Exploring of Sliding Window Visualization System to Understand Flow and Error Control Mechanism of Data Link Layer

Ayushi Chaudhary

Abstract- The perspective of this paper is to provide user friendly Visualization System for Sliding Window protocol. The Sliding Window Visualization system (SWV) is designed to understand the flow and error control mechanisms of DLL (data link layer) protocols. The system is interactive and allows the user to modify some parameters of the protocol. In this paper a visualization system has been developed where a user can easily understand the working principle of sliding window protocol and it can be used to compare two algorithms. In the mean time except the visualization of this network protocol also we have sent data packets at the back end. Thus giving an opportunity to the user to understand the mechanism of real time data transfer, where communication is often possible between physically separated machines. The software has major responsibility is to help to visualize newly developed algorithms where a researcher formulates some mathematical model of an algorithm and a developer converts it into a visualization system. This let us to compare two algorithm's efficiency with respect to some parameter. This paper is designed and developed in such a manner that it provides a vast scope of further development. Number of modules can be added without many modifications in it code for new algorithm and to compare their efficiency with respect to existing algorithms. Since, this visualization system has been designed by taking care of the needs of users, their tasks. This SWV can be used as a teaching tool for a term at the community college level. SWV will be useful in a laboratory or self-study situation after the student has been introduced to DLL protocols. SWV's strong point is in helping to create mental images of the protocol mechanisms, and in allowing easy and painless experimentation with the supported protocols.

Keywords: Sliding window protocol, Interactive animation, visualization

I. INTRODUCTION

Advances in computing technology and the affordability of software and high performance graphics hardware enabled rapid growth of visual tools. Today, not only very expensive workstations, but also low cost PCs are capable of running computationally demanding visualization systems. Visualizations or the graphic depictions in execution are being used in explaining, designing, analyzing algorithms, and in debugging, fine-tuning, and documenting programs. Although many tools have been developed over the past twenty years, little attention has been paid to the analysis of users, their needs, tasks, and goals. So our paper focuses on visualization and Simulation of sliding window protocol. This protocol is one of the most widely used protocols in the field of networking.

In this paper we have developed a visualization system where a user can easily understand the working.

Principle of sliding window protocol and it can be used as Simulator to compare two algorithms. In the mean time except the visualization of this network protocol also we have sent data packets at the back end. Thus giving an opportunity to the user to understand the mechanism of real time data transfer, where communication is often possible between physically separated machines, over so called ―faulty channels‖ — i.e., an asynchronous channel that can lose and duplicate messages and can receive messages in an order different from the one in which the messages were sent. So, Sliding Window protocol enables the reliable transfer of a data stream from one machine to another machine over a faulty channel, i.e., the receiving side outputs a data stream that is equal to the data stream input on the sending side. Furthermore, the protocol should not hamper throughput unnecessarily, the protocol must be provably correct, and the protocol must exhibit progress.

The Sliding Window Visualization system (SWV) can be designed to understand the flow control mechanisms of DLL (data link layer) protocols. The system is interactive and allows the user to modify some parameters of the protocol. The main goal was to create an interactive simulator where we can visualize the working of an algorithm i.e. sliding window algorithm which is widely used in many standard network protocols. I am going to implement a sliding window protocol which is a bidirectional protocol, at any instant of time; the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window and the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept.

The Sliding Window Visualization system (SWV) is designed to help students understand the flow control mechanisms of DLL protocols. The system is interactive and allows the user to modify some parameters of the protocol
and to create events while the protocol is being visualized. SWV was designed to meet the following goals [2]:

- Animated and Interactive – student watches the protocol in action
- Protocol parameters can be modified
- Protocol events can be generated at run-time – student watches protocol reaction to events
- Easy to use, platform independent – student may only spend a few minutes with SWV, so it has to be simple enough to provide immediate benefit

Revised Version Manuscript Received on August 31, 2015.
Ms. Ayushi Chaudhary, Student, Department of Computer Science & Engineering, Marathwada Institute of Technology, Bulandshar Uttar Pradesh, India.

II. LITERATURE SURVEY

A. Data Link Layer Protocol

The purpose of the DLL in computer networks is to convert noisy lines into channels free of transmission errors for use by the network layer [3]. Data is segmented into frames, each of which is transmitted until properly received. Flow control mechanisms are provided by DLL protocols to prevent a faster sender from swamping a slower receiver. This section provides a brief description of the established DLL protocols, which form a basis for the work of this paper.

