Using of assembly technology for the construction of users interfaces in network informational computational system

Piskunov S.V., Kratov S.V.
Network Information Technologies Chair
Novosibirsk State Technical University
Novosibirsk, Russia Federation
piskunov@ssd.sscc.ru, kratov@sscc.ru

Veselov A.V., Ostapkevich M.B.
Supercomputer Software Department
ICMMG SB RAS
Novosibirsk, Russia Federation
alexandr_v@ngs.ru, ostap@ssd.sscc.ru

Abstract - The description of assembly technology and its application to the construction of modules of the network informational computational system is presented in the paper. The application of the technology allows the system to evolve as a product with free access, giving both developers and users equal capabilities to extend the system's functionality at all the levels of its modular structure.

I. INTRODUCTION

It is well known that the simulation is an important part of the innovation process. It helps to build an implementation of an innovation project quickly and easily. The supercomputers have become an efficient tool of simulating both academic and applied problems of big scales. The need for high performance computations constantly grows. The list of supercomputer application areas expands while the number of users from very versatile areas augments. In order to use supercomputers efficiently in the innovations, it is necessary to construct a network informational infrastructure. It must include not only system and application functions that provide simulating on supercomputers, but also the tools that support the interactive communications between the developers of an innovation project and that give access to huge amounts of common and professional information.

The requirements for network informational computational system (NICS) were formulated in [1, 2, 3]. The NICS architecture, its objective, and its prototype implementation were presented ibid. The foundation of NICS is represented by a kernel that is composed by a library named Dynamically Configurable Modular System (DCMS) [4] and a set of modules that form the system's skeleton. These modules implement the minimal set of functions required so that the system could be used: a unified access to OS API-s (file I/O, network communications, management of processes, threads, shared libraries and virtual memory), an implementation of versatile data structures, a management and an authorization of users, an implementation of data security and a number of functions for the input, editing and searching of miscellaneous data in the system through a Web browser. External modules are built above the kernel with the help of an assembly technology. A number of modules were developed. Particularly they provide a formation of informational space, an interaction of users and a construction of a virtual team of users for collaborative implementation of a project. The informational space is represented by a repository, which is a reliable storage of user's data fragments called cards and by classifiers that help to order cards by one or more classification hierarchies. The construction of the system was carried out by its developers. The assembly technology, that makes it possible to enrich the system by modules produced not only by the developers themselves, but also by users, is covered in [5, 6, 7].

We focus mainly on the development of the system as a product with a public and free access in this paper. By saying this it is meant that the developers and the users equally participate in the evolution of the system by creating external libraries of modules and of complete applications.

The *objective* of this work is to demonstrate that the uniting of all the steps of the technology of NICS construction, that were presented earlier, into a certain complex assembly technology gives both developers and users a capability of extending the system's functionality at all the levels of its modular structure and primarily in its GUI and Web parts. All this can help to make the development of NICS more dynamic and versatile.

II. SELECