COMPUTATIONAL PROBLEMS OF ELECTRICAL ENGINEERING

Vol. 4, No. 2, 2014

MODULAR NETWORK INTERFACE FOR DISTRIBUTED INFORMATION NAVIGATION SYSTEMS

Iryna Pasternak, Yuriy Morozov
Lviv Polytechnic National University, Lviv, Ukraine
pas_irusj@ukr.net, m_urij@ukr.net

Abstract: A modular approach providing unification by dividing a network interface into the universal and special parts in the context of distributed information navigation systems is proposed. The main principles of such approach and its software implementation are considered in the article. It is shown that the efficiency of the modular structure of the interface of distributed navigation systems, including the development time and speed of data processing, depends on the techniques used for their implementation and tools of object-oriented technologies.

Key words: modular construction, computer networks, network interface, client-server interaction, distributed navigation systems.

1. Introduction

Competition is forcing manufacturers of electronic systems to reduce the development time of new products. Recently, two years of lifetime for electronic devices were considered normal. Today this time is reduced to six months with the perspectives of further decreasing. The mobile phone is the vivid example of this situation. In addition, the growth of the “Internet of things” calls for reducing design time for network applications, services, etc. [1–6], which need specialized hardware and software solutions. The special aspects are power consumption, a range of wireless networks, cloud computing [7–10], security [11–15] etc. Besides, each task needs specialized solution for network interface, as universal solutions are not suitable here. Thus, the unification of specialized network interfaces development is a topic of current interest.

Modern mobile navigation systems include built-in devices and related services that allow users to combine them into one system via a wireless network through the appropriate interfaces. In the hardware and software part of navigation systems there are network interfaces of three link classes: 1) between navigation devices and navigation services; 2) between navigation services and a user interface; 3) between navigation services and a database.

Navigation devices are rapidly changing and outdated ones are decommissioned. There are dozens of types of these devices that are constantly updated. They are not compatible with each other. The quick change of types of navigation devices requires constant implementation of network interfaces designed for them. In order to reduce the amount of programming for realizing the network interface intended for connecting the navigation devices with navigation service in each new type of the navigation device, we need to unify the network interface for different types of devices.

Different users require different information, which is made up of the same navigational data. There are many types of navigational information and they are always caused by new types of users’needs. This in turn requires a large number of different forms of information. The user interface should display all kinds of navigation information on screens, in reports etc. The network interface generates a stream of requests to navigation service and interprets its answers. Accordingly, we should always handle distributed flow of requests that will come from the user interface to the navigation service, so it would be necessary to implement a network interface that will implement this interaction.

Navigation tasks require to be promoted by databases. According to new and modified existing navigational problems for different users, database contents change. The database can be located on a remote server, but even if the database is physically placed in the software and hardware navigation service, it interacts with the server through the network interface used for classic client-server interaction.

These interfaces use different communication protocols and data transmitted through them. But for them there is a common problem of reducing the amount of work on developing interfaces and improving their technical efficiency while their life is reduced.

The above problems can be solved by the decomposition of interfaces into universal and special parts, so the interface will become modular. The universal part will be a standard for many interfaces, and the special part will be unique to each network interface and quasi-universal, as developed on the basis of a single template. So, this can reduce the amount of work and time of development. The task of designing network interfaces can be implemented using object-oriented technologies [1-4, 6, 7].

The main tasks of the analysis and synthesis of hardware and software models of various parts of the interfaces are: 1) identification of objects and structures for functional algorithms and methods of implementation, which can be combined in appropriate modules; 2) the selection of classes, services and procedures for optimal criteria of the minimum development time and maximum performance of implemented interfaces. The aim of this work is an attempt to solve these problems basing theoretically on one possible mathematical model of interface, and developing practical guidelines for its implementation by means of object-oriented programming using the methods of inheritance, use and instantiation, and of the comparative analysis of different solution models of client-server interaction between interface modules.

2. Mathematical Model of Network Interface

The analysis of network interface behavior is simplified by neglecting the parameters requested by users as well as their responses. If some features of functioning the network interface are to be examined, you can ignore or the user's request or an incoming stream of a lower-level network interface [8, 9–14].

Considering the rules of network interface composition and using it as the result of certain operations applied to its components, we take into account that requests to component are formed by the network interface.

The network interface is described by this set:

$$IS = (Q, R, A, St, f, y) \tag{1}$$

where Q are input requests network interface, R are responses of the lower level of the network interface, A is the output alphabet network interface, St is the set of conditions of the network interface, f, ψ are functions of transitions and outputs.

Let us consider each component of the network interface. The set of characters that form the input alphabet of the network interface is described as follows:

$$Q = \{Qi\}, \tag{2}$$

$$Q = \left\{ Id_Q^{(i)}, Pq_1^{(1)}, \ldots, Pq_{Na^{(1)}}^{(1)} \right\} \tag{3}$$

$$P_j^{(i)} \subset D_{Q_j^{(1)}}^{(1)} \times \cdots \times D_{Q_j^{(1)}}^{(l)}, l = N_{P_{Qi}}^{(j)}, \tag{4}$$

where Id_Q is the set of unique identifiers queries.