Unrestricted Simplex Protocol (utopia)

This is the most simple DLL protocol possible. This protocol assumes that the communication channel is error free, and the receiver can process the data infinitely fast

Simplex Stop-and-Wait

With simplex stop-and-wait, the sender must be prevented from flooding the receiver with frames, since it is unrealistic that a slower receiver could keep up with a faster sender. The general solution to this problem is to have the receiver return an acknowledgment (ACK) frame to the sender, in essence giving the sender permission to transmit the next frame. An enhancement to this protocol is to add a 1-bit sequence number to the data frames and acknowledgments. This enhancement to the protocol is called Positive Acknowledgment with Retransmission (PAR - protocol 3 in [3]). The PAR protocol is the version of Simplex Stop-and wait that is supported by SWV.

Sliding Window Protocols

Sliding window protocols (SWP) all use full-duplex data transmission. The simplest way of achieving full-duplex data transmission is to have two separate communication channels; a forward channel for data and a reverse channel for acknowledgment frames. This is the full-duplex situation supported by SWV. In all SWP, each outbound data frame contains a sequence number, usually 0..2n-1 so the sequence number can fit into an n-bit field. The sender contains a list of (consecutive) frames it is allowed to send, which are called the sending window. The receiving window is a similar set of frames the receiver is permitted to accept. The sending and receiving window need not be the same size, and SWV preserves this flexibility. The sequence numbers in the sending window are frame numbers that have been sent but not acknowledged. The upper edge of the sending window is advanced with new packets arriving from the network layer. The lower edge of the sending window is advanced when an acknowledgment arrives for the oldest outstanding sent frame. The sender must buffer unacknowledged frames for possible retransmission. The receiving window is the list of frames that may be accepted, and any frame which arrives outside the window is discarded. The receiving window always remains its initial. size, and when the frame whose sequence number is on the lower edge of the window arrives, the receiving window slides forward. Receiving windows of size > 1 must contain buffers equal to the window size, since the receiver's DLL must forward frames to the network layer in the proper order.

One Bit Sliding Window

This is the most basic SWP with a window size of one for both sender and receiver.

Go-Back-n Sliding Window

Go -back-n uses a technique called *pipelining* in which a sender continuously transmits frames for a time equal to the frame round trip transit time. This technique does not require the sender to wait for an acknowledgment before sending the next frame. When a frame in the middle of this stream of frames is damaged or lost, the receiver simply discards all subsequent frames, sending no acknowledgments. The receiver refuses to accept any frame except the next one it must forward to the network layer (receiver window size=1). Eventually, the sender times out and resends all unacknowledged frames in order, starting with the lost or damaged frame.

Selective Repeat Sliding Window

Selective repeat uses pipelining in the same way as go-backn, but the receiver stores all correct frames following the bad one. The sender times out and begins retransmitting the unacknowledged frames in order again, as in go-back-n, but
when the lost data frame is received, the receiver can acknowledge all frames present in its buffer. This corresponds to a receiver window size greater than 1. Go back-n and selective repeat are trade-offs between bandwidth and receiver buffer space. Both of these protocols require separate sender timers for each frame.

B. How Dll Protocols Lend Themselves To Visualization

DLL protocols can be described in the following ways:

- Pseudo code
- State machines
- Sliding Windows Diagrams
- Time -based frame transmission diagrams
- Performance formulas

With these several ways to describe and define DLL protocols (most of them visual), there are generous opportunities to combine diverse representations into a visualization system. SWV does not use pseudo code or state machines to visualize DLL protocols, although they could be used in a visualization system. Sliding windows diagrams are an abstraction used to convey the current state of a sliding window protocol. The sliding windows in SWV are actively animated as protocol events occur, and are a vital visual element in assisting the student to understand the current state of the sender/receiver windows. Time -based frame transmission diagrams are brought to life with SWV. Instead of imagining the progression of time. Through the diagram, SWV animates these diagrams and allows the user to affect or realize the diagram by damaging or losing frames as the protocol proceeds.

III. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new

system and in giving the user, confidence that the new system will work and be effective. Descriptive life cycle models characterize how software systems are actually developed.

