images should be already created on computer hard drive and grouped in one folder. It is impossible to issue command multiplying one image n-times.
- During data transfer from/to ESXi host client computer network link has to be up. It the link goes down it will result in transfer brake and will force teacher to start over.
- When the tool is used from remote location with low bandwidth connection, time needed for transfer rises dramatically.

To solve these issues it is best to use SSH connection to ESXi host and make template replications using */bin/ash* shell. In order to enable this hidden feature one must have physical access to the virtual machine host server. By pressing ALT+F1 keys hidden console will be activated. Within the console the command *unsupported* should be typed in (will not be visible) and after pressing enter key, system will ask for root password. Providing these credentials will result in instant access to shell (Figure 1). From there, */etc/inetd.conf* file must be edited. Inside the line containing *#ssh* has to be uncommented.

After ESXi reboots, SSH access on default port 22 will be accepting connections.

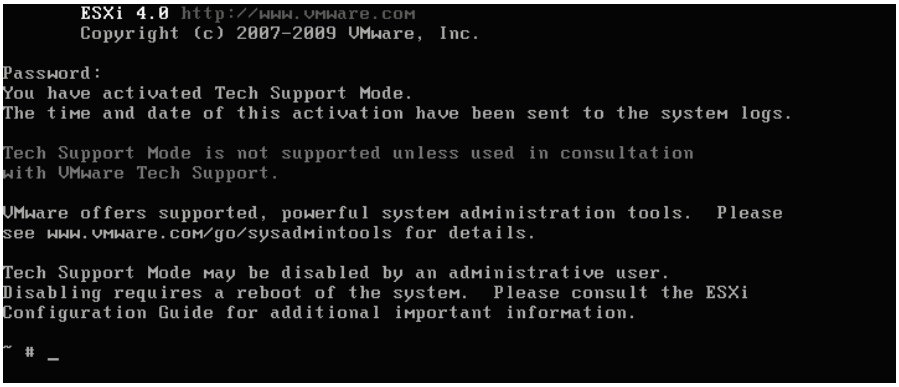

Figure 1. Hidden ESXi shell

3.2.1 Test 1 Replication of one copy of the template

As calculations show in Table 1, time needed to replicate one copy of template is 877 seconds. The following test will show how much time is needed to copy the same amount of data using SSH within ESXi box to the same physical hard drive.

Table 2. ssh copy

~ # cd /vmfs/volumes/datastore1/ /vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a # time cp -r szablon_sieciowcy/ sieciowcy1/ real 10m 40.19s user 0m 53.46s sys 0m 0.00s

Table 2 presents result of Test 1. Time needed for operation was significantly reduced. Time savings gained by this solution will multiple n-times for each replication.

3.2.2 Test 2 Copying the template between the physical disks

It is possible to connect another physical hard drive and perform copy between physical devices. To do that, after connecting hard drive to ESXi server, HDD has to be initiated and used as Datastore. In panel *Configuration>Storage>Add Storage* newly added device has to be selected.

Datastore is now ready for use. Test 2 will copy template from old *datastore1* to newly created *datastore2*

Table 3. Replication to another HDD

```
    /vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a # time cp -r
szablon_sieciowcy/ /vmfs/volumes/datastore2/sieciowcy3
    real  6m 45.37s
    user  0m 51.45s
sys   0m 0.00s
```

Executing Test 2 also reduced time needed for operation. With this setup results are even more vivid. Total time needed for copying the file was reduced almost by half.

3.2.3 Replication automation

Automating this environment will provide multiple benefits:

- Reduce time wasted by human activity
- Huge workload can be scheduled for specific time (for instance at night)
- Scripts can be easily modified and tuned to new purposes

Scripts will be created in / path. They will be issued from within */vmfs/volumes/datastore1*. After creation, proper rights to execute have to be granted by issuing *chmod +x /skrypt**

Table 4. Content of skrypt1.sh

```
  for x in `seq 1 2`; do cp -r szablon_sieciowcy
/vmfs/volumes/datastore1/sieciowcy$x; done
```

Table 5. Content of skrypt2.sh

```
  for x in `seq 3 4`; do cp -r szablon_sieciowcy
/vmfs/volumes/datastore2/sieciowcy$x; done
```

3.2.4 Test 3 Double replication test

Test 3 will consist of two actions, first, it will replicate twice the template to destination of *datastore1*, and after that replicate the same template to *datastore2* also two times. Commands and times of execution are show in table 6.

Table 6. skrypt1.sh and skrypt2.sh execution times

```
time /skrypt1.sh; time /skrypt2.sh
real  14m 13.91s
user  0m 0.00s
sys   0m 0.00s
real  19m 27.32s
user  0m 0.00s
sys   0m 0.00s
```

3.2.5 Test 4 Parallel replication

It is possible, to perform tasks simultaneously. It can be achieved by sending task to background using ampersand symbol at end of each command.

Table 7. Forking processes into background

```
/vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a #
real    24m 32.21s
user  0m 0.00s
sys   0m 0.00s
/vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a #
real    29m 48.87s
user  0m 0.00s
sys   0m 0.00s
```

As table 7 shows, concurrent replication needed more time to complete. Reason for that is because of cheap hard drive used - performance drops significantly when accessing similar area of data by two different threads. This is a good proof of why those benchmarks should be performed before actually making real production scripts.

3.2.6 Test 5 Remote storage template copy

Connecting remote resources such as NFS or iSCSI to ESXi host and using them as a remote datastore is another possibility that VMware ESXi provides. This option is very useful mainly because it skips limit of how many physical devices could be connected to ESXi host. Remote resources can be accessed from both intranet and Internet giving administrators variety options to choose from.

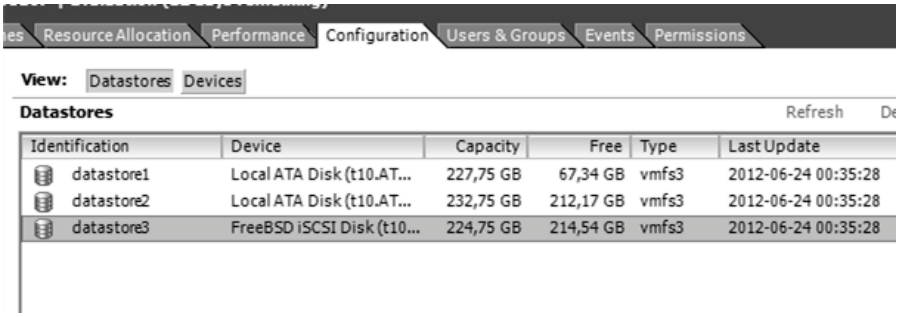

Figure 2. Remote iSCSI datastore

In this example remote iSCSI resource is used. It is located physically in the same network. Connection link speed between ESXi and iSCSI hosts is 1000Mbit/s.

One has to create *skrypt3.sh* and tune it to perform the same replication activity as shown in previous tests, except destination which, in this case, is *datastore3*. Execution time is shown in table 8.

Table 8. Replication to remote iSCSI datastore

```
    /vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a # time
/skrypt3.sh
    real  11m 11.94s
    user  0m 0.00s
sys   0m 0.00s
```

Using remote destination is another time saver, and again the amount of time saved is significant.

All tests above point that the most efficient copy operation can be achieved by queuing tasks one after another.

3.2.7 Test 6 Final benchmark

Last test will simulate complete preparation of environment for new group of students. For that following list of copies will be created:

- 9 copies on datastore1
- 9 copies on datastore2
- 19 copies on datastore3

Table 9. newclass.sh script

```
for x in `seq 11 20`; do cp -r szablon_sieciowcy /vmfs/volumes/datastore1/sieciowcy$x; done
for x in `seq 21 30`; do cp -r szablon_sieciowcy /vmfs/volumes/datastore2/sieciowcy$x; done
for x in `seq 31 50`; do cp -r szablon_sieciowcy /vmfs/volumes/datastore3/sieciowcy$x; done
```

With all already created images current number of virtual systems will be equal to 50. That number should be enough for most classes and laboratories.

Table 10. execution of newclass.sh

```
vmfs/volumes/4e4318fb-eaab2e19-f6d9-0002b3ee588a # time /nowaklasa.sh
    real  4h 48m 24s
    user  0m 0.03s
sys   0m 0.00s
```

Table 10 shows execution time of newclass.sh script that replicated 39 additional virtual machines (from 11 to 50). Those VMs are also ready to use right after replication finishes. As mentioned before, number of already crated systems should be enough to cover multiple groups of students for this subject.

Adding the system to the inventory is the last step after system replication. After choosing option Browse Datastore on each DS, newly created folder must be opened. Then, right-click on *.vmx* file and select *Add to Inventory* (Figure 3). Later, name for imported VM should be chosen. (Figure 4)

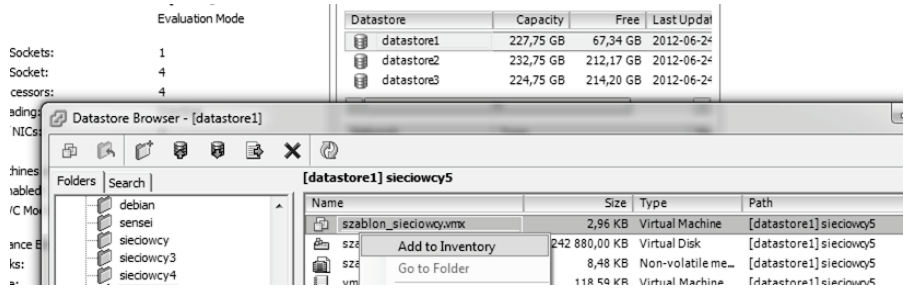

Figure 3. Adding VM to inventory

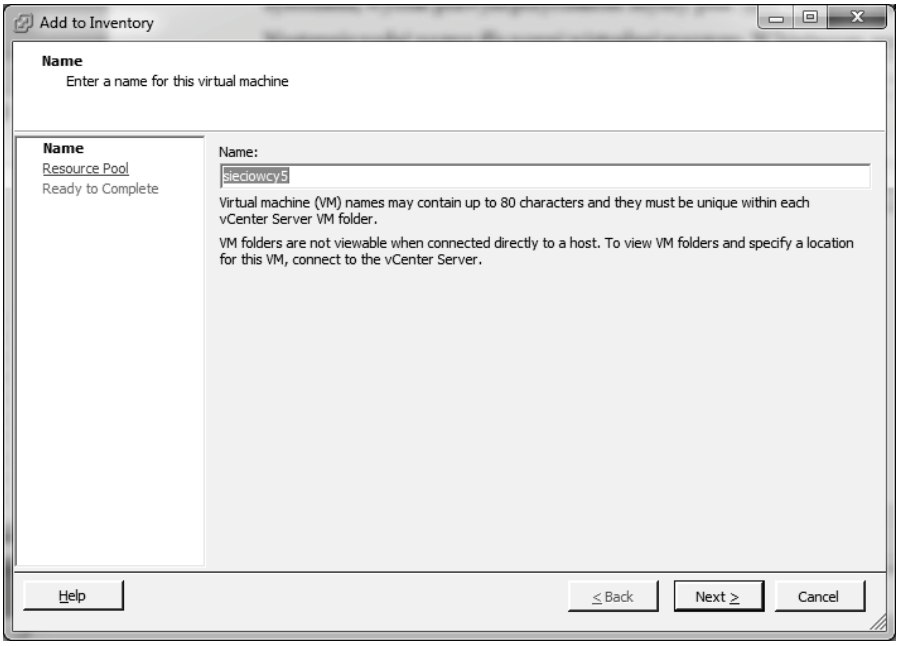

Figure 4. Naming new VM

4 Example usage

After powering on all systems SSH connection can be established. To create new definition of SSH cluster, *.cssh* file must be edited. Portion of file is presented in Table 11

