**Seminar of I. Vekua Institute
of Applied Mathematics
REPORTS, Vol. 30, 2004**

ABOUT SOME PROBLEMS OF MODERN PROGRAMING

Veliashvili N., Jibuti M.

I. Vekua Institute of Applied Mathematics,
I.Javakhishvili Tbilisi State University

Received: 29.09.2004; revised: 22.12.2004

Abstract

In this article some problematic aspects of Client/Server technology, Object-oriented Programming, Visual Programming, and Event-Driven programming are considered.

Key words and phrases: Client-Server, Object-Oriented,OOP,Visual Programming, Event-Driven.

AMS subject classification: 68N19; 68N15; 68N01.

The methods of modern programming are moulding in the rapidly developing network environment. As is well known, Client/Server technology and Object-Oriented Programming are holding the front line in this field. Also, modern visual programming systems, which are depended on the Event concept, that is program's execution flow is controlled by the events. This method is known under the name Event-Driven Programming.

In this article we attempt to reveal some problematic aspects of these methods.

Client/Server technology arose in the mainframe era [1,2,3], when terminals were dumb. Therefore, data processing's every operation was performed by the central computer mainframe. With the advent of personal computers, i.e. when terminals became intellectual, the opportunity appeared to distribute data processing operations among central computer (Server) and intellectual terminals (Workstations). The general idea of distributed processing is the following:

a. Centralcomputer - Server contains Centralized Repository of Data.
b. Each Workstation - Client usually works with this Repository: it receives appropriate data from Server and processes them locally.
c. If necessary the result of data processing is sent back to Server.

Thus, Data direct processing is performed by a workstation, while server in such schema fulfils only functions of dispatching. Indeed, at first years of coming-to-be of personal computers networks, there was developed the File-Server Schema, which represents the maximalistic shape of Client/Server architecture. There was developed appropriate Operation Systems and NOVELL NATWARE was the best among them. About high level of Reliability and Performance of this product specialists know very well. Unfortunately, the File-Server schema experience has showed serious lacks of this approach. The main lack of this schema is just the fact, that Server does not execute any other processes except dispatching processes. On the other hand, any modern powerful database's functioning requires simultaneous execution of dozen nondispatching

processes. This means that powerful databases can't carry out full-scale functioning in the File-Server environment. In other words, the File-Server's capabilities do not meet the powerful databases' requirements. However, there were attempts to realize the powerful databases in the File-Server environment (e.g. ORACLE in the NOVELL NATWARE). Unfortunately, these attempts had no success.

Consequently, it is possible to make up the conclusion: for successful realization of Client/Server architecture a Server must be a mainframe type. It's obvious that an Operation System for such Server must be a corresponding type - it will be able to execute several programs and subprograms simultaneously, i.e. it must be a multiprogram and a multithread system.

Network Traffic problem is a well known one - a Network Admittance is significantly behind the computers productivity. Therefore, it is bottleneck of Client/Server architecture. Usually, this problem arises when network performs intensive information interchange and/or interchange covers a large amount of information. So, in some cases Network Traffic can reduce an application's serviceableness to zero, i.e. the main goal - source task may turn out practically unsettled.

In such situations, the recommendations are given to application developers, something like this:

a) If amount of receivable information is not large, then a workstation can receive and process it locally.

b) Otherwise, when amount of receivable information is large, information processing must be entrusted to a server using, so-called, stored procedures.

During the realization of such recommendations, often, information processing may be completely entrusted to a server, while for workstation is remained only interaction with users. Of course, in such allocation the main idea of Client/Server schema is disrupted, i.e. the idea about distributed processing. On its part, distributed processing is one of the ways for traffic problem solving. In this case, a server turns into a logical mainframe, while Client/Server schema becomes a pseudo-Client/Server schema. It seems that today's Client/Server applications are mostly pseudo-Client/Server products.

Certainly, this situation is not casual. The point is that at a client application's designing stage only rough estimation is possible for receivable information's volumes and information interchange's intensities, what, of course, is not sufficient for exact decisions. In addition, any estimation of receivable information's variety is simply not on. Therefore, to avoid any surprises developers finally are inclined to entrust information processing to a server.

So, we can conclude that generally the cause of mentioned problem is the static estimation of information flows at a designing stage, whereas, in reality, for the problem solving is necessary dynamic estimation - during an application execution. This implies that estimation facilities must be the part of an application itself.

Generally, mentioned problem's solving, undoubtedly, is not confined to information flows assessment. Estimation facilities must also cover receivable information's relations to other characteristics of network, such as:

- Parameters of network;
- Servers and Workstations productivity (performance);

- Number of Servers and Workstations in a network;
- There activities;
- And other important relations.

Thus, any similar approach's promotion requires:

a) Special assessment methods development for critical characteristics of network;

b) Investigation of developed methods efficacy.

Unfortunately, today similar methods are not matured; they most likely are topics of research.

The modern visual programming systems are built on the Object-Oriented Programming ideas [4]. We think that such systems are ideal to slip quickly from a plan to its realization, i.e. for quick construction of first (draft) version of an application. But for serious and reliable applications they can't be considered as good instruments. The point is that:

- Visual programming systems contain numerous built-in objects.
- Each object has three types of characteristics: Properties, Methods and Events, which can occur with this object.
- The total number of object characteristics varies from 40 to 100 (including inherited characteristics).
- In a real, serious application, as a rule, figures dozen objects at least.

So, total sum of these characteristics in an application reaches to enough large number. Consequently, to think over the relations and the interdependency among these characteristics, all the more, to control them is practically impossible. Therefore, a developer, usually, selects from this huge set of characteristics only several and works with them. Correspondingly, the rest characteristics stay beyond the attention of a developer.

Hence, it turns out that the developer in reality controls (manage) relatively small part of the whole, whereas the greater part stays beyond his attention. An application built in such a way actually represents, so called, "black box". As is well known, systems built on the "black box" principle **are not reliable** (they can't be reliable).

Also, it is important that experienced developer upon analyzing some situation collects its characteristics and groups them together. This permits him to control characteristics behaviour more or less easily, hence to control situation itself. Unfortunately, modern programming tools do not permit such practice, because different characteristics are scattered among different objects. Consequently, this also makes difficult to control situation behaviour and, surely, this factor affects negatively on application reliability.

It seems that **situation** concept [5, 6] is one of the items of developer's thinking, but modern programming tools do not contain features related with situation.

So, it could say that modern complex applications designing and constructing logically requires new concepts, such as a situation, and related features in the programming tools.

We think that together with Event-Driven Programming it would be effective to base applications developing on the **Situation-Driven Programming.**

It would be an attempt to raise programming logic to a higher level - a real attempt to consider program's logic using more higher level, logically capacious concepts and

features.

Unfortunately, modern programming tools do not contain such features.

REFERENCES

1. W.Page, D.Austin and others. Using Oracle 8/8i. Que Corporation, 1999.
2. V.Gofman, A.Xomenenko. Delphi 6. BKHV- Peterburg, 2002.
3. P.Litwin, K.Getz, M.Gunderloy. Access 2002. SYBEXInc., 2002.
4. B.Liskov, J.Gateg. Ispol'zovanie abstrakzii i spezifikazii pri razrabotke programm. Izdatel'stvo "Mir", Moskva, 1989.
5. M.Bertgeimer. Produktivnye myshlenie. Moskva, PROGRESS, 1987
6. J.Barwise. The Situation in Logic. Lecture Notes, Number 17, CSLI, 1989.