The set of characters output alphabet network interface, describe as follows:

$$A = \{Ai\}, \tag{5}$$

$$Ai = \left\{ Id_A^{(1)}, Pa_1^{(1)}, \ldots, Pa_{NA^{(1)}}^{(1)} \right\}, \tag{6}$$

$$P_j^{(i)} \subset D_{A_j^{(1)}}^{(1)} \times \cdots \times D_{A_j^{(1)}}^{(l)}, l = N_{P_{Ai}}^{(j)}, \tag{7}$$

where Id_A is the set of unique request identifiers.

The set of characters that make up the response of the lower level of the network interface, we describe as follows:

$$R = \{Ri\}, \tag{8}$$

$$Ri = \left\{ Id_R^{(i)}, \mathrm{Pr}_1^{(1)}, \cdots, \mathrm{Pr}_{NR^{(1)}}^{(1)} \right\}, \tag{9}$$

$$P_j^{(i)} \subset D_{R_j^{(1)}}^{(1)} \times \cdots \times D_{R_j^{(1)}}^{(l)}, l = N_{P_{Ri}}^{(j)}, \tag{10}$$

where Id_R is the set of unique response identifiers of the lower level of the network interface.

Each request to the network interface and its answer has two parts: identification and parametric. The identification part uniquely identifies the type of request. The parametric one is actually additional information that accompanies requests and responses to them by the network interface. The description of the network interface state is also based on similar approach. In general, a specific state St of the network interface can be described as follows:

$$St^{(i)} = \left\{ Id_{St}, St_1^{(i)}, \cdots, St_{NSt}^{(i)} \right\}, \tag{11}$$

when the network interface has some components, or

$$St^{(i)} = St^{(i)} \subset D_{St}^{(1)} \times \cdots \times D_{St}^{(l)}, \tag{12}$$

when the network interface has no components,

where $St_1^{(i)}$, …, $St_{NSt}^{(i)}$ are states of the network interface components.

Thus, the formal description of the network interface is recursive (to a certain low level of refinement at which states are of atomic nature).

Each state $St^{(i)}$ contains the Id_{St} state and status of all components belonging to the network interface. The number of states is limited to the number of possible component states.

However, in reality this number is significantly reduced by the constraints and relationships that can exist between the states of the network interface.

The reaction of the network interface cannot be fully determined in case of parametric descriptions. That is, the system response to each request is described by the set of possible transitions, outputs and their probabilities.

Taking into account formulae (1), (11), (12), we obtain the function of transition to the network interface:

$$j(Q_i, St_j) = \left\{ (St_k, \Pr(St_k, Q_i, St_j) \right\}, \qquad (13)$$

$$\sum_k \Pr(St_k, Q_i, St_j) = 1. \qquad (14)$$

Then, with the use of the same formulae (1), (11), (12) we can obtain the output function for the network interface:

$$\Psi(Q_i, St_j) = \left\{ (A_k, \Pr(St_j, Q_i, A_k) \right\}, \qquad (15)$$

$$\sum_k \Pr(St_j, Q_i, A_k) = 1. \qquad (16)$$

In our opinion, the mathematical description and approach to grouping input, transition and output functions of the network interface allows further correct distribution of client-server interaction services among universal and specific parts of hardware and software interface implementation. In addition, it causes multi-device distributed control processes for several processor cores depending on their configuration, and, therefore, the increase in the communicative efficiency of the system.

3. Model of Modular Client-Server Interaction

Firstly, we have to analyse the network interface and modular network interface for correct modelling of modular client-server interaction. The network interface has standard and special features shown in Fig. 1.

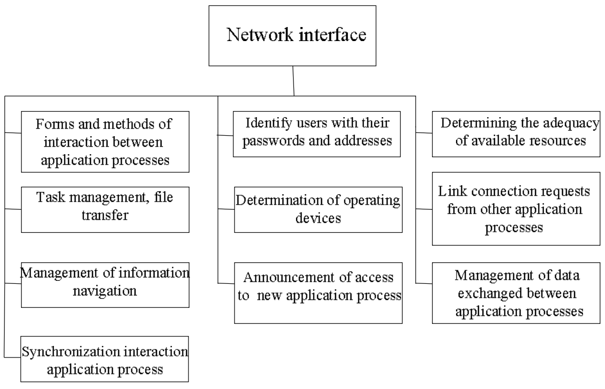
Network interface
Forms and methods of interaction between application processes
Task management, file transfer
Management of information navigation
Synchronization interaction application process
Identify users with their passwords and addresses
Determination of operating devices
Announcement of access to new application process
Determining the adequacy of available resources
Link connection requests from other application processes
Management of data exchanged between application processes

Fig. 1. The functions of network interface.