Table 11. .cssh file

```
clusters = sieciowcy
sieciowcy = root@192.168.1.110 root@192.168.1.111
root@192.168.1.112 root@192.168.1.113 root@192.168.1.114
root@192.168.1.115 root@192.168.1.116 root@192.168.1.117
root@192.168.1.118 root@192.168.1.119 root@192.168.1.120
root@192.168.1.121 root@192.168.1.122 root@192.168.1.123
root@192.168.1.124 root@192.168.1.125 root@192.168.1.126
root@192.168.1.127 root@192.168.1.128 root@192.168.1.129
root@192.168.1.130 root@192.168.1.131 root@192.168.1.132
root@192.168.1.133 root@192.168.1.134 root@192.168.1.135
root@192.168.1.136 root@192.168.1.137 root@192.168.1.138
root@192.168.1.139 root@192.168.1.140
```

Next step is to physically connect local system to cluster by issuing *cssh sieciowcy --options "-i .ssh/uczen"*. After making connection, system will ask about unknown certificate. It is necessary to accept it by typing “yes”. Confirming certificate by typing "*yes" is displayed on Figure 5[2]*

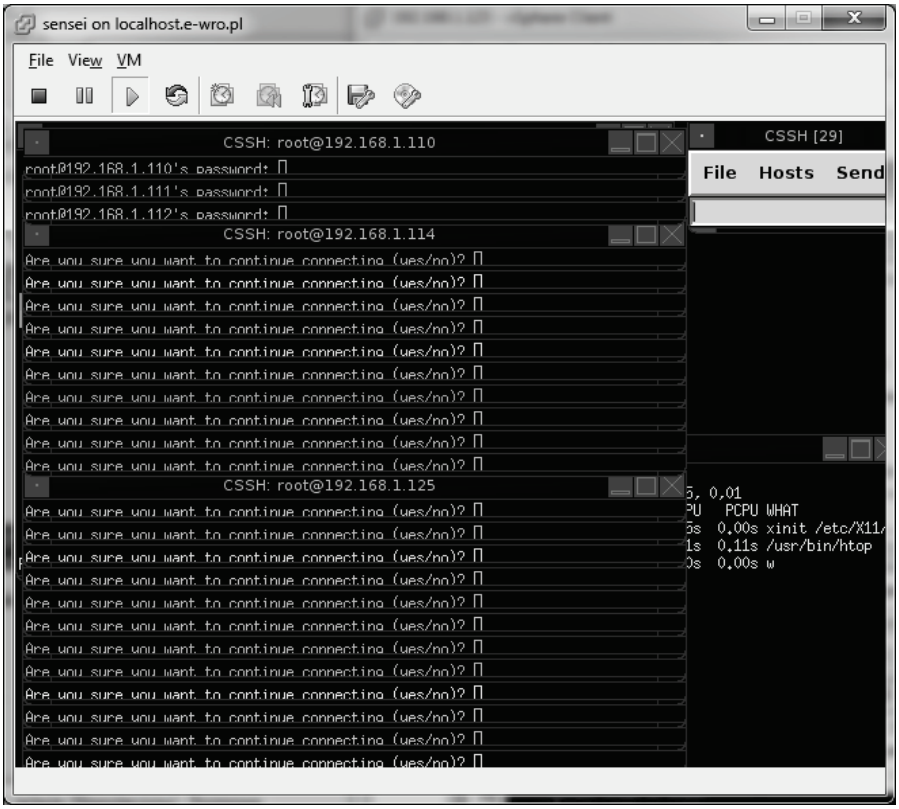

Figure 5. Initiation of SSH connection

4 Summary

Tests performed measured time needed for replication of exemplary system. Measurements showed that slowest method of replication is the one where source and destination datastores are the same. Slightly faster solutions is that with different source and destination datastores. The fastest time for task completion was the one that used remote datastores connected via iSCSI protocol. It is worth to point out, that components used in those test were rather low-class, where low cost significantly impacts performance (but low cost is the very important parameter for school budgets). Along with greater budged comes more time saving. Also, given that values shown in examples

are rather low, environment fulfills its task. Time needed for copying was shrunk from 15 minutes down to 10 at worst scenario. If one needs to install and configure the system from scratch it takes normally hours. Reducing the installation to 10 minutes is significant. Ease of maintaining the infrastructure will result not only in reducing time for preparing classroom for subject, but also improve , thanks to system centralization , all housekeeping/cleanup tasks such as removing unnecessary VMs. After having implemented this solution, effects are noticed immediately on first use. Infrastructure size is no longer an important factor in this scenario, because this solution benefits when used for either 5 or 500 systems. Moreover, implementation brings more benefits described below.

4.1 Financial benefits

One of the benefits of this system is reduction of financial costs. They derive directly from reducing used hardware. Instead of constant hardware upgrades in classrooms, which often are equipped with many computers it is possible to upgrade just one central server.

Secondly, moving high load from clients to one centralized server results in no more need for powerful hardware. For remote administration any type of hardware will be enough (both Windows and Linux). What is more, all classroom computers can be sold and superseded with brand new, cheap "Thin Clients". Another option is using embedded computers like Arduino or Raspberry PI that were meant to be small and cheap minicomputers with super-low power consumption reaching 2W - 5W at high load.

All that transformations lead into money savings. Electricity spending is reduced right after system implementation.

4.2 Time benefits

Automation of tasks will have impact on time needed for preparing classes environment and also reduce time needed for tasks during lectures and labs. As effect teacher gains more time that could be spent for focusing on subject instead of struggling with hardware problems

4.3 Management

Centralization of management simplifies control over infrastructure. It can provide monitoring facilities for resources used, and prevent from over or under utilization of equipment. Implementing additional monitoring tools and logging solutions provide teacher ways to ensure the system security.

4.4 Flexibility

Flexibility of system in both hardware and software aspects guarantees wide possibilities of further development. Possibility of main system reconfiguration is essential if system is going to be used for variety of subjects. Physical development of the server will benefit in greater savings mainly because of consolidation of many VMs into one server and at the same time reduce time needed to complete any operation. Figure 6 shows the chart of time needed for copying 5GB virtual machine image related to disk speeds of different sort. With proper budged, time needed for copying can be reduced over ten times. Multiplying it by n-times number of disks will result in even less time needed for classroom preparation.

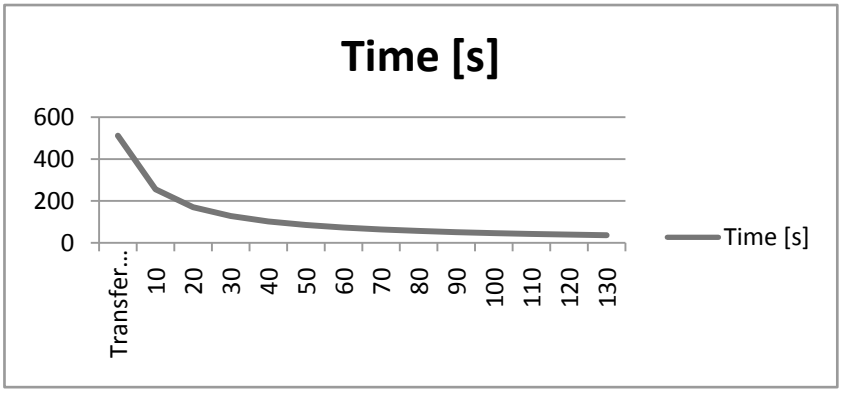

Figure 6. Disk performance

4.5 Meeting the objective

Environment created and presented in this paper meets all goals. Thanks to minimal financial input needed to implement presented solution, ease of management, better hardware utilization and money savings coming from less hardware usage were achieved. Time saved can be spent on subject/topic matter instead of wasting it on classroom preparation. Possibility of immediate logon to student systems gives teacher ability to quickly diagnose and fix problems, or mistakenly configured system. Also snapshot functionality can reduce time needed for systems maintenance and provide safe and quick facility for testing new features. Placing multiple systems onto one server is also useful when devices (classrooms) have to be moved from one place to another. There is no need for moving tens or even hundreds of devices. All that is reduced to moving one server. Any costs regarding hardware failures are also minimalized. Periodic fan, PSU or disk swapping that before used to take

significant amount of money are now independent regardless of how many systems are moved to ESXi host. Possibility of remote access to VMs for both teachers and students at home is another important feature.

4.6 Possibilities of reusing solution

Improving educational process in any type of educational facilities was one of objectives of this environment. Universality of this solution gives possibility of using it in:

- companies, as a training environment. Useful especially for new employees. Facility similar to real production environment is best solution for teaching newcomers, and yet because of not affecting the production servers any mistakes won't result in downtime or loss of data.

- companies, as testing and development environment. Possibility of creating one to one copy of production environment and testing there new functionality is an great asset. It gives opportunity to completely test new features before implementing them on production servers.

- home, as environment for learning networking and operating systems. Functionality of this solution guarantees possibility of learning different types of tasks. From basics of computer networking, analyzing how it works, up to its reconfiguration using complicated IT solutions, e.g. using HA and load balancing. All these tasks could be achieved before, but for a cost of really expensive architectures.

- courses and presentations, as lab environment. In IT branch some problems have to be shown, or require specific methods of presentation used by speaker (live presentation). In most cases regular slide presentation is not enough especially if problem only occurs during specific scenario/situation. Trying to replicate those situations in regular hotel network is extremely difficult if not impossible. Earlier prepared environment can be easily installed on ESXi host and presented on location, or remotely, in lab environment, via Internet connection.

- applications, for testing them. Great number of systems running on ESXi host and ability to simultaneously control them are very useful when simulating high traffic or specific behaviors on application or website. Thanks to this tool most common mistakes that come from improper resource estimation can be found even before passing software to tester group. As a result software preparation should require less time during creation process and what comes out of it will be resistant to some type of anomalies.

To wrap-up this paper, presented solution brings great amount of benefits, that impact both time and quality of work and education environment. All that can be achieved with almost no cost when using hardware already possessed, or with minimal investment depending on solution purpose. Lack of investment when implementing presented solution won't impact on any possible development of environment. At any time infrastructure can be upgraded to more power full one without the need of VM reconfiguration. It is also possible to completely move all systems to brand new ESXi box without any loss of data.

References

1. Lowe S., 2010, Wiley, VMware vSphere 4 Administration, , pages 21-58
2. Hertzog R.; Mas R., 2012, *The Debian Administrator's Handbook*, Freexian SARL,
3. VMware Inc, 2011, *ESXi Configuration Guide ESXi 4.0*, VMware, Inc, pages 11-153
4. Kacprzak B, 2010, *Wirtualizacja w systemach X86 X64 i z/*9, SWSPIZ Diploma work, pages 3,5
5. Sysło M.M.,, *Myślenie komputacyjne. Informatyka dla wszystkich uczniów,* http://www.up.krakow.pl/ktime/symp2011/referaty2011/syslo. pdf, page 1
6. Stanecka B., Stanecki C. *Program Nauczania Informatyka W Szkole Podstawowej*, http://www.stanpol.edu.pl/Pdf/Program-InformatykaSP. pdf, site 13
7. Filinowicz E., *Program nauczania informatyki w gimnazjum, Informatyka dla Ciebie,* http://www.nowaera.pl/index.php?option=com_docman& task=cat_view&gid=100036&Itemid= site 9
8. Goleń, P., *EFS kontra TrueCrypt, czyli dlaczego szyfruję cały dysk* http://wampir.mroczna-zaloga.org/archives/377-efs-kontra-truecrypt-czyli-dlaczego-szyfruje-caly-dysk.html
9. Strona Główna. *Informacje dla wykładowców i Studentów*, http://wazniak.mimuw.edu.pl/

A TESTING ENVIRONMENT FOR DISTRIBUTED SYSTEMS

Marcin Sztandarski, Grzegorz Sowa,
Piotr Goetzen, Alina Marchlewska

IT Institute, Academy of Management, Lodz, Poland
marcin.sztandarski@gmail.com, (gsowa, goetzen, amarchlewska)@swspiz.pl

Abstract

The article presents the basics of modern software testing theory. Testing automation and the integration of testing into code writing will be examined in detail, and concept of a testing environment for distributed systems will be introduced.

Key words: distributed systems, software testing, testing automation, test-driven development

1 Introduction

The architecture of modern software systems is complex as most systems are distributed systems. Testing this type of system is a fairly complicated process, due to the various system platforms on which the software is based, the lack of the specification of the interfaces between system modules, and the difficulty in preparing the whole environment of the distributed system.

Over time, a variety of tests and testing methods have appeared. Along with the development of agile methodologies, testing has gone hand in hand with software creation from its earliest stages.

2 Software testing theory

2.1 Quality management and software system testing

Institutions that choose distributed systems tend to be, for example, financial organisations or large logistics companies. These systems are expected to function reliably and above all in line with their specifications. Quality management is essential, and one part of this is testing. In quality engineering

interdisciplinary methods are used. Practical skills in the business processes which the system is designed to serve are needed [2]. The testing is aimed at component systems of varying granularity– individual classes, components and the whole system are tested. [3]

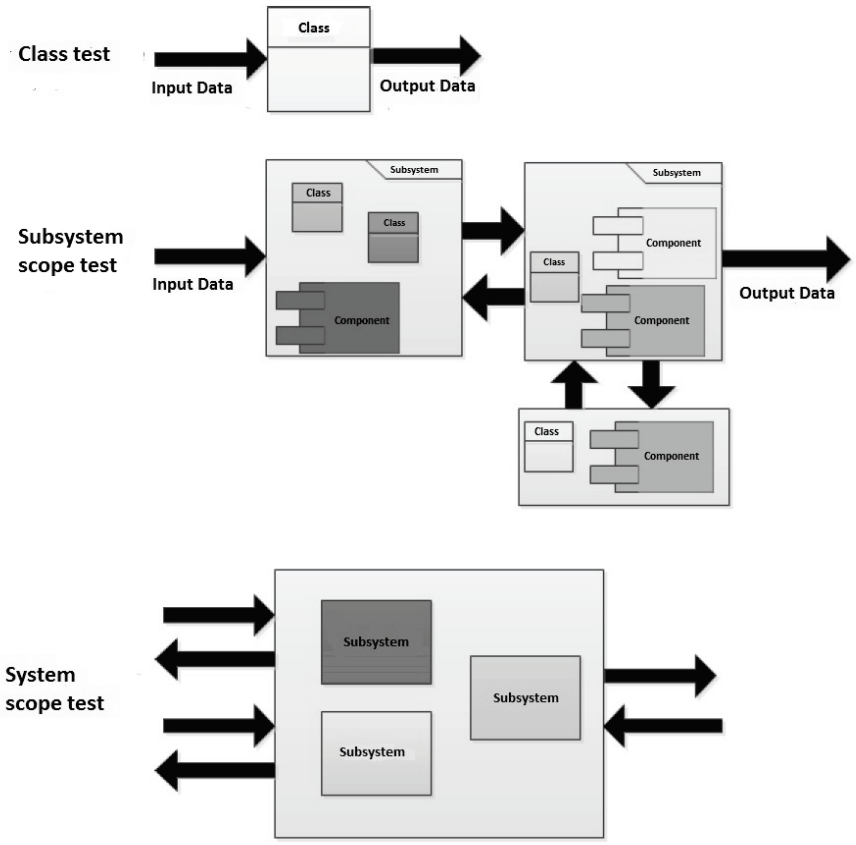

Figure 1. Typical test range (on a basis of [3])

Test design generally includes the following steps:
- analysing and modelling the expected system behaviour
- designing test variants (entry and exit)
- designing test variants arising from structural analysis and other error detection methods (e.g. heuristics)
- stating the expected results for each test variant.

Several models are used to cope with the complexity of the system. Each model describes a particular type of test and has a particular aim.

2.2 Test automation

Testing a large amount of software creates the need for test automation. An automatic test systems allows entry data to be applied and for test results to be verified. Such systems must be compatible with the interfaces and infrastructure of the system being tested. Test automation systems ensure that tests can be carried out repeatedly. This facilitates regressive tests, for example, which are generally carried out after the introduction of a repair or new function. Although it is estimated that testers find only 15% of errors through automated tests [6], the most interesting attribute of this kind of test is its repeatability, which allows the test procedure to be generated in a different hardware platform, for example, or in another configuration.

2.3 Black and white box testing

There are two ways to design tests. If the designer only takes the entry and exit specifications of the system into account, then the test is a black box test. For example, testing a log-in box with two fields: “user” and “password” and the “log in” button. The tester enters data according to the specifications of the test into the appropriate fields, and then checks the effect of pressing the button. In the case of this test, the internal data processing is not relevant – only the exit data obtained for specific entry data are checked [10].

The opposite approach in known as white-box testing. Here, the internal structure of the system is taken into account. The steering path routes in the system being tested are analysed, and also solution implementation methods. In the case of the log-in window, the internal components and the interaction between them will be checked.

It is increasingly common for a distributed system to be rolled out by many teams simultaneously, with solutions being delivered at different points in time. Part of the system maybe created outside the main organisation. White-box tests are designed only for the parts of the system, which are implemented by the designer, whereas complex black-box test are the best means of quality control in cases when we ourselves have not created the code.

2.4 Typical software development processes and testing

Generally speaking, the process of creating software is based on the translation of information, such as information about a business process, into source code. This information is transferred during the following stages of software creation [8]:

- Needs establishment: the user (the sponsor of the project), working with an analyst, defines what is expected of the system that is to be created. These expectations are written down as the formal aims of the project.
- Analysis and description of goals: the details are agreed upon and the relationship between each goal is specified, taking into account priorities, affordability, and any compromises.
- Creation of external specifications: system elements are described as "black boxes", in other words only interfaces and the interaction with the user (and in the case of batch systems, entry and exit specifications) are specified.
- Creation of the project structure system structure: the system is divided into a series of elements of decreasing granularity. In other words, it is divided into programs, components, etc, and interfaces are also defined.
- Functional specification of modules and their interfaces: the function of each module, the relationships between modules and working guidelines are established.
- Exact specification of each module, defining the interface and functional elements.
- Creation of source code.

Incorrect transfer of information can occur at any of the above stages. In order to eliminate mistakes, testing processes are used simultaneously with each stage. Each testing stage is responsible for eliminating a particular kind of mistake. The relationship between software creation processes and testing processes is often represented as a "V" shape, where the series of stages associated with creating the system are on the left side, and the corresponding tests on the other. The "V" model, which is an extension of the typical waterfall model, can be adapted to methods, where an iterant approach is used, as well as to agile methodologies. In this case, the route through each stage is completed for each iteration. This approach improves the reliability of each stage of system creation, as each particular type of mistake is eliminated as soon as it might appear. Thus, when the external specification of the system is being tested, functional testing is carried out rather than broad system testing. The tester focuses on mistakes in functionality implementation, and not on, for example, data processing efficiency.

The test structure along with the corresponding stages is as follows:

- Acceptance testing at the needs establishment stage. This type of test assesses to what extent the system being tested corresponds to the expectations set out in the specification. Often this stage of testing is carried out by the client's own team of testers. Functional as well as non-functional expectations are tested (e.g. efficiency).

– Systems testing at the goals description stage. This kind of test is essential due to the face that certain characteristics and functions of the system are only visible when the software is treated as a whole. The difficulty of designing this type of test is due to the fact that the document which describes the project aims is a general one, and so cannot provide specific systems tests. Therefore, user documentation is also used when designing systems tests.

There are several categories of system tests which focus only on specific aspects of the system. They are not used in every system, however.

– Facility testing – which tests their compatibility with the established aims.
– Volume testing – which checks how the system copes with a large amount of entry data.
– Stress testing - which assesses the system's ability to process a large amount of data in a short time.
– Usability testing – which checks the how user-friendly the interface is.
– Safety testing (data protection) checks, among other things, whether the system is vulnerable to data leakage.
– Effectiveness testing assesses how the system copes with varying demands, i.e.: whether the time it takes the system to produce an answer in a given configuration matches the estimated times.
– Configuration testing checks the system in terms of its ability to cope with different equipment configurations and environments (for example different browsers in the case of internet applications.)
– Compatibility and conversion testing checks whether data can be converted from one version of the system to another (for example, whether data can migrate from previous versions) or whether the system can work in so-called compatibility mode.
– Installation procedure testing aims to identify mistakes in the software installation process.
– Reliability testing establishes whether the system can carry out given functions in particular conditions.
– Emergency function testing checks how resilient the system is in case of breakdown. Most frequently, the average time it will take for the system to recover after a breakdown is estimated (Mean Time To Recovery).
– Service testing involves checking to what extent service and conservation of the system are possible. For example, the test establishes whether a status report can be produced.
– Documentation testing aims to eliminate ambiguity, and the documentation is assessed for completeness and detail, among other aspects.

– External specification functional testing identifies any discrepancies between the expected behaviour of the system from the perspective of the user and its actual behaviour.

The testing processes listed above are very much horizontal tests. In subsequent development stages, tests at the level of individual units are used, as follows:
– Integration testing aims to identify defects in interfaces and in the interaction between units.
– Unit (module) testing.

2.5 Units testing

Program units are part of the program code, in the form of sub-programs, classes or methods. Units consist of the smallest element of the system which it is worthwhile testing. Initiating testing while writing a unit, for example a class, has a host of benefits for the programmer. The specification of how the code behaves in tests will significantly facilitate the analysis of the code by other programmers. Refactorisation is safe – the programmer does not worry that he or she will change the way the code works when changing the code, creating a mistake [9]. It is important that the programmer is expected to take care over the project: dividing the code into parts, according to which particular unit they are intended for. Unit tests which have had their connection with other parts of the system removed can be carried out simultaneously.

Michael Feathers [4] outlines certain desirable characteristics of unit tests:
– Unit tests should work fast,
– Unit tests should not communicate with the database,
– Unit tests should not used network communication,
– Unit tests should not use the file system,
– The programmer cannot carry out additional preparation procedures in order to carry out a unit test. Programmers sometimes introduce additional connections into unit tests, for example a connection with the data base, which turns the unit test into an integrative test.

Creating unit tests requires a well designed system, in which there are not many connections between modules. This enables classes and methods to be tested independently of each other. One method of isolating classes is to use mock objects, which does, however, increase the complexity of the system. Another method is appropriate system design and application, for example, dependency injection [11].

2.6 An outline of test-driven development

Test-driven development (TDD) is a software development system which consists of three steps: creating a test, writing an appropriate code, refactorisation. This cycle is often known as the "red-green-refactor mantra" among programmers who use TDD, which is due to the behaviour of TDD support tools, in which red means that the test produced a negative result, and green that it produced a positive result. The most important, seemingly simple rule of TDD is the golden rule: never write a new functionality before you have written a test, which produces a different result than expected (red).

Code writing using TDD consists of the following steps [4]:

1. Choosing the task and creating the test – the programmer starts writing a testing code which specifies the desired behaviour of, for example, a certain method. Of course, the code is not compiled, as there is no implementation code.
2. In the next step (represented as red), the minimum implementation code necessary for the particular class/method being tested is created, with the aim of enabling compilation. The testing tools are marked as red.
3. Writing the correct code (represented as green) means implementing the method in such a way, as to fulfil the requirements of the test. This step lasts up until the point when the colour green appears.
4. Refactorisation, which means modifying the structure of the code that has been tested without changing its functionality. These changes generally aim to improve the code's readability.

It is very important to note that steps 1-4 are carried out cyclically (even multiple times an hour), whereas completing one cycle gives the programmer immediate feedback. Another advantage of the TDD technique is that the programmer focuses on one task – either code writing or refactorisation. Moreover, applying this procedure fully in a project gives the programmer a sense of security when introducing changes in the code later on, since the base code is covered by tests. This technique is also suitable for legacy application. Both bug fixes in existing code and changes introduced to existing functionalities should begin with test writing [1].

2.7 The effectiveness of test-driven development

The principal benefit of TDD is the assurance that any mistakes made during the implementation of corrections or new functionalities will not introduce hidden errors. The "golden rule of TDD" ensures that every functionality has its own test. There is also a collection of regressive tests which allow the program to be retested in order to identify newly introduced mistakes. Following

the TDD rules generally leads to hundreds of tests a month and thousands of tests a year being carried out, which in practice cover more than 90% of production code [7]. Tests from which dependencies have been removed should only take a few minutes, even if there are a few thousand of them. This means that the programmer can check the effects of introducing a correction on the rest of the code within this short time. Similarly with refactorisation, the programmer is not afraid of making changes even in "messy" code (for example code in which abstractions are mixed, such as business rules and limited access to data) since it is practically impossible to "break" the code. Unit tests are also the most readily comprehensible documentation for programmers, since they are written in the same language as the system is created in.

In 2007, Ron Jeffries and Grigori Melnik [5] presented the results of research into Test-Driven Development techniques in the IT industry. Regardless of the benchmarks used, all the research results indicated that product quality increased significantly.

2.8 The problem of units integration

During unit testing, units are tested in isolation from other elements, which means that their code does not establish a connection with „the outside world". The units being tested do not communicate through the network, do not save files, do not go beyond the boundaries of the process. Unit testing should then be expanded into integrative testing. Various strategies are used to integrate modules: growth integration, increasing and decreasing integration [11]. The purpose of integrative testing is to identify defects in the interface and in interactions between units. Units in complex systems (and also distributed systems) give access to the interface or carry out calls for methods made accessible by other units and cannot be tested individually. It can be difficult to test interaction, because certain parts of the system may not yet be accessible.

In order to resolve this, environmental elements which replace the surrounding modules are used:

- The driver unit – calls to the tested unit are carried out form the level of the driver.
- The stub unit – creates access to the interface of the unit, whose methods are called up.

Figure 2 shows models of the configuration of test units for two examples of distributed systems:

System I illustrates an imaginary configuration for complex tests where interaction is taking place between three units (unit 3 has been replaced by an element which gives access to the interface of unit 2.)

System II is an integrative test which checks the interaction of module A with the outside world. The driver of module A enables calling up the methods of interface A, whilst the element of module B enables carrying out calls in the range of module B's interface.

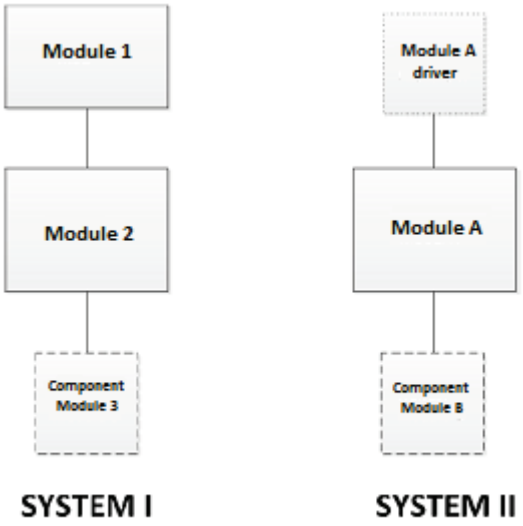

Figure 2. Model of configuration of units in integrative tests (complex texts) for examples of distributed systems. Source: own design.

3 System concept for testing distributed systems

3.1 High-level design architecture

High-level design describes components and their functions in the process of testing distributed systems. Only significant aspects will be discussed here. Figure 3 illustrates the main subsystems and their communications.

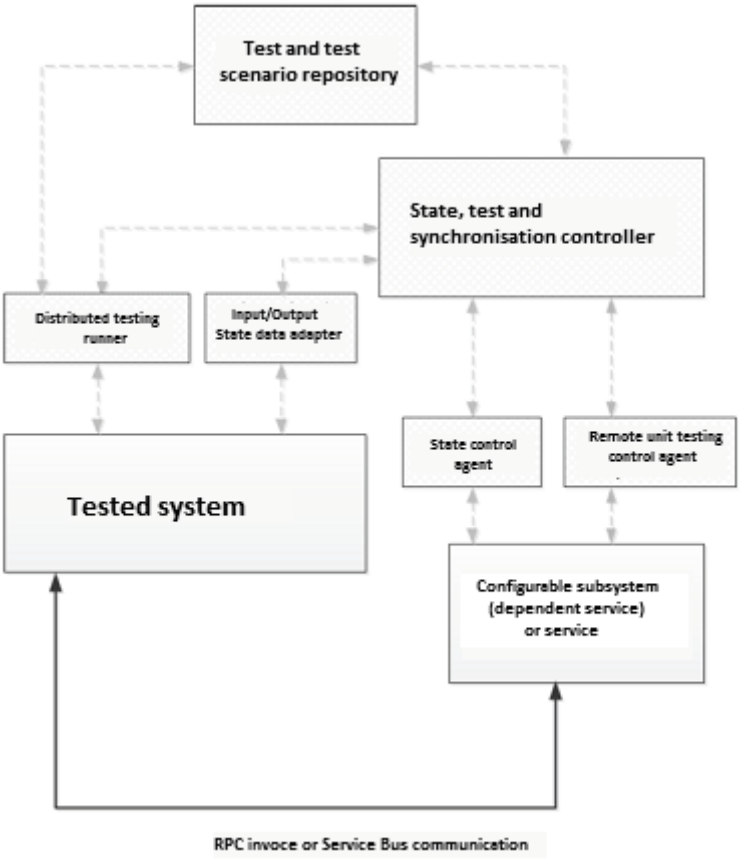

Figure 3. A diagram of the testing environment architecture of distributed systems. Source: own design.

3.2 The concept of test fixture

A test fixture is an agreed system status which ensures that test conditions are repeatable. The test fixture is created by the test environment. In practice, this is a service steered from the level of the status control agent. The test fixture is created by the system being tested along with the adapter.

3.3 What is required of the system being tested

In order for the system to be testable, two conditions must be met:
- The communication interface must be separated from the outer components which the tested system communicates with
- The data adapter must be completed.

3.4 The role of components (of subsystems)

The runner subsystem of distributed testing is a component, which has the function of communicating with the system being tested in order to initiate the given command, forcing shut down of the testing procedure, or dealing with a system failure.

The data adapter is a component which enables the entry data transferred to the system to be read, the exit data to be stored and the system status to be dumped.

The status, test and synchronisation controller is a subsystem responsible for the co-ordination of the whole environment and synchronizing tests. Status co-ordination involves sending commands to the status control agent. The commands are sent according to the content of the test fixtures. The controller also sends test signals to the runner controlling the system being tested.

The status control agent carries out commands to set up a service or an imitation service in a particular way.

The remote unit test control agent.

System imitation is an external service, which the system being tested depends on.

The repository of tests and test scenarios is an application which allows test scenarios and reports of tests that have been carried out to be collected.

4 Case study

4.1 The concept of the Long Running Process Server framework

A high-level design test for a system which supports key business processes is described below. Testing this system depends on the availability of many components and external systems.

The Long Running Process Server framework is a collection of components and interfaces installed in business applications which demand service for processes that are not synchronized. Processes that are not synchronized allow the application to send a command for a long-term task to another machine, and then to continue functioning without waiting for the result. It is also possible to finish the command process and memorize the task handler,

which would make it possible to refer to the result after the next command process has been initiated (or alternatively to delegate the reading of results to another process).

4.2 Architecture of the Long Running Process Server framework

A description of the high-level components needs to be added to the framework concept described below in order for the testing scenario to be clear.

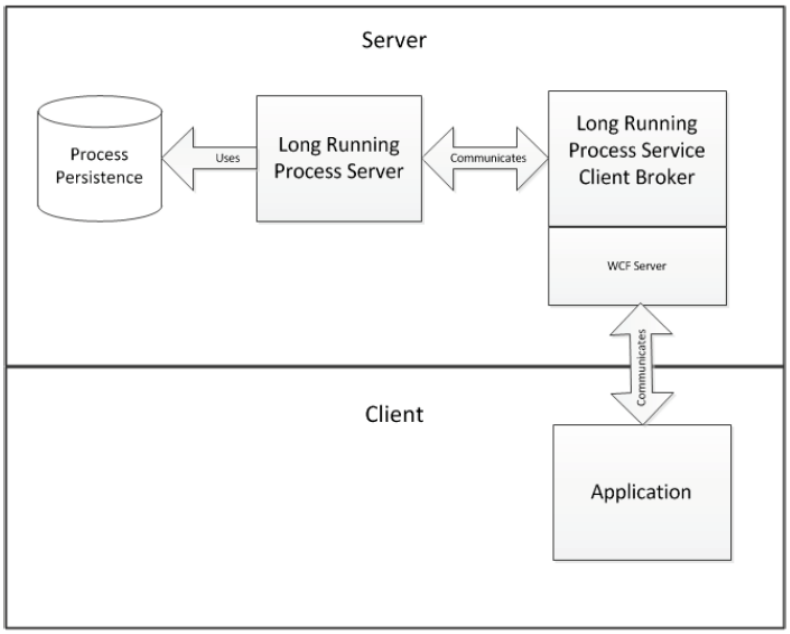

Figure 4. Architecture of the Long Running Process Server framework. Source: own design.

The structure and organization of the server are as follows. The Long Running Process Server (executive server) is a service for the Windows operating system. When the service is switched on/started, a configuration which identifies plugins is loaded, with a task executive element. After the plugins are loaded, the service process checks whether there are any tasks to be performed in the queue, and if so begins to process them according to the allotted task ordering algorithm. Both the executive parameters and the task process exits are consolidated in line with process persistence; in this implementation these are XML communications saved in the MS SQL Server base. The ser-

vice also opens up the interface for communication with the broker process. The executive server deals with all connections with the outside world for the tasks requested.

The Long Running Process Service Client Broker is a service for the Windows operating system which is a bridge between the Long Running Process Server and client applications. In practice, the broker is dedicated to one particular application and only serves specific tasks. The component is made up of the self-hosted Windows Communication Foundation Server, which serves different kinds of communications with clients, and a functional part which supports the loading of plugins for particular tasks. The broker is responsible for passing on instructions, carrying out tasks with entrance instructions and for answering questions about their status (and enabling results to be recorded.) The service makes the service accessible for client applications in the form of Remote Procedure Call requests. The Long Running Process Server can serve requests from many brokers and many applications, which means that the brokers can be specialized in terms of function.

Client applications are processes which have a shorter life cycle than the tasks requested by them. They are generally made up of a presentation layer and a layer which is responsible for communication with the broker. The presentation layer is built from component provided by the framework library.

4.3 Examples of business system testing scenarios, based on the Long Running Process Server.

The scenarios described below are examples of integrated systems testing which supports business processes in financial institutions offering clients credit, credit cards and medical care/insurance. There is often a considerable delay in processing tasks which require communication with outside systems, due to business working conditions such as the need for a form to be approved, or the availability of outside services, and so these tasks are transferred to the Long Running Process Server by the Window application operator. The correct implementation of the business process completion can be found in the BusinessTaskDisposal component.

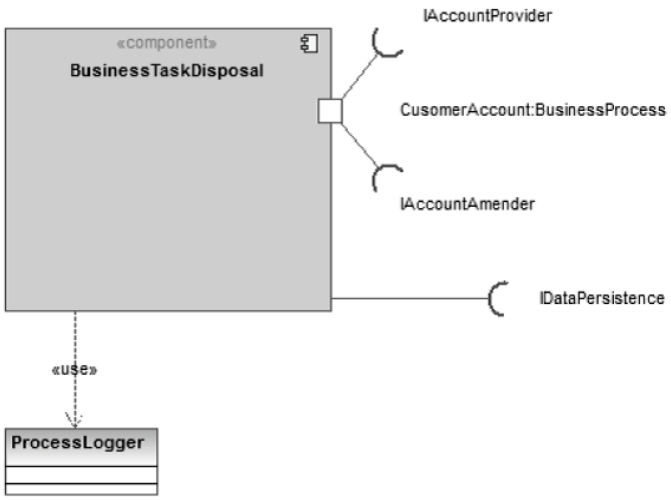

Figure 5. View of the BusinessTaskDisposal component. Source: own design.

The component that carries out tasks through the use of the Separated Interface model uses the potential of the external implementations of servicing client account services (IAccountProvider, IAccountAmender) and data consolidation (IDataPersistence). This approach also allows the stubs of any service to be used and focus tests to be carried out on only one service. The dependence on IDataPersistence is due to the fact that the Long Running Process Server consolidates entry data, exit data and the task status. Another important dependency is the ProcessLogger class, which consolidates data related to communication about the process status (this data is used to diagnose and display the status on the client side). The testing environment configuration in the case of each hub is made up of: the status control agent, the remote test control agent and the imitation of certain dependent services, which the task processes communicate with.

Smoke test scenario (preliminary test)

The preliminary test checks whether the system is ready for further, detailed tests to be carried out.

Test scenario 1 – read status of all system hubs for the configuration that has been loaded, expected result – status {OK} for each hub.

Test scenario 2 – load imitation of every dependency for BusinessTaskDisposal, and then carry out the given remote functional test.

Regressive test scenarios for given tasks

The aim of regressive tests is to ensure that no new mistakes have been introduced when making changes in the software. This test consists of repeating tests carried out before the changes were introduced.

Test scenario: load the regression test baseline set which contains the test examples that were saved along with the test fixtures, run the automated tests, check the results.

End-to-end test scenario

The aim of end-to-end tests is to test business transactions at the level of their components, i.e. to ensure that all components are working together correctly and processing data correcting (at the level of the business process).

Test scenario for a single end-to-end test:

1. Set the hubs to a status with no imitations.
2. Carry out a preliminary test for each hub which is part of the process
3. Transfer data to be processed on the client side
4. Run the test
5. Check the processing results for each hub.

Test scenario for functional tests for business processes which are carried out on the server side

During functional tests the implementation of the business function is tested (a black-box test).

Test scenario:

1. Set the dependent service hubs
2. Set the service statuses
3. Load the entry data
4. Run the test
5. Compare the actual and expected exit data

5 Conclusion

5.1 Complexity

Testing distributed systems is complicated. The behaviour of the network is to some extent unpredictable and preparing the testing environment is difficult. The test designer can only create an approximation of the conditions in which the system is going to function. Carrying out tests related to subsystem communication in distributed systems is often very expensive, because an organization might only have one production environment serving mass

communication with many clients at their disposal. Another problem is the question of how to simulate the behaviour of hundreds or thousands of clients. Moreover, network technologies are constantly being improved, while the life cycle of distributed systems is relatively long, which means that these systems have to function in a network environment which will be ten or more times faster in a few years.

5.2 Safety

The role of safety testing in distributed systems is increasingly important. The characteristics of distributed systems, their division into many subsystems, leads to an increased risk of data leakage. Testing the safety of the system as a whole at the end of the project, i.e. during the final integration, may not be sufficient, since repairing problems at this stage could be very expensive. Safety is a factor which should be clearly defined at the beginning of the project when the client's expectations of the system are set out. Safety testing begins with checking it at the level of components, and ends with testing at the level of the final integration and acceptance tests.

5.3 Distribution

The distribution of the project may extend beyond the boundaries of the organization. Many companies benefit from outsourcing and off-shoring. The quality of a system which has been prepared outside the organization must be measured. The test designer has to deal with the difficulties of constantly expanding acceptance tests which check both functional and non-functional characteristics. One method is the introduction of iterative methodologies which support such an approach. Creating distributed systems in a non-iterative way (for example the waterfall model) is too risky, as problems become apparent at the end – during the integration of the whole system.

5.4 Heterogeneity of environments

Designing tests for distributed systems requires a knowledge of the characteristics of many environments: equipment platforms, operating systems, debuggers. Often distributed systems are built with the aim of adapting old systems to new business conditions. One example would be any kind of banking system, which display part of their functionality in client systems like home banking. Therefore, an architect designing tests for this kind of system needs to know the characteristics of both the new and the old parts of the systems (e.g. a banking system such as core).

5.5 Automation

In order to automate tests for distributed systems, they need to be built in a particular way. In practice, this means, for example, adding a thin interface layer between the software containing the use interface and the layer below (for example, the business layer). An approach called hexagonal architecture is used, which means adding a range of adapters to the system which allow, for example, interaction with the user to be replaced with a range of automated API calls.

5.6 Synchronisation

In distributed systems, tasks are carried out in parallel and sometimes the processing cannot be done by a single machine. Tasks carried out on many different machines influence each other constantly, which means that they must be synchronized. This may cause errors that are difficult to diagnose, for example the appearance of blockages (in the file system as well as in the database).

Summary

The above attempt to design a testing environment for distributed systems aimed to solve a particular problem – testing an asynchronistic task processing server. The high-level design presented here was a very simplified one. A detailed design for the status controller component, tests and synchronization will be a considerable challenge. The flexibility of the system and ease of testing will depend on this implementation. The way that the system deals with unforeseen circumstances such as breakdowns, transaction cancellations or the peculiarities/characteristics of network communication will be significant.

References

1. Baley K., Belcham D., Manning 2010, *Brownfield Application Development in .NET*, , p.105
2. Bereza-Jarociński B., Szomański B., Helion 2009, Inżynieria Oprogramowania. Jak zapewnić jakość tworzonym aplikacjom, p. 69.
3. Binder V. R., WNT 2003, *Testowanie systemów obiektowych*, p. 48.
4. Feathers M., 2012, *Working Effectively with Legacy Code*, http://www.objectmentor.com/resources/articles/WorkingEffectivelyWithLegacyCode.pdf, downloaded Oct the 22

5. Jeffries R., Melnik G., 2007, IEEE SOFTWARE, *Professionalism and Test-Driven Development*, May/June 2007, IEEE Computer Society, p. 28
6. Kaner C., Bach J., Pettichord B., Wiley 2002, Lessons Learned in Software Testing: A Context-Driven Approach, p. 101.
7. Martin R., 2007, IEEE SOFTWARE, *Professionalism and Test-Driven Development*, May/June 2007, IEEE Computer Society, p. 33.
8. Myers G., Sandler C., Badgett T., Thomas T., Helion 2005, *Sztuka testowania oprogramowania*, p. 151.
9. Osherove R., 2009, *The Art of Unit Testing*, Manning, p. 55
10. Shaefer H., 2012, *What a Tester Should Know, even After Midnight*, http://www.sjsi.org/webgears/files/sjsi/File/tester/tester_5.pdf, downloaded Oct the 22, p. 40
11. Shore J., Warden S., Helion 2008, Agile Development. Filozofia programowania zwinnego, p. 354

WAYS OF SELECTING INTERNAL PATTERNS IN MULTILAYER PERCEPTRON NETWORK

Marcin Kolibabka[1], Andrzej Cader[1],
Agnieszka Siwocha[1], Marcin Krupski[2]

[1] Information Technology Institute,
University of Social Science, Lodz, Poland
(mkolibabka, acader, asiwocha)@spoleczna.pl

[2] Department of Computer Science in Economics
Institute of Applied Economics and Informatics
Faculty of Economics and Sociology
University of Lodz, Poland
mkrupski@spoleczna.pl

Abstract

Creating and later learning one-way neural networks depends on many factors. Selecting many of them has estimated and experimental character. The suggested method is the Allows weakness of the influence of the not optimal choice of the net structure, also speed and momentum values are less influential in classic Back then Propagation Method. There are few modes of choosing elements to use in Followed algorithm

Key words: neural networks, artificial intelligence, back propagation

1 Introduction

Simple to implement one-way, multi-layer, non-linear neural networks called MLP (Multi-Layered Perceptron) [1] are conventional. For practical use of the network, however, it is necessary to construct an appropriate network structure as well as teaching it the proper reactions, relevant to the problem given.

The principles introduced in the late 80's of the twentieth century, describing the capabilities of neural networks - each limited continuous function can be approximated with arbitrarily small error by a network with one hidden layer [2,5], moreover, any function can be approximated with arbitrary accuracy by a network with two hidden layers [2,4] - and the development of algorithm of error back propagation (English EBP - Error Back Propagation) [4]

directly contributed to their prevalence, after earlier, long-term abandonment of research on them. For many applications, they are also predisposed by the relatively simple structure of the taught network, which combined with properly organized, parallel processing of signals allows for fast obtaining network's reaction to change of the input parameters.

Classification tasks are one of the key issues that are being solved with the usage of perceptron network. In the process of learning network „remembers" the patterns from the training ensemble and generalizes their forms in order to be able to recognize new input. This is obviously possible in the perfectly extending learning process. In practical applications such optimal solutions can be achieved by experimentation with learning parameters and network's structure. Each limitation of the number of experiments is therefore beneficial. Work developed method allows to reduce the impact of not-optimal network's structure and increases the speed parameter range of values and learning momentum, at which one achieves beneficial learning results. This method is an extension of the classic error back propagation method of enforcing a common standard for group of scales [6,7].

2 The selection of a multi-layer perceptron Network's structure

Selecting the proper number of layers and neurons for the usage of the network in the problem given has highly experimental nature. Kolmogorov's theorem for its theoretical nature has little practical significance, and even then it can only refer to a network with a single output and moreover with a linear activation function.

More significant in this regard is the statement:

Let's suppose that Φ is any continuous sigmoidal function. Then for every continuous function f defined in the [0.1] n> = 2, and for any ε> 0, there exists an integral number N and the ensemble of constants α_i, θ_i *and in j,i=1,..., N, j = 1, ..., n, such that the function*

$$F(x_1,...,x_n) = \sum_{i=1}^{N} \alpha_i \Phi\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i \right) \tag{1}$$

approximates the function f, ie,

$$|F(x_1, \dots, x_n) - f(x_1, \dots, x_n)| < \varepsilon \; for \; all \; \{x_1, \dots, x_n\} \epsilon [0,1]^n$$

However, it also has its limitations. For example, it cannot be used in classification problems for more than two groups.

Apart from the problem of selecting the number of layers, the proper selection of number of neurons in each of them has great importance. Obviously,

too small number of neurons prevents network from learning, because the network has too small information capacity then. Alternatively, one could select too big network, but this solution has even several disadvantages. The least troublesome is the extension of the learning time. The most impending the usage of the network is the fact that the redundant network tends to „over-learn". It manifests by a loss of the ability to generalize knowledge, which means that the network can recognize only the data from the training ensemble in such case. It cannot properly identify the data, which are in scope of the task domain, but have not been used during the learning phase.