A. Supported Protocols

of the protocols mentioned in section 3, most can be visualized using SWV:

1) Go Back N Visualization
2) Selective Repeat Visualization
3) Simulation with data Transfer for Selective Repeat ARQ

We found it more interesting to concentrate on the complex protocols, since these are generally more difficult to understand [2]. Protocol 1 is merely a building block to understanding DLL protocols and it was thought that protocol 3 is so closely related to protocol 2 as to make protocol 2 support uninteresting.

B. Design of Sliding Window Visualization

At the core of SWV is a framework which allows a protocol to control the animation and feeds user input to the protocol. Protocols can be modified, added, or removed, independent of the framework [2]. The protocol itself can optionally use sliding windows, and can have a fixed maximum sequence or window size. When SWV was designed the graphical display layer was separated from the underlying framework of protocol and visualization system core (Figure 1). This facilitated a cleaner design and Object Oriented implementation. The graphical display layer includes a sliding windows component which is encapsulated and could be used by another Java application needing a sliding windows GUI control.

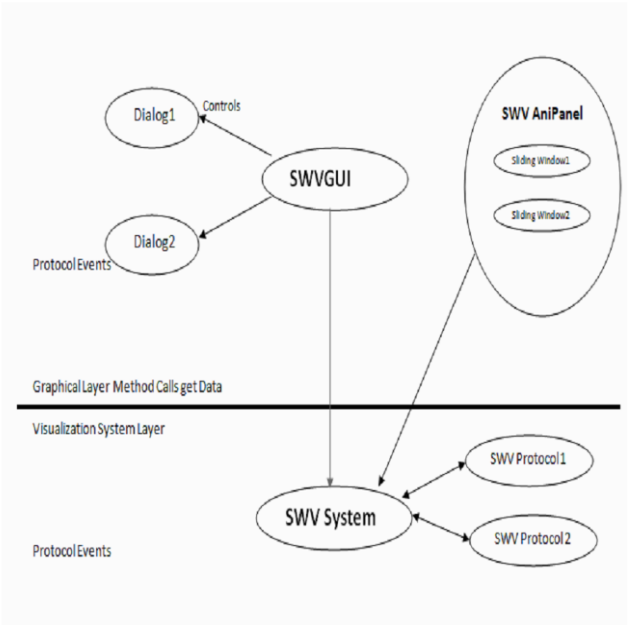

Figure 1: Architecture of SWV

C. Interaction Levels

Interaction is possibly the most important issue a visualization system designer faces. Interaction is what differentiates a visualization system from a simple animation of an algorithm or protocol. A visualization system should encourage a user to test a hypothesis through interaction. A interaction level should be simple as possible use case diagrams figure 2 as shown to the user and swv. How to get the packet transmitted and received to packet. During the connection establish the client and server. Given the parameters user and swv how many packets are received and discarded information.

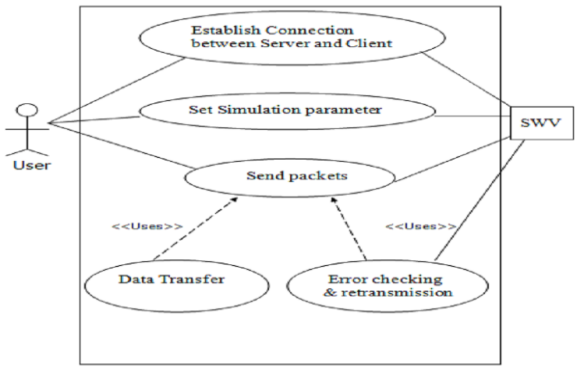

Figure 2: Uses Case Diagrams

D. Activity Diagram for the Client

In Fig3 I can show overall activity diagram for the client as follows

- Request for IP address
- Establish the connection
- Connection is established to receive the data
- Connection is not established to display the error message
- Send the acknowledgment number and display msg
- Send the data reordered simulation result.