The analysis of these features shows that some of them do not depend on implementation and can be generalized (for example, task management, access to application, linking etc.), while others depend on their implementation and can not be generalized. The first part of services, which are universal for different objects, we will develope and test once, and the second part of services, which are different for different network interfaces, we transferred to a special module, which is to be developed for each network interface separately (see Fig. 2).

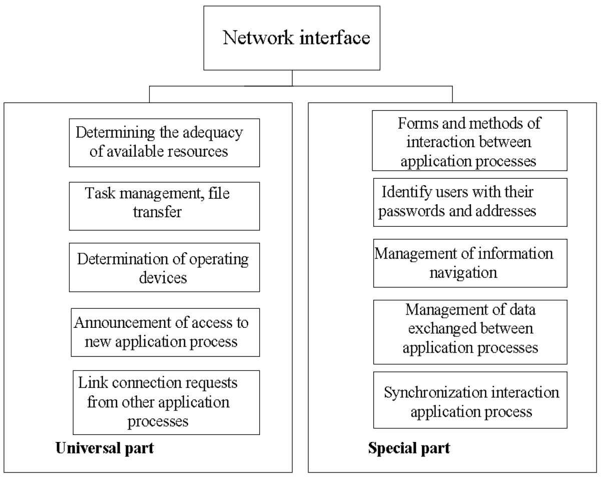
Network interface
Determining the adequacy of available resources
Task management, file transfer
Determination of operating devices
Announcement of access to new application process
Link connection requests from other application processes
Universal part
Forms and methods of interaction between application processes
Identify users with their passwords and addresses
Management of information navigation
Management of data exchanged between application processes
Synchronization interaction application process
Special part

Fig. 2. Modular network.

The model of modular client-server interaction consists of client and server parts, and interaction between them occures through the network interfaces in a distributive network (Fig. 3).

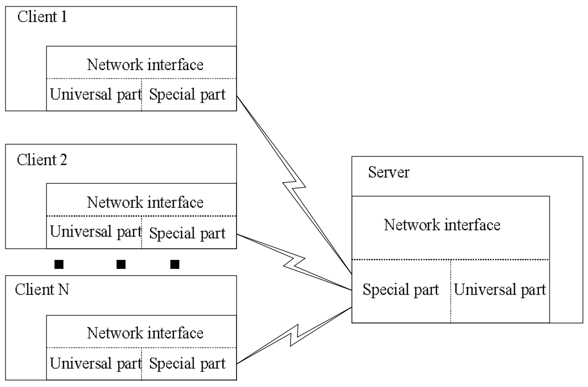

Fig. 3. Model of modular client-server interaction.

The difference is that on the server side the network interface is in the waiting mode, during which it waits for a signal from some port or channel. On the client side the network interface is in a passive state, but when a user sends a request to the server, it will create a connection and call to the server.

The model of the modular network interface consists of the universal part, whose functionality is supported by all clients, and the specialized part specific to the type of client's implements realizing specific commands and data types. Being based on a general model of modular client-server interaction, the model of modular network interface on the basis of inheritance, instantiation and use can be realized.

4. Model of Modular Network Interface Based on Principle of Inheritance

Static-dynamic method of network interaction. In cases, where the structure of command and data customers does not change and repeate, that is, it is static and dynamic, it is possible to implement the modular network interface based on the principle of inheritance. The principle of inheritance is used in the model of the modular network interface realized by the author. This principle allows one class to emulate the characteristics of another. Thus, a derived class gets all features of a base class, and can be enhanced by adding its own features. The derived class is a specialized version of the base class. The derived class inherits all members defined in the base class, and adds its own unique elements to them. At the process of inheritance the base class remains unchanged [5, 19–21]. A class can have "subclasses", that is, special, extended versions of a superclass. Whole trees may even be formed on the basis of the principle of inheritance. Subclasses inherit attributes and behavior of their parent classes, and introduce their own features. Inheritance can be single (one immediate parent) and plural (several parents). It depends on the choice of a programmer who implements the class and programming languages. The scheme of the modular network interface based on the principle of inheritance is shown in Fig. 4.

The model of modular network interface based on the principle on inheritance and serving for the connection of each client to the server should have its own network interface (1, 2, .., N), which contains a communication protocol (1,2, .., N).

The communication protocol can be divided into an universal part, common to all network interfaces, and a special part specific for each type of customer. The universal part of the network interface will describe the base class. For each type of communication protocol its own class is created by inheriting from the base class with the expansion of its range by the set of special commands.

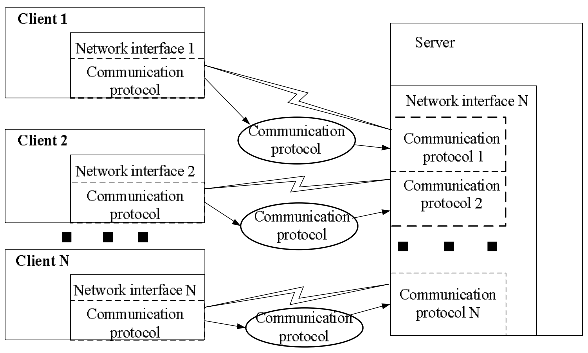

Fig. 4. Model of modular network interface based on the principle of inheritance.

Thus, we have a base class network interface. It is inherited by subclasses for each type of client and according to each type of the communication protocol, and they in turn shape objects which themselves are the network interfaces designed for a certain communication protocol.

The advantage of this model is that the principle of inheritance allows us to use the existing code of the parent class for all derived classes many times. Using a modular network model based on the principle of inheritance, we can create the base class that will determine the features inherent in many objects. And then we can use it to create any number of specialized derived classes with the addition to each of them its unique features.

The model of the network interface based on the principle of inheritance was practically implemented in an informational navigation system "ZITtrack" (Lviv, Ukraine). The user can access this information system (IS) through a web interface which is created interactively by users' activities using information from an IS database. The user interface is implemented in the programming languages Java and JavaScript. In this case, the user interface consists of a template designed according to a Java Server Page technology (JSP) and a content formed programmatically. Information forms of the user interface can be divided into several categories. The information form of each category is built by the same principle. For example, all statements look similar but contain different information [10-15].

The user interface is loaded into a user's browser as a web page, which interact with the server using IP technology Ajax, and it is a standard client-server interaction. The network interface of a class can be divided into the universal part, which includes web site design and programming on the page using JavaScript, and the special part which includes information form data.

The universal part of the network interface is realized in a WebPageAdaptor class (Fig. 5), and special parts of modular network interface for each information form are described in appropriate derived classes. For example, the information form of navigation devices is formed by the DeviceInfo class, which is inherited by a WebPageAdaptor class. All other classes of the network user interface also inherit the WebPageAdaptor base class. Taking into account these facts, we can define the base class, knowing the functions and variables which are inherited by its subclasses.

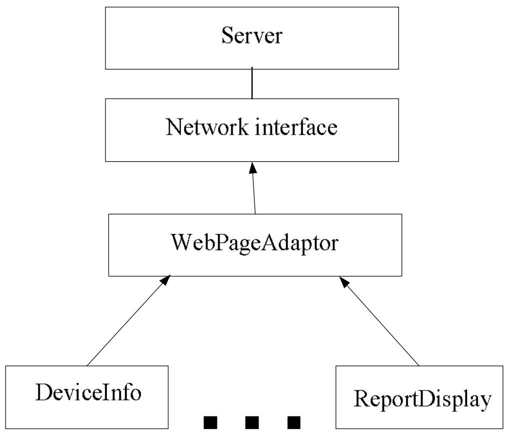

Fig. 5. Practical implementation of model modular network interface based on principle of inheritance.

The universal part is implemented as a WebPageAdaptor class. The special part of the network interface can be realized, for example, as a DeviceInfo class.

5. Model of Modular Network Interface Based on Principle of Instantiation

The dynamic method of network interaction. In cases when the structure of customer data will often change, that is, it will be dynamic, but commands are not changeable, the modular network interface based on the principle instantiation can be realized. The model of the modular network interface based on the principle of instantiation (Fig.6) involves the use of generalized programming or programming-based templates. A template is a class in which the data type is a parameter; for the creation of an object of this class the data type must be specified. This process is called instantiation. The modular network interface can be implemented as a template using the instantiation principle [16, 17]. The universal part of the modular network interface can be described by a template class and a special part by the parameter class.

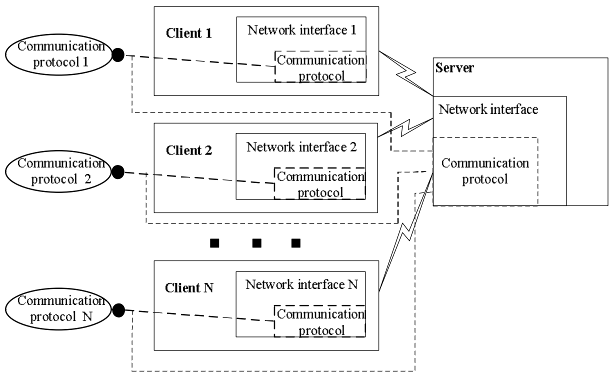

Fig. 6. Model of modular network interface based on instantiation principle.

A certain network interface in the model is the object of the parameterized class of the template instantiated by the special class of the network interface. Instantiation of the template by different classes of the special part of network interface leads to the creation of different network interfaces. That is, for implementing many network interfaces using the instantiation principle it is necessary to create one template class of the universal part of the network interface and required number of classes of the specific parts of the network interfaces. A template class exports operations which can be performed on its sample.

On the contrary, a parameterized class argument is used to import classes and values which are used as communication protocols.

We use software in Java programming language, precisely because at the compilation it checks their interactions when the instantiation is carried out.

For the practical implementation of the model of the modular network interface based on the principle of instantiation, the server of the information navigation system "ZitTrack" is connected as a client to the database server. The information in database is subdivided into tables by functionality. The access to any table is gained by using queries in SQL language. The same queries to different tables are similar, differing only in names of tables and lists of attributes, and the conditions of selection results. Interaction with the server database and the server itself can be described by a set of commands which look similarly but have a different content.

The appearance of commands providing the access to the database can be considered the universal part of the network interface, and data tables, attributes and conditions form the special part of the network interface.

Any template class is instantiated before being used. The template class of the modular network interface based on the instantiation principle is created as parameterized.

The essence of the model of the modular network interface based on the instantiation principle is to create the template of the access to the database whose parameter will be the relevant database table. The network interface is implemented as a DB server, which provides the access to the database. This server has a lot of classes and all of them are instantiated. So the whole work with the database is instantiated. The classes of the server will be: DBRecord, DBFactory and other supporting classes, for example, DeviceRecord (Fig.7).

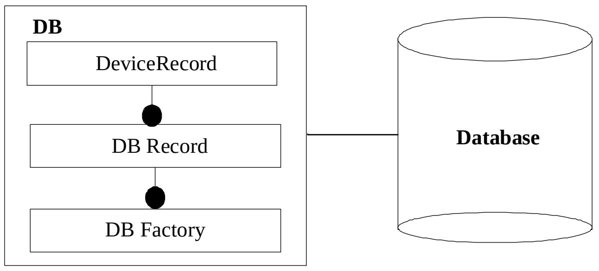

Fig. 7. Implementing a practical model of modular network interface based on principle instantiation.

How to create universal methods when all tables are different and they have different arguments? For this task the pattern of the access to the table is used which is called DBRecord. To use this template the class DBFactory is used. In addition, there is a set of classes, each of them describing a particular table and working with it. Each method of working with these tables uses the object which is instantiated by the class DBRecord, and this object is a table object, so it can call itself through the same pattern. That is, a table template exists, and there are universal techniques for working with tables. When it is necessary to refer to a specific table, this pattern is instantiated by the object of the table which already contains mechanisms of working with this certain table, but not with others. So there are generalized methods and individual implementations for each table.

In practical implementation of the model of the modular network interface based on the principle of instantiation (Fig. 7), in which the DeviceRecord class is instantiated by the DBRecord class, the template of the universal part of the network interface is presented in the DB Record class.

6. Model of Modular Network Interface Based on Principle of Use

The static method of network interaction. In cases when the client network interface is specific to each customer and includes unique commands but similar data structure, i.e., the network interface is static, you can implement the modular network interface on the basis of use. The model of the modular network interface on basis of use (also called composition or inclusion), we mean the method of creating a new class from existing classes by incorporating (also known as delegation) the attached object. Nested objects are usually announced closed classes, making them unavailable for application programmers working with the class. On the other hand, a class creator can change these objects without disturbing the operation of existing client codes [18–21].

In addition, the replacement of embedded objects at the stage of program operation allows us to change its behavior dynamically. The model of the modular network interface based on the principle of inheritance does not have this flexibility because for the derived classes some restrictions are set which are checked at the stage of compilation. On the basis of the model of the modular network interface based on the principle of use a delegation method is implemented when the task set before the external object is delegated to the internal object, specializing in solving problems of this type. In this model, there is a “part-whole” relation between two equal objects: the network interface and communication protocols. Thus, one object (a container) has a reference to another object. Both objects can exist independently if the container is destroyed, its content will still exist. Fig. 8 shows the appearance of the modular network interface based on use patterns.

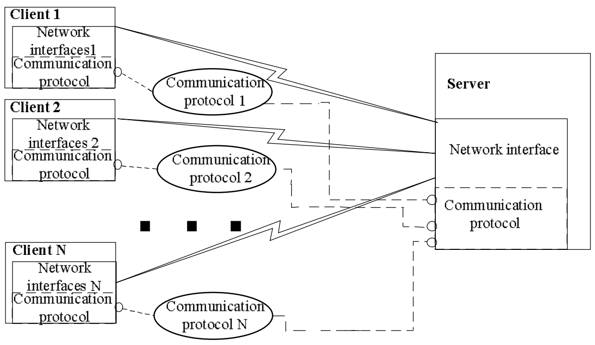

Fig. 8. Model of modular network interface based on use.

In the model of the modular network interface on the basis of use the network interface is the object of a container class. In practice, the network interface is only one for one server, but it uses the first communication protocol for the connection with the first client, the second communication protocol for the second client, and the N-th communication protocol for the N-th client. In general, we can say that each client has its own type of the network protocol and this type will determine the type of the client. Each type of the network protocol is characterized by a set of commands, and each command is processed by some feature. The set of command formats and processing functions contained in the object communication protocol are used by client and server. Only with one difference, each client has his own network interfaces, but for the server there is only one network interface for all of them. In turn, communication protocols (1,2, .., N) will be included in the object of the network interface. In this model we have one network interface object and the corresponding class, and many objects that are responsible for communication protocols.

The practical implementation of the modular network interface based on the principle of use are realized in information navigation system “ZITtrack”. In this information system navigation devices transmit their information to the server, and each type of devices has its own communication protocol. Also, the device can be in different modes. Information from the device is stored in the same database table. This means that all data must be given in the same form. In the communication protocols of navigation devices the universal part can be identified which is the same for all of them, namely, the establishment and termination of connections, logging and so on. Meanwhile, each device has its own message format and connection settings.

The special part of the modular network interface consists of constants of the configuration of the universal part and functions of data processing of the communication protocol. While implementing practically the method of use, we face the problem of conveying information. The whole set of information consists of telemetry data. Typically, they are sent over network in the form of data streams consisting of a header and some data packets received from the sensor subsystem. This program being implementing, everything looks like a simple set of data.

The network interface based on the principle of use in the information navigation system "ZITtrack" is used for communication with navigation devices. In practical implementation of this model there is an object connection as the method of the object of the universal part of the network interface which, being a parameter, has the objects of the special part of the network interface. The universal part of the network interface for the communication with navigation devices consists of the object of the communication server (such as GPSEvent) and linking object. The special part of the network interface is implemented as a constant object and the object of the communication protocol (such as TrackClientPacketHandler) (see Fig. 9).

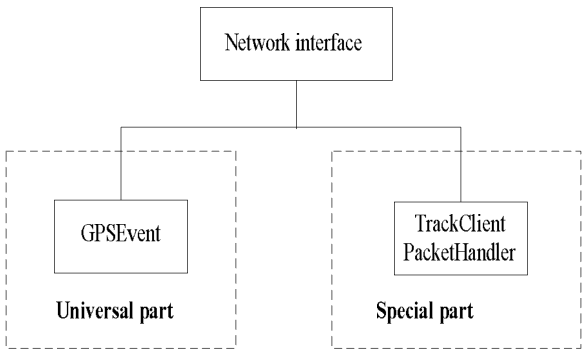

Fig. 9. Appearance of modular network interface on basis of use.

The universal part of the network interface on the basis of use will contain GPSEvent. As we can see, the universal part of the network interface on the basis of use, which is used by all IS servers and is the same for all of them, irrespective of the number of the servers, contains the file GPSEvent. In the first file program the ports are announced via which the client will communicate with server and vice versa in the network. The second file describes server configuration.

The special part of the modular network interface based on the principle of use in this case is contained in TrackClientPacketHandler.

We can see that the special part of the modular network interface based on the principle of use is simplified and contains two files of Track Client Packet Handler. The Track Client Packet Handler file will include information on connections and transactions having occurred and occurring at client-server network interaction. The modular network interface based on the principle of use, as the term "use" implies, uses the date of the function named getHandlePacket. In the modular network interface based on the principle of use, the client will have access to the universal part of network interface connection with devices, information on which will be on the server.

7. Comparison of Technical Performance and Reliability of Models

The example of the network interface of the navigation system is the network interface based on Jini architecture. Its disadvantage is a relatively long programming and adaptation to the specific system (Table 1).

Network interface consists of several options for communication protocols which are loaded on the client side when looking for server and then executed locally. When the client calls a communication protocol, it sends a request to a server on the network that creates the connection. The network interface can use specialized equipment to fulfill client requests. One of the important principles of Jini architecture is that the protocol used for communication between the object in proxy service and the remote server does not have to be known to the client. However, it is not important for the client, because the communication protocol is the part of the implementation of the already known network interface.

The advantage of modular interfaces is that the approach to implementation may change over time. Alternatively, the network interface module can only be used as a proxy to a remote server.

The comparison of the performance parameters of the Jini interface and the proposed solutions of modular network interfaces was made. The criterion for assessing the technical efficiency may be a runtime of the server client request. On the client side the time measurement for processing the request is not correct, because there are network delays affecting the passage of connection command. Measurements of processing time on the server side give the time duration from receiving a request to sending a reply. The laptop Asus K73E-TY272DAt plays a role of the client at the setting of an experiment.

According to the accepted standard methodologies, for example, those used for distributed GRID-systems and so on, technical efficiency of the interface is determined by the average indicators of processing requests in terms of the maximum (100 %), minimum (at 1 %) and moderate/nominal (10 to 30 %) by a load processor system. In all cases, when tests determine a stable stay in a fixed position, they limit switch to the next cycle. Summary results of testing are shown in Table 1.

Based on the mathematical model of the network interface module compared models were calculated, the modular network interface was built by three principles and the following data was obtained. The processing time of the client request has a value about 0,2 ms for the modular interface implemented on the basis of the principle of use, and it is determined by a direct time value of processing the command. Similar values for the interface implemented on the basis of inheritance principle are ~ 0,3 ms, and on the basis of instantiation ~ 0,04 ms. In the latest version, the increasing of the number of requests from the client can lead to increasing the processing time, which is needed for processing requests to the database, and on the basis of these criteria modular network interface is built on the instantiation principle.

Table 1

Comparison of the performance parameters of modular network interfaces, 2015

Interface type Applied /Ethernet	**The response time of network interface, ms**	**Time for developing a network interface, folk. / hr.**
Specialized network interface	1,6	8,7
Modular network interface is built on principle:		
Inheritance	0,3	2,7
Use	0,2	0,7
Instantiation	0,04	1,8
Modular network interface (thresholds)	~0,3-0,04	~2,7-3,64
Network Interface based on Jini	1,2-2,0	7,4

Thus, on basis of mentioned above it can be said that the modular network interfaces based on the principles of inheritance and instantiation are more reliable than the interface based on the principle of use. But in turn, the interface based on the principle of use is also very efficient.

As we can see in Table 1, the response time of the network interface built in a modular fashion depends heavily on the method of its implementation by means of object-oriented programming. In particular, the interface implemented on the basis of instantiation method has a timing at 20-40 times better than the interfaces based on the principles of use and inheritance, and more then ten times better than specialized interface settings, including the interface realized on the basis of Jini technology.

These data suggest that the development of the modular network interface saves time and reduces complexity. Their implementation is not always, but in most cases more efficient than the generic network interface.

8. Recommendations for Implementation of Modular Network Interface

According to above mentioned principles, the network interface should be divided into 2 parts: universal and special. To implement the modular network interface, it is also necessary to identify which principle is to be used. The universal part contains shared functions and constants, and functions specific to each network interface will be included in its special part. That is necessary to analyze classes of network interfaces and identify common features, which are to be considered in the universal class, and features unique to the given network interface which should be separated into some class of the special part.

The specific model of implementation of the modular network interface is chosen for practical reasons.

The algorithm of the selection of the principle of implementation of the network interface is shown below:

1. It is necessary to describe the network interface in abstracto to be implemented in the terms of functions.
2. Then the decomposition of network interface into two parts should be carried out, one of which includes features that are the same for all interfaces, and the second part consists of special functions.
3. Then we need to evaluate the structure of the argument of the special function:
   - 3.1. If there is a group of functions which are common to all network interfaces but have a different implementation for each network interface, we need to use the principle of inheritance (see part 4 of the article).
   - 3.2. If the function described by the special part of the network interface is unique to each network interface, then it is advisable to use the instantiation principle, (see part 5 of the article).
   - 3.3. If functions making up the universal part of the network interface are identical and their arguments are identical too, but they get various data at their implementation, it is advisable to apply the principle of use (see part 6 of the article).
   - 3.4. If none of options is suitable, see part 1 of the article.
4. Programmatic implementation of the selected principle for the construction of the network interface is fulfilled.

For example, in order to speed up the development of the network interface it is advisable to apply the principle of use, as it is shown above in the recommended algorithm.

9. Conclusions

The main results of this article address the solution of an applied problem, namely, the improvement of methods of designing hardware and software of network interfaces for distributed navigation systems. It was theoretically grounded and experimentally confirmed that the response time of the network interface built in a modular fashion considerably depends on the method of its implementation by means of object-oriented programming. In particular, the interface implemented on the basis of the instantiation method operates twenty times faster than interfaces based on the principles of use and inheritance, and more than ten times better than specialized interface settings, including based on the Jini technology. The modular network interface based on the principle of inheritance is less technically effective, but reliable. The interface on the basis of the principle of use is more technological, and can be used for the complex solution of the problem of increasing query processing speed and reducing the time of developing the modular interface. The proposed recommendations for designing the modular interfaces, in our opinion, may be useful in dealing with basic issues of improved and effective information processing in navigation systems.

References

[1] L. Pierce and S. Tragoudas, “Threshold network synthesis”, *ACM Journal on JETC*, vol. 10, no. 2, pp. 17–21, 2014.

[2] P. Wettin, A. Vidapalapati, A. Gangul, and P. P. Pande, “Complex network-enabled robust wireless network-on-chip architectures”, *ACM Journal on JETC*, vol. 9, no. 3, pp. 24–30, 2013.

[3] R. Beckman, K. Channakeshava, F. Huang, J. Kim, A. Marathe, M. Marathe, S. Saha, G. Pei, and V. S. Anil Kumar, “Integrated Multi-Network Modeling Environment for Spectrum Management”, *IEEE Journal on Selected Areas in Communication, special issue on Network Science*, vol. 31, no. 6, pp.1158–1168, 2013.

[4] P. Dourish, R. E. Grinter, J. Delgado de la Flor, and M. Joseph, “Security in the wild: user strategies for managing security as an everyday, practical problem,” *Personal and Ubiquitous Computing (ACM/Springer)*, vol. 8, no. 6, pp. 391–401, November 2004.

[5] R. J. Wirfs-Brock, “Refreshing patterns,” *IEEE Software*, vol. 23, no. 3, pp. 45–47, May/June 2006.

[6] H. H. Thompson and R. Ford, “Perfect storm: The insider, naivety, and hostility,” *ACM Queue (Special Issue: Surviving Network Attacks)*, vol. 2, no. 4, pp. 58–65, June 2004.

[7] D. P. Anderson, “BOINC: A System for Public-Resource Computing and Storage”, in *Proc. of the Intl. Workshop on Grid Computing. IEEE*, pp. 4–10, 2004.

[8] H. Blohm, *Hierarchical Arrangement of Modified Class Loaders and Token Driven Class Loaders and Methods of Use*, United States Patent 7.536.412.B2. 2009.

[9] M. D. Dikaiakos, G. Pallis, D. Katsaros, P. Mehra, and A.Vakali, “Cloud Computing – Distributed Internet Computing for IT and Scientific Research”, *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, 2009.

[10] K. Geihs, Selbst-Adaptive Software. Informatik Spektrum. Springer. pp. 133–145, 2007.

[11] R. L. Grossmann, “The Case for Cloud Computing”, *IEEE IT Professional*, vol. 11, no. 2, pp. 23–27, 2009.

[12] G. Pei and V. S. Anil Kumar, “Distributed Approximation Algorithms for Maximum Link Scheduling and Local Broadcasting in the Physical Interference Model”, in *Proc. IEEE INFOCOM 2013*, pp. 1339–1347. Turin, Italy, April 14–19, 2013.

[13] C. Kuhlman, V. S. Anil Kumar, and S. Ravi, “Controlling opinion propagation in online networks”, *Computer Networks*, vol. 57, no. 10, pp. 2121–2132, 2013.

[14] G. Pei, S. Parthasarathy, A. Srinivasan, and V.S. Anil Kumar, “Approximation algorithms for throughput maximization in wireless networks with delay constraints”, in *Proc. IEEE INFOCOM*. Shanghai, China, April 10–15, 2011.

[15] V. S. Anil Kumar, M. Marathe, and S. Parthasarathy, “Cross-layer capacity estimation and throughput maximization in wireless networks”. In *Algorithms for Next Generation Networks*, pp. 67–98, 2010.

[16] L. Pierce and S. Tragoudas, “Threshold network synthesis”, *ACM Journal on JETC*, vol.10, no. 2, pp. 17–21, 2014.

[17] P. Wettin, A. Vidapalapati, A. Gangul, and P. P. Pande, “Complex network-enabled robust wireless network-on-chip architectures”, *ACM Journal on JETC*, vol. 9, no. 3, pp. 24–30, 2013.

[18] R. Beckman, K. Channakeshava, F. Huang, J. Kim, A. Marathe, M. Marathe, S. Saha, G. Pei, and V. S. Anil Kumar, “Integrated Multi-Network Modeling Environment for Spectrum Management”, *IEEE Journal on Selected Areas in Communication, special issue on Network Science*, vol. 31, no. 6, pp. 1158–1168, 2013.

[19] C. Ottow, F. van Vliet, and P.-T. de Boer, and A. Pras, “Impact of IPv6 on Network and Web Application Penetration Testing”, in *Proc. EUNICE 2012 (LNCS 7479)*, Budapest, Hungary, August 2012.

[20]P. Dourish, R. E. Grinter, J. Delgado de la Flor, and M. Joseph, "Security in the wild: userstrategies for managing security as an everyday, practical problem," *Personal and Ubiquitous Computing (ACM/Springer),*, vol. 8, no. 6, pp. 391–401, November 2004.

[21]R. J. Wirfs-Brock, "Refreshing patterns," *IEEE Software,* vol. 23, no. 3, pp. 45–47, May/June 2006.

МОДУЛЬНИЙ МЕРЕЖЕВИЙ ІНТЕРФЕЙС ДЛЯ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ НАВІГАЦІЙНИХ СИСТЕМ

Ірина Пастернак, Юрій Морозов

Запропоновано модульний підхід, що забезпечує уніфікацію шляхом розділення мережевого інтерфейсу на універсальну і спеціальну частини в контексті розподілених інформаційних навігаційних систем. Розглянуто основні принципи такого підходу та його программне застосування. Показано, що ефективність модульної структури інтерфейсу розподілених навігаційних систем, враховуючи час розробки та швидкість обробки даних, залежить від методик, використаних для їхнього застосування та інструментарію об'єктно-орієнтованих технологій.

Iryna Pasternak – Assistant Professor of Computer Engineering Department, Institute of Computer Technology, Automation and Metrology, Lviv Polytechnic National University, Ukraine.

Yuriy Morozov – Ph.D., Associate Professor of Computer Engineering Department, Institute of Computer Technology, Automation and Metrology, Lviv Polytechnic National University, Ukraine.