The number of neurons hidden in the network allows to estimate the so-called Vapnik- Chervonenkis dimension (VCdim) [8]. This dimension for the ensemble of functions is defined as the maximum number of vectors, that can be grouped in all possible ways, by using the function from this ensemble. For the neural networks, it allows to estimate the generalization capabilities through expressing the relationship between them, the amount of learning samples, network's learning error and the generalization error. Unfortunately, the assignation of this dimension is usually very difficult and the evaluation is a very „imprecise".

$$2\left\lfloor \frac{K}{2} \right\rfloor N \le VC\dim \le 2N_W\left(1+\lg N_n\right) \tag{2}$$

where:
K - the number of neurons in the hidden layer
N - size of the input
N_W- the number of network scales
N_n- the number of networks neurons

In practice, this requires tedious testing networks with different amounts of neurons anyway. Such testing requires a cyclic learning, testing, and removing the redundant scales. And even using the algorithms: Optimal Brain Damage [9] and Optimal Brai n Surgeon to reduce the network's structure, does not accelerate the process of obtaining its optimal working significantly. Therefore, it would be beneficial to obtain such a learning process that would allow the network with not optimal structure, to work as well as the optimally structured network.

3 The method of enforcing the internal formulas

In methods from the error back propagation group the algorithm is based on the assumption of minimizing the error E. This value is the sum of the errors calculated for each training data vector. In such methods, a change in the learning scale value depends directly only on its previous value. None of

these changes is combined with the change of the other scale in the same iteration, and even in different iterations this change is indirect through the value of the inherited error.

In the method of enforcing internal formulas it has been proposed, in the learning process, adding additional relations between selected scales [7]. These relations can be very simple. In the simplest case, it is the sum of scales.

$$\sum_{w \in B} w = const. \tag{3}$$

Where B is the ensemble of selected for the „interlock" scales. Interlock word was used in quotation marks, because in reality one does not lock individual value of scales, and only their sum. So in the process of learning the different scales may change, however in the way that their sum remains constant - change then depends also on changes of other scale values in the group. Because of this in the solution space a hiperface is selected, on which the solution is being searched.

Adding the condition caused the necessity to modify the redundancy defining the change of a single scale. This condition can be taken into account by using the method of Lagrange'a multiplicators.

After the introduction of the condition [7] we get to solve the set of linear equations:

$$\begin{aligned} & w_{1,i+1}^{p} + w_{2,i+1}^{p} + \cdots + w_{n,i+1}^{p} = C \\ & w_{1,i+1}^{p} - \lambda = w_{1,i+1} \\ & w_{2,i+1}^{p} - \lambda = w_{2,i+1} \\ & \cdots \\ & w_{n,i+1}^{p} - \lambda = w_{n,i+1} \end{aligned} \tag{4}$$

Where $w_{1,i+1}^{p}, w_{2,i+1}^{p}, \ldots, w_{n,i+1}^{p}$ are searched values of the scales in step i+1, $w_{1,i+1}, w_{2,i+1}, \ldots, w_{n,i+1}$, scale values resulting from the classical method of error back propagation. The system can be easily solved, and the result gives new scale values.

Blocking the sum of the scales is not only possible to use redundancy between the scales. Another type of relation between the scales can be their product. In this case, however, to keep the flexibility and speed of resolution one should be reduced with blocking the scales up to pair in the ensemble B:

$$w_i^k * w_j^k = C_k \; where \; B_k = \{w_i^k, w_j^k\} \tag{5}$$

k - the number of another interlock and j k(w and k, in j l) is scale grouping in k-numbered relation. Therefore, in order to obtain new scale values in the next iteration it is necessary to solve ensemble k system of three equations. In every system there is one non-linear equation, the one with the interlock condition. On the other hand, systems of equations themselves are independent from one another, which is ensured by the divisibility of the ensembles B_k.

Interlock in the form of the sum allows for finding the minimum of hyper planes which are parallel to each other, while a multiplicative relationship changes the direction of the search, which may be advantageous for the targets set.

4 The ways of selecting the scales to „interlock" esembles

In the testing phase, various criteria of selecting the scales in the form of both the interlock of the sum and the products were examined. For the additive relations the first method was the selection of scales with the highest absolute value (maxAbs) [7], as the ones, that have quantitatively the greatest opportunity to influence the result of the networks performance. Resulting directly from the above method is the reverse method, which is the selection of scales with a value as close as possible to zero (minAbs).

The third and fourth method include scales, the change of values of which resulted in the largest and respectively the smallest, change of error on the result of networks performance in relation to the scales values (maxRatio [7] / minRatio). The tests used a threshold value δ by which the scales were changed, afterwards the full calculation was carried out for the learning data without modifying the values of scales. Obtained at the end root-mean-square error at the output of the network was divided by the value of the scale. One selected to interlock the scales, for which the so obtained value was the highest, or in the opposite method, the smallest.

Another method, not algorithmic anymore, was a manual selection of scales. It showed, that blocking all the scales in the neighboring neurons decreases the learning results.

For many „interlock" ensembles selection criteria were many groups related with one another, because in addition to the ranking of the groups algorithms of division of the scales for more ensembles were also required. On the other hand, it was necessary to examine into how many ensembles the selected scales can be divided, which resulted in the need for parameterization

of the scales finding including the number of groups algorithms. Selecting the groups process consisted of several steps:

a. ranking arrangement the relative scales relative to criteria for one interlock maxAbs / minAbs, maxRatio / minRatio
b. decision on the number of the groups and the number of scales in each group
c. the way of selecting scales for each group

The first step is analogical to the one with the selection of one group. The second determines the parameters with which we induce the third step algorithm. In the second point were tested:

- the number of groups equal to the number of classification groups
- two groups
- the number of interlock ensembles a grade larger than the number classification groups
- in the next stages of learning increasing or decreasing the number of groups

Having the scales sorted out and information about the number of groups constructed as a selection criteria of them into the respective ensembles. For every above mentioned case, the described experiments were checked for different amounts of scales in the interlock ensemble. Obviously in the presented method for product interlock, the number of scales in the group is stiffly set to two.

Having sorted out the weight and the information about the amount of groups a group selection criterion in the respective sets is constructed. These criteria may be analogical to those described earlier for the single interlock group, but a multiplicity of groups significantly broadens the possibilities of choice. And so the basic criterion maxRatio can be modified in a number of ways. The simplest way is to assign a certain amount of scales with the highest module to the first group, subsequent to the second and so on („main" assignment). Because of this the most important, regarding the determined criterion, scales are grouped together.

Another applied solution was assigning the scales to every group in order, first with the highest module to the first group, the next one to the second and so on. Thus, for example, with four groups into each group there will belong scales from every fourth position from ordered by selected criterion structure („proportionate" assignment).

Another modification was such the selection of the groups, that within a single interlock ensemble were the scales, which values of the applied criterion are in balance. This means that the scales were paired up, the one with the biggest and smallest value of the module („equilibrium" assignment). Analogously, one can select the scales according to other criteria.

5 The results of tests on exemplary classifying networks

The tests on the method have been conducted on the networks with the optimized structure for the task given, as well as on the redundant structure. In the first case, the difference between the non-interlock method, and the learning with enforcing the standard was not significant. However, at the stage of searching for the optimal networks structure, learning with enforcing the standard, noticeably improved the efficiency of learning. Networks with to big structure have decreased ability to classify data, absent in the learning ensemble, so they generalize problem worse [6]. Blocking changes this situation.

In both algorithms with blocking the sum and the product, learning effectiveness was highly dependent on the selection of the number of the groups, the scales in the group and the method of their selection. The tests were conducted in the following way: the network structure was being generated, which was afterwards taught with the standard method, and the same network with the same initial scales with enforcing the standard and with the same parameters, such as learning speed and momentum, but with different methods of enforcing the standard. The criterion of improvement was the number of identified samples from the test ensemble. Tests were conducted on the problem of classification of irises and the „Zoo" classification (based on 16 features the animals were divided into 7 groups), the classification of the glass (10 features, the classification into two groups). The test files contain respectively 45, 30 and 24 samples. Experiments for each set of blocked scales were repeated several times and the results provided in this thesis are the average of several tests for each of the networks.

For all the three classification questions blocking the set containing more than 90% of scales from the network grouped in one ensemble resulted in the network almost completely ceased to learn (Table 1). No effect of the interlock was observed with blocking about half of the scales from the network with all the possible methods. The network was learning comparatively to the absence of the interlock (Table 2).

Table 1. Number of well examined samples at the average for 15 learning attempts of redundant network with different speed and momentum parameters for different methods with the interlock of the majority of the scales in the network

	Irises	**Zoo**	**Glass**
Classic BP with momentum	41.3	22.4	20.7
maxAbs	13.8	11.4	10.2
maxRatio	12.2	18.2	9.8

Table 2. Number of well examined samples at the average for 15 learning attempts of redundant network with different speed and momentum parameters, for different methods with the interlock of the half of the scales in the network

	Irises	**Zoo**	**Glass**
Classic BP with momentum	41.3	22.4	20.7
maxAbs	41.1	21.8	21.6
maxRatio	41.2	22.8	20.8

Another attempt was based on increasing the amount of blocked scales, every 5000 iterations. With maxAbs approach no improvement of the network was observed. However, with maxRatio and classifying network for the problem of „Zoo" smaller influence of the learning speed selection on the networks performance has been observed. As far as with the classical method the number of identified samples ranged from 10 to 25 depending on the selected learning speed n and the momentum than with the same parameters for the enforcing the standard method the interval was from 20 to 25

This effect was observed as well in case when in the first step half of the scales in the network were blocked by selecting them using the maxRatio method, and in subsequent steps, their amount was reduced to half. This time the benefit was observed in all three questions. The best results were obtained by blocking half of the scales in the first learning cycle (selected by maxRatio method), and in the next steps the ensemble was reduced by removing half of the scales (Table 3). The achieved results were on average 8.7% better than the conventional method.

Table 3. The number of correctly identified samples on average for 35 samples of learning of the redundant network with different speed and momentum parameters, for different methods with decreasing number of scales blocked in the following stages

	Irises Stage I	**Irises Stage II**	**Irises Stage III**	**Zoo Stage I**	**Zoo Stage II**	**Zoo Stage III**	**Glass Stage I**	**Glass Stage II**	**Glass Stage III**
Classic BP with momentum	20.1	35.7	41.6	11.7	18.4	22.9	13.8	19.5	21.9
maxAbs	19.4	33.1	41.1	10.4	18.4	21.8	12.4	19.6	21.6
maxRatio	20.9	35.8	41.9	15.8	19.1	26.2	12.1	21.0	22.7

The next stage of the study was the selection of more than one group. The studies of this criterion for the sum had to be linked to choosing of the method of selecting the scales for the inter locks ensemble.

It proved that with moving the scales using maxAbs method and learning with many groups the criterion of scales selection has little importance. In the extreme case the completely randomly selected scales to three groups in the problem of irises achieved the results compatible with the best result from the selection using algorithms. The mean values of results of the experiments with maxAbs tables scheduling are shown in Table 4.

Table 4. The number of well-recognized average samples for 10 experiments of learning of the redundant network with different speed and momentum parameters, for different methods with blocking different numbers of groups of scales and max-Abs criterion together with sum blocking

	Irises	**Zoo**	**Glass**
Classic BP with momentum	40.2	23.1	21.4
maxAbs 2 gro-ups	39.1	23.4	22.1
the number of groups as the number of net-work outputs	40.2	22.7	20.1
Number of groups 10 or more	40.1	39.4	21.5

In case of selecting for interlocking the sum more than one group and ma-xRatio scheduling criterion in some ways of scales selection to the ensemble the results did not differ significantly from the selection of one group with the exception of extreme cases, where on one hand the total discrepancy of the network occurred followed for the problem of glass classification and almost perfect performance of the network for the classification of irises. In the worst case for the glass classification the network in 4 cases in the 15 experiments did not give any correct classification. In the best experiment with the classi-fication of irises the achieved result was equal to the performance of the op-timal network. The overview of the results is shown in Table 5.