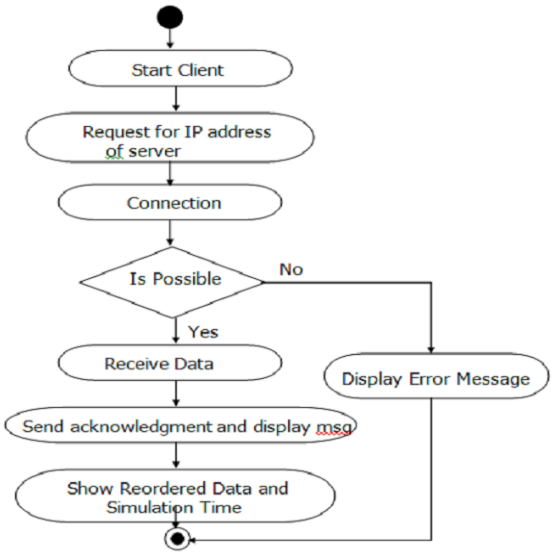

Figure 3: User Activity Diagram

E. Display Out

Sami Khuri comments on the design of visualization system interfaces [4]: "Interactive visualization systems should be forgiving to the user. In a highly-interactive system, there is always a situation when the user will press the wrong button, input invalid data, or manipulate the wrong graphic object. The vast majority of user errors occur because the developer of a system allows the error to occur. Most error messages therefore, can be eliminated by reducing the possibility of errors, by making sure that number fields only accept numbers, but providing lists wherever possible, by providing file selection dialogs rather than asking users to type filenames, and by providing default

values." If you choose Simulation with data Transfer for Selective Repeat ARQ, the sever starts execution and waits for connection with some client.

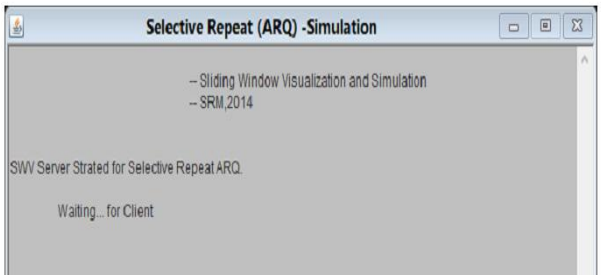

Figure 4: Server side snapshot when the server is waiting for client

On Client machine run the SWV Client Application (here it is SWVClient2) given with the package. When the client starts it requires the IP address of Server machine.

Figure 5: Client side snapshot when the user enters ip address of server to connect

After getting IP address the connection between Server and Client machine is established.

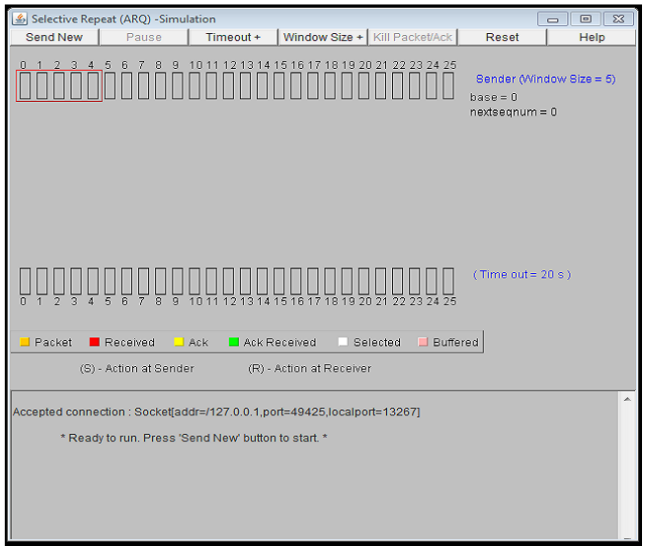

Figure 6: Server side Snapshot when the server is connected to specific client.

First of all take for 3 or 4 frames given the menu option send new, timeout, window size, kill packets, reset etc. Here packets 3 and 6 have not been acknowledged and treated as packet lost or Ack lost and shown in specific color (light blue) in sender window because the SWVClient2 has dropped packets 3 and 6 and, in receiver side packet 4, 5 and 7 have been received but treated as out of order kept in buffer and shown in specific color (pink for Buffered packet) in receiver side. When timeout occurs sender checks its window for all packets has been acknowledged or not. The packets which have not been acknowledged will be sent by sender automatically after timeout. The Client side application is running that receives data and shows appropriate message while simulation as below.