Table 5. The number of well-defined at the average samples for 15 to learn a redundant network with different speed and momentum parameters, for different methods with blocking of different amounts of groups of scales with maxRatio criterion as well as interlock of the sum

	main assignment	**equilibrium assignment**	**proportional assignment**
Irises BP	41.2		
Irises 2 Groups	40.1	40.9	40.5
Irises 3 Groups	40.0	44.1	41.5
Irises 30 Groups	39.2	42.9	40.5
Zoo BP	22.9		
Zoo 2 Groups	21.6	21.5	23.8
Zoo 7 Groups	21.1	23.2	21.4
Zoo 70 Groups	18.9	22.4	21.5
Glass BP	21.1		
Glass 2 Groups	12.1	21.5	20.2
Glass 10 Groups	19.6	14.2	22.3

6 Summary

The enforcing the internal standards method, regardless of its form can improve the redundant networks chances in recognizing the data, which is not part of the learning subject. Although the network continues to achieve worse results than the optimal network, but they are very similar and in situations where the quick reaction of the neutron network is needed without the seeking of the optimal structure the described method can bring measurable time benefits.

References

1. Tadeusiewicz R., 1993, *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa.
2. Cybenko G., 1989, *Approximation by Superpositions of a Sigmoidal Function*, Mathematics of Control, Signals, and Systems, Vol. 2 ,pp. 303–314.
3. Rutkowska D., Piliński M., Rutkowski L., 1997, *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, Wydawnictwo Naukowe PWN, Warszawa.
4. Rumelhart D., Hinton G., Williams R., 1986, *Learning Internal Representations by Error Propagation*. Parallel Distributed Processing, Vol.1, pp.318–362.
5. Hornik K., Stinchcombe M., White H., 1989, Multilayer feedforward networks are universal approximators. Neural Networks, 2, pp. 359–366.
6. Rutkowski L., 2005, *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, Warszawa.
7. Kolibabka M., Cader A., 2006, Metoda wymuszania wewnętrznych wzorców w jednokierunkowej sieci klasyfikującej, Automatyka, 10, 3, pp. 497–502.
8. Haykin S., 1994, *Neural networks: A Comprehensive Foundation*, Macmikkan College Publishing Company, New York.
9. Bishop Ch.M., 1995, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, New York.

SPANISH SIGN LANGUAGE INTERPRETER FOR MEXICAN LINGUISTICS

Arturo Pérez

University ITS Chapala at Mexico
aperez@itschapala.com

Abstract

We present here the first visual interface for a Mexican Spanish Sign Language translator on its first development stage: sign-writing recognition. The software was developed for the unique characteristics of Mexican linguistics and was designed in order to use sentences or a sequence of signs in sign-writing system which are decoded by the program and converted into a series of images with movement that correspond to the Mexican sign language system. Using a lexical, syntactic and semantic algorithms plus free software such as APIss from Java, video converter software, data base manager like MySQL, Postgres and SQlite, was possible to read and interpret the rich and complex Mexican language. Our application for visual interface showed to be capable of reading and reconstruct each sentence used for the interpreter and translate it into a high definition video. The average time of video display vs number of sentences to interpret, probed to be in linear relation with an average time of two seconds per sentence. The software has overcome the problem of homonym words frequently used in Spanish language and verb tense relation for each sentence, special symbols such as #, %, $, etc. are still not recognized into the software.

Key words: Mexican Sign Language (MLS), spoken language translation, sign animation, sintactic algorithm

1 Introduction

The Sign Language is a system employed to stablish communication between persons with dis-capability both auditive and phonetic. A person with such dis-capability faces several obstacles while integrating into society. In order to overcome such difficulties (between a regular person and a deaf or deaf-mute person) Sign Language Interpreters softwares have been developed to attend this imperative need of communication. In the last two decades there has been more and more advances in visual interfaces for Sign Language Interpreters (Pardoa et al 2009, Halawani 2008, Dyng et al. 2008, Prada et al. 2008, Barra et al. 2007, Masakata 2006, Meurant 2004, Nyst 2004, Endbarg-Pederson 2003, Stouke 1960). The development of a Sign Language Interface

(SLI) strongly depends on the country where the Sign Language is used since not only the SLI is variable within different countries but also each country has its own characteristics for their official languages. It is well known that Mexico uses Spanish as its official language, however, the mexican linguistics (ML's) differ abruptly between the Spanish used in other countries such as Spain for example or the kind Spanish linguistics used in south American countries such as Colombia. This is why on the present date there are still many issues and misunderstandings towards persons who have a knoledge os Spanish different than the mexican linguistics and use it triyng to communicate in Mexico. Hence, translating Mexican text into Mexican Sign Language (MSL) requires a unique and special knowledge within this characteristic language. SLI's for Spanish language have been already produce for Rodriguez (1991), Prada et al.(2008) and Pardoa et al (2009), all of these works were produced only for European spanish linguistics and use 3D avatar technology to translate text into SL. To the date the are no works related to a Mexican Sign Language Interface (MSLI).

Producing a low cost software for Mexican language has been 47 a priority in Mexico ever since 2006, Mobil hardware devices have been created as SL Interpreter (Leybo´on et al. 2006), nevertheless, the software applied to this new mobil device was inefficient to decoding ML's, the system does not accounts for different hand positions, place of the hand gesture, hand direction and most important: face expression, which is one of the prevailing factors for a deaf-mute communication, since emotional expression plays a decisive factor on add meaning to each prhase for deaf-mutes (private communication Desarrollo Integral de la Familia, DIF, Jalisco).

This device (Leybo´on et al. 2006) was too robust and expensive for mass production. The consequences of not having such tools for deaf and deaf-mute persons has created an enormous incapacity of communication among the society added with discrimination factors towards the person with such discapability.

Another factor to be taken into account at the time of creating a new type of SLI, points at the unique cultural characteristics present in each country and then, with different priority of basic needs. The tools developed for our MSLI originate on the basis of two main priorities for mexican society: (a) the need to communicate from one person to another in order to obtain and provide one or more services and (b) the need of the person with a dis-capability to have a meaningful response to this communication. On the latter issue, mexican persons with discapability find it more meaningful to interact with a video image that can show up a sensible emotion than a 3D avatar.

While 3D avatars can indeed be constructed to represent an emotion while displaying, it has been prooved for local experts on SL (Desarrollo Integral para la Familia, Jalisco, private communication) that Mexicans do not relate well with virtual images while trying to communicate a feeling or necessity.

At the same time we do not discriminate the advantage of having 3D avatars for the case of simple written instructions or web page translation in a human-machine or in a more general perspective: human-object interpretation, here, Prada et al. (2008) offers the best deal for the interaction when communication 74 between people it is not a prime manner or can be avoided.

As an overall picture, the development of the MSLI does not involves any new relevant work in the rea of signal processing, the algorithms used in this work remain the same known at date, however, this is a work that focuses on the development area of engenieering, meaning a new practical and inovative tool devoloped to meet current social needs in the mexican society. The outline of this paper is presented as follows: section 2 describes the material and methodology employed inorder to explain the differences of an SLI for mexican language.

2 Methodology

In order to have an appropriate translation from speech transcriptions into SL it is necessary to have a parallel corpus institution to fit the translation models, test them, evaluate them and have them corrected in each phase of the process. In our case, the development of our MSLI was performed with the aid of a well founded federal institution: Desarrollo Integral para la Familia de Jalisco (DIF, Jalisco), which is a solid foundation institution in Mexico dedicated to aid persons with this type of discapability (among other functions in its primary activities). DIF, Jalisco, provided us with the unique opportunity to work with several experts (making a total group of 8 instructors for MSL) in order to test and apply the MSLInwith their respective students. This advantage gave us the opportunity to attend to the basic and real needs for a deaf or deaf-mute person in our society.

At the time of working with this corpus it was noticed that a 3D avatar would not be helpful at adressing person-person translations, where, a 3D avatar is more suitable to adress a human-object translations in a good feasible way. A mayor emphasis was pointed from DIF Jalisco over to a deaf-mute person having a real meaningful communication, hence, the need of a consistent translating system involving human facial expression. In this manner, itwas choosen the use of interactive videos as the apropiate way 99 of providing such answer.

Table 1. Software and Hardware

Employed software	Employed Hardware	Sampled Group
APIss (Java)	video camera	MSL students (7 children)
video converter software		MSL teachers (3 adults)
data base manager MySQL	keyword	
Postgres SQlite	screen	

The development of the platform was divided into four main stages: character recognition modulus, the syntactic modulus, the semantic modulus and the syntax modulus, these stages explain a sufficient coordinated method and a good efficiency degree of quality.

Figure 1. From left to right: (a) SLI Man window. (b) Interpreter screen, (c) Introduced sentence for SLI. The window of the SLI contains a text field were the sentence is introduced, this sentence must be written in simple tenses, the system accepts lower and upper case text.

The first stage (Figures 1,2) is the character recognition process performed with a lexical algorithm. The design of our lexical algorithm consist on a double buffer compiler system. For the double buffer system we have taken the couple used by Aho et al.1990.

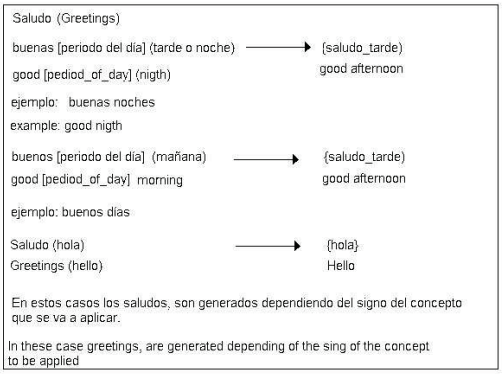

Figure 2. First type of simple sentence, greeting.

The lexical analysis consists on the identification of the word or sentence and the elimination of not usable words for the MSL. Within this process the text to be translated is first captured on screen using a basic keyboard. Afterwards, the system indicates to the user if any invalid character has been typed, the invalid characters in this stage are #, %, ¯ , $, ,´ .¨ , etc. Once all characters are read , the process of translation is allowed to continue only if all characters are recognized as valid characters. If the program finds one invalid character the translation process stops and a warning message is displayed on screen asking for the text to be retyped.

The second stage consists on the syntactic analysis. The algorithm applied was the normal form of Chomsky's algorithm (Chomsky 1965) this process consists on the identification of the sentence structure: subject, predicate, nouns, conjunctions, verbs, transitive verbs, intransitive verbs, complements, adverbials, and the use of the tences on each verb in the sentence (present, past, and future tences). During this process words like articles or conjuntions in the sentence are eliminated (since sign language does not uses any of those). Once the structure of the sentence is analyzed the process is allowed to continue (Figures 3,4).

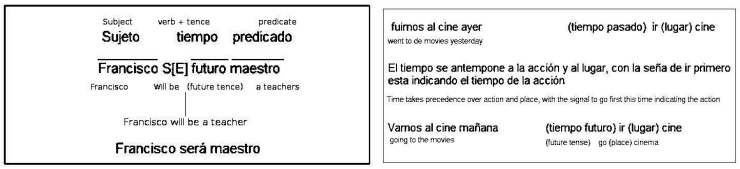

Figure 3. Construction of a sentence using simple tenses.

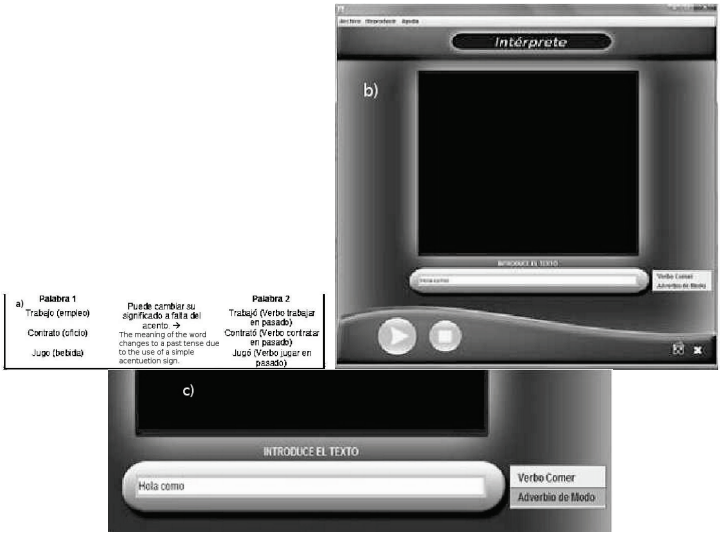

Figure 4. From left to right: (a) Change of meaning in the word according to accentuation sign. (b) current banner for unknown sign, (c) Sentence selection in the case of homonym and homograph word. In (b) the SLI system is programed to detect homonym words on the database by showing an current banner. The desired meaning for the sentence or word is given to be chosen afterwards.

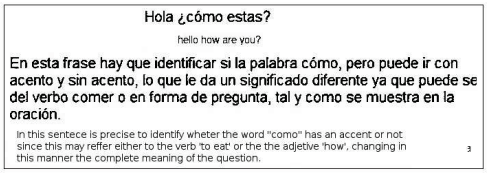

Figure 5. Changes in the meaning of a sentence depending on grammatical sign applied in the sentence.

Figure 6. Form left to right: (a) Error window. (b) Correction Screen, (c) Sentence. The SLI system does not adtmits signs such as $,%,&, dot, colon. If the user types an invalid character the systems displays a message on (a). The system can is able to interpret no more than 12 digits and the structure of this should be in a continuous form (without separations or blank spaces).

On the third stage we find the semantics analysis, this process uses the algorithm to arrange the main sense of the sentence and eliminate any ambiguous senses. The main objective of this process is to identify either homonym and homograph words (words that sound or are writen in a similar way such as casa or caza, both existent on the mexican language and with different meanings depending on the form they are aplied in each sentence). This stage was completed at a 70% rate due to the complexity of the algorithm, in order to complete this stage an emergent banner appears at the time the user uses an homograph or homonime word, the banner shows up a menu with options for different meanings and gives the user the option to choose the more convenient one.

The last stage of the SLI is the generation of the sequence of images selected from a re spective database of the sign language. Once the semantics is produced, the syntax algorithm(tal de tal ao) simply uses the before structure to select and display the appropriate video providing then the expected translation.

3 Results

Our main results are described as presented in the methodology. In the first stage we obtained the products of sentence typing as well as the voice recognition process, it was mesured that this process took an average time of 2.5 seconds per recognized sentence. The average time of pattern recognition is faster when the sentence is only written instead of spoken, this caveat waves on the fact that voice capture processes are still not fully undestood and keeped yet under development.

For the second stage of the MSLI it was possible to identify subject, predicate and verb within the sentence's grammar and distinguish between the present, past or future tences for the verb (since most of SL does not uses verb tences) at this point it was also possible to eliminate other words not used by MSL such as conjunctions or intransitive verbs. As shown in Figure 4 the recognition process of the SLI produces a screen for the video sequence and a space at the bottom to show the written or spoken sentence. This typed sentence either, follows or it is made to adapt to the grammatical order of Mexican linguistics (noun, verb, predicate, et) in order to continue the translation. The process aldo stops when a unknown symbol is typed or a unrecognized word is spoken (such as caaa instead of casa).

Afterwards, the MSLI produces the correspondent video in the database and differentiates similar sentences with similar words but different gramatical puntuation (such as the spanish mexican accent, wich then adds a total different meaning to the sentence).

The semantic modulus was completed at a 70% due to the complexity of the algorithm dealing with homograph words. In order to solve complete such inconvenience a recursive help banner was applied, every time the user types an homograph word a graphic menu appears with several options to choose from. Once the convenient option is chosen by the user the interpretation process continues and the video is displayed. For the case of compound sentences the SLI produces only a single video to show with the main ideas of the compound sentence.

Within our results it was also created a small buffer during the process of lexical analysis. This buffer is a thread manager created to storage words which might be captured with voice recognition in the implementation of future hardware device for sound recognition, not yet employed at this stage of the SLI but left aside in order to focus only on this first stage of SLI, written sentence recognition transformed into video image. Finally it was noted that the quality of the final videos shows a small flash in between videos this caveat is due to the multimedia java version employed for the development of MSLI.

4 Discussion

The obtained results with the SLI were taken to DIF Jalisco and it was proved how the general needs demanded at time were mostly covered. Regarding the employed algorithms used in this SLI: lexical, syntactic and semantic, it is plausible to modify and generate the current syntactic and semantic algorithms employed here as mentioned by Montero (2004) and Earley (2001). As for the lexical algorithm it is now currently used the standard algorithm of Tokens since it only consists on pattern recognition and there are not new and specifics needs that reacquires to employ or perform any kind of modification to it.

An important fact to be addressed is our use of video images instead of a 3D avatar such as the case of Prada et al. 2008 or Halawani 2008. This difference creates both advantages and disadvantages regarding to the area of application for the SLI, the fact of having a 3D avatar results very convenient at the time of simple interaction for instructions or guidelines using tools such as web pages or any other human-machine interface, nevertheless the 3D avatar faces limitations at the moment of human-human interactions where the occurrence of of facial gestures takes a mayor role to be taken into account in order add meaning to the conversation for a person with this type of discapability. Adding more detailed expression to this caveat has a high cost in development but if such improvement could be achieved this type of tool on the SLI could have a mayor impact on many areas of interaction for a deaf-mute person.

The fact of a deaf-mute interacting with another person creates 186 a basic need to receive some type of facial gesture in each sentence in order to have a complete meaning of the conversation. In this case, the turn point comes into crating a complex conversation, hence, having a complete sequence of images to follow up such type of interaction, the use of video images faces such difficulty and it is been left as future work.

During the process of translating a sentence into image the average time resultant in each process (2 seconds) clearly indicates that each algorithm and the employed process produces a reliable and consistent response that connects with sufficient feasibility the continuity of the translation process as such. The relation of the main variables with time (Figure 1) and between, presents a linear correspondence. This linear correspondence is the mathematical proof of how each one of this variables has an independent role in the process besides that the orthogonality between them it is also a way to describe the evolution of the system.

As for the flash seen between images for translated sentence it was noted that such effect was due to operative system and its different versions. This flaw is strongly seen for Windows Operative System (OS), specially, Windows Vista and Windows 7. The effect greatly diminishes at Windows XP,

while for Mac OS there is no appearance of such flash. We strongly believe that such discontinuity is due to the Java Media Framework API for Windows which most probably needs a new compatibility upgrade for the multimedia version for Windows, which is not the case of the Mac OS.

The case of a SLI as a new tool for Mexican linguistics has the advantage of being the first tool in this country capable of reproduce and translate simple human-human conversations whereas it has been mention the appearance of an photo-electric sensor for hand movement as a SLI tool (Leybon-Ibarra et al. 2006) this particular software uses a photo-electric electrode system within an adaptable glove with 3D avatar screen images to produce the translation, nevertheless, the main limitation of such system consists in the incapability of freedom of hand movement, this is: on hand direction, orientation, crossed or bended finger positions and facial expression.

5 Conclusions

The new SLI for Mexican language was presented in this article, our main results showed to be reliable and satisfactory within the selected proof sample which were selected within the current needs of the actual MSL for Mexicans and specially aimed at the expressed needs of DIF Jalisco. The program showed to be capable of overcoming many MSL difficulties such as homograph and homonym words to construct a sentence.

Our visual interface showed to be capable of reading and reconstruct each sentence used for the interpreter and translate it into a high quality video. The average time of video display vs number of sentences to interpret, took an average time of 2 seconds per sentence and probed to behold within a linear relation, which shows how both variable are independent between them (hence able to describe different characteristics of the evolution of the system).

Nevertheless, some caveats are still to be considered and investigated such as the use of special symbols such as #, %, $, etc. which are still not recognized into the software. The use of complex compound sentences are still yet not recognized by the SLI and has to be taken into account for more elaborated dialogues. More work for facial gesture recognition has to be done since most of SL persons uses them to change the meaning of each sentence and it is also a way to recognize the meaning of a phase used to communicate with them. In the mean time, the use of SLI for an average conversation and average needs of a deaf-mute person with a regular non-sapient MSL person is now covered and capable to perform its main task.

References

1. Aranda., B. E., 2008. La vulneracin de los derechos humanos de las personas Sordas en Mxico. Comision Nacional de los Derechos Humanos, CNDH.
2. R. Barra, R. Crdoba, L.F. Haroa, F. Fernndeza, J. Ferreirosa, J.M. Lucasa, J. Macas-Guarasab, J.M. Monteroa and J.M. Pardoa, *Speech to sign language translation system for Spanish*, Aplied Soft Computing, 2008.
3. Comparn, J. J., 1999. Lengua Espaola I. Mxico: AMATE. Discapacidades, E. C. *La sor dera y la prdida de la capacidad auditiva.,* http://www.sitiodesordos.com.ar/sordera.htm.
4. Ding Lilia, Modelling and recognition of the linguistic components in American Sign Language, Aplied Soft Computing, 2008, 421, 105.
5. Dons, R., & Ortz, C., 2005, XXXV *Simposio Internacional De La Sel*: http://www3.unileon.es/dp/dfh/SEL/actas.htm.
6. J. Earley, *An Efficient Context-Free Parsing Algorithm, PhD tesis,* University of California, Berkeley, California, 1970, pp. 94-102, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/cmt-55/lti/Courses/711/Class-notes/p94-earley.pdf.
7. El Universal. 2006, : http://www.eluniversal.com.mx/articulos/30484.html.
8. Estrada, B., 2008, Sordos: www.sordos.org.mx/articulo.doc.
9. Galicia, S. N., 2000, Instituto Politcnico Nacional Centro de Investigacin en Computacin Laboratorio de Lenguaje Natural,Anlisis sintctico: http://www.gelbukh.com/Tesis/Sofia/tesisfinal.htm.
10. Garca, J. R., *& Giner*, B., 2007, Pearson, Prentice Hall.
11. Leybon I. J, Ramirez B. M.R., Picazo T. V., Photo-Electric Sensor Applied to Hand Fingers Movement, Computacin y Sistemas, 2006, 10, 556.
12. Lodares, J. R., *Aplicaciones Lexemticas a la Enseanza Del Espaol*, 2009, Clarn, Revista de Nueva Literatura, 78.
13. J. M. Montero M., *Desarrollo de un Entorno para el Anlisis Sintctico de una Lengua Natural,* Universidad Politcnica de Madrid, Espaa, 2004, http://lorien.die.upm.es/juancho/pfcs/JMMM/pfcjmmm.pdf.
14. Nuno, R. (1998). Correlatos neurofisiolgicos del lenguaje de senas en el nino sordo. Proyeto de Investigacin.
15. J.M. Pardoa, J. Ferreirosa, V. Samaa, R. Barra-Chicotea, J.M. Lucasa, D. *Snchezb and A. Garcab Spoken Spanish generation from sign language*, 2009, Aplied Soft Computing, 123.
16. Dr.Sami M.Halawani, *Arabic Sign Language Translation System On Mobile Devices*, International Journal of Computer Science and Network Security, 2008, 8, 1.
17. Suphattharachai Chomphan, Towards the Development of Speaker-Dependent and Speaker-Independent Hidden Markov Model-Based Thai Speech Synthesis, 2009, Journal of Computer Science, 5, 905.