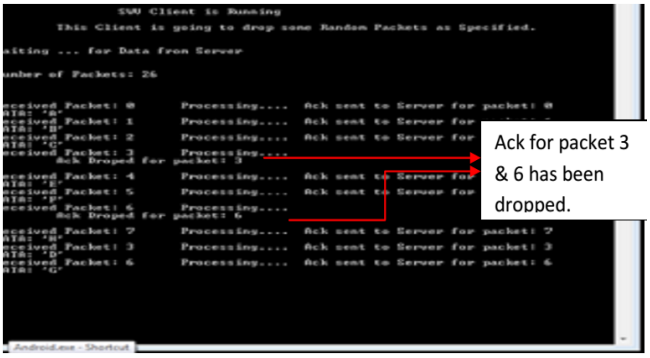

Figure 7: Client side snapshot when the client is receiving packet & dropping acknowledgement.

Here this Client (SWVClient2) has dropped packets 3 and 6 as shown and after timeout when sender sends it again it is received successfully. Different color used for specific packets and Acknowledgement as shown below:

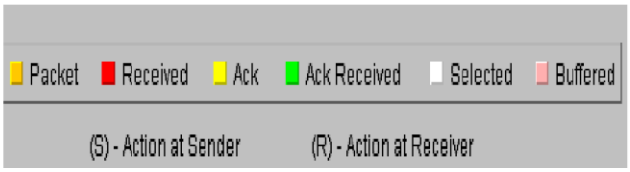

Figure8: Snapshot of different color and meaning

Orange - Moving Packet color.
Red -Received Packet at Reviver side.
Yellow - Moving Acknowledgement color.
Green - Packet Acknowledged at sender side.
Pink - Buffered packet at receiver side.

In the below snapshot it is shown the Acknowledgement for packets 20,22 and 23 has been received at sender side. Packet 24 is an the way.

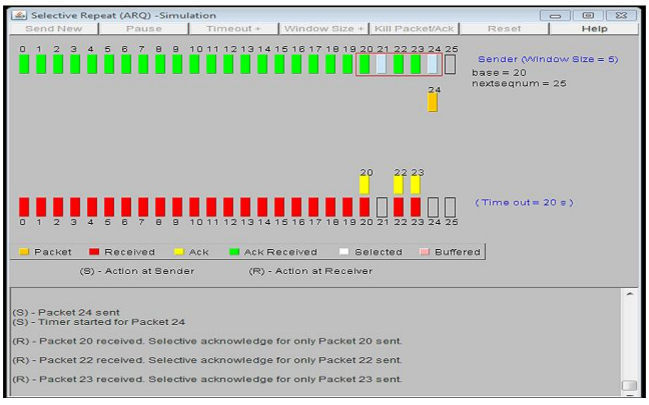

Figure 9: Server side snapshot when the server sending the packet

When all data has been received by client it shows all data in order and total time taken for data transfer as simulation time.

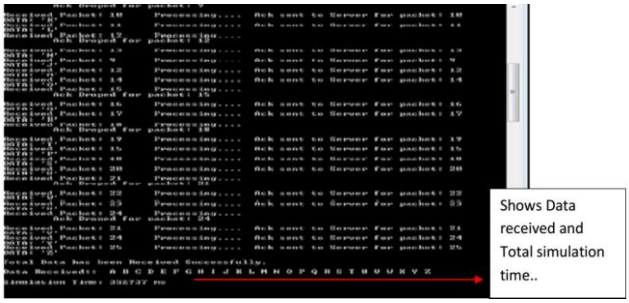

Figure 10: Client Snapshot when simulation is complete.

IV. FUTURE ENCHANCEMENT

Many visualizations are being designed without paying too much attention to the needs of users, their tasks, and their characteristics. Creating visualizations requires substantial time and effort. Mapping an algorithm to an animated representation is a non-trivial problem; it requires careful thought and knowledge of a particular algorithm animation programming framework. There are many underutilized techniques for creating effective visualization, such as sound, 3D, and artificial intelligence, which will continue to spark the creativity of the developers in the years to come. But the main problem still remains the same, there is no single visualization, specialized or general-purpose, that can satisfy all kinds of users, all kinds of tasks and can be used in all kinds of environments. Without the careful analysis of the users‘ needs, tasks, scope, information, and resources, the effort put in developing a visualization package will probably be wasted. It can help to visualize newly developed algorithms where a researcher formulates some mathematical model of an algorithm and a developer converts it into a visualization system. Since, this visualization system has been designed by taking care of the needs of users, their tasks. This SWV can be used as a teaching tool for a term at the community college level. SWV will be useful in a laboratory or self-study situation after the student has been introduced to DLL protocols. SWV's strong point is in helping to create mental images of the protocol mechanisms, and in allowing easy and painless experimentation with the supported protocols. It will be more intuitive to some students than discussing slides of sliding window diagram progressions. Viewing and interacting with an animation can provide significant intuition about the behavior of a protocol. Proper abstraction of protocol elements can be used to filter out extraneous information, to draw the user's attention, and to efficiently convey information.

V. CONCLUSION

The concept of peer-reviews helped to rectify the problems as and when they occurred and also helped us to get some valuable suggestions that ware incorporated by us. Developing those has helped us to gain some experience on real – time development procedures. SWV has been used as a teaching tool for a term at the community college level. Only a handful of students were involved in evaluating SWV as a teaching tool, so there was not enough data to perform statistical analysis, but useful comments were collected which helped direct improvements in usability and usefulness as a learning tool. [9] The authors recommend using SWV in a laboratory or self-study situation [10] after the student has been introduced to DLL protocols. SWV's strong point is in helping to create mental images of the protocol mechanisms, and in allowing easy and painless experimentation with the supported protocols. The author has also used SWV in the classroom while teaching DLL protocol concepts, which seems more intuitive to some students than discussing slides of sliding window diagram progressions. However, the authors stress that this use of SWV should *not* replace discussions of DLL protocol code and algorithms.

REFERENCES

1. Lawrence, A.W., Badre, A.N., & Stasko, J.T., “Empirically evaluating the use of animations to teach algorithms”, Proceedings of the 1994 IEEE Symposium on Visual Languages IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 48-54.
2. David Henery, Yashwant K. Malayia, “A Visualization System forSliding Sliding Windows Protocols” Proceedings of the 2003 IEEE frontiers in Education Conference Boulder, CO ppT2C1-T2C6.
3. Kehoe, C.M., & Stasko, J., “Using Animations to Learn about Algorithms: An Ethnographic Case Study”, Georgia Institute of Technology Technical Report GIT-GVU-96-20., 1996.
4. Tanenbaum, A.S., Computer Networks, 3rd Edition, Prentice Hall PTR, 1996, pp. 212-250.
5. Khuri, S., “Designing Effective Algorithm Visualizations”, Available at http://www.mathcs.sjsu.edu/ faculty/khuri, 2001.
6. Brown, M.H., & Herschberger, J., “Color and Sound in Algorithm Animation”, DEC Systems Research Center Technical Report, 1991.
7. Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P., Glaser, R., “Self Explanations: how students study and use examples in learning to solve problems”, Cognitive Science, #13, 1989, pp. 145 -182.
8. Price, B.A., Baecker, R.M., & Small, I.S., “A principled taxonomy of software visualization”, Journal of Visual Languages and Computing 4, 1993, pp. 211 -266.
9. Cox, K., Roman, G., “Abstraction in Algorithm Animation”, Proceedings of the 1992 IEEE Workshop on Visual Languages, 1992, pp. 18-24.
10. Henry, D., “Master’s Project: A Visualization System for Sliding Windows Protocols”, Colorado State University Technical Report, available at http://www.cs.colostate.edu/testing/, 2002.
11. Hundhausen, C., “Toward effective algorithm visualization artifacts: Designing for course”, Doctoral dissertation, University of Oregon, 1999.
12. Hansen, S.R., Narayanan, N.H., & Schrimpsher, D., “Helping learners visualize and comprehend algorithms”. Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 2000.
13. Stasko, J., Badre, A., & Lewis, C., “Do Algorithm Animations Assist Learning? An Empirical Study and Analysis”, Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems, ACM Press, New York, 1993, pp. 61 -66.
14. Lattu, M., Tarhio, J., & Meisalo, V., “How a Visualization Tool Can Be Used - Evaluating a Tool in a Research & Development Project”, Proceedings of the 12th Workshop of the Psychology of Programming Interest Group, 2000.
15. Douglas, S., McKeown, D., & Hundhausen, C., “Exploring Human Visualization of Computer Algorithms”, Graphics Interface ’96, 1996, pp. 9 -16.
16. Michail, A., “Teaching Binary Tree Algorithms through Visual Programming”, University of Washington Technical Report UWCSE-97-05-01, 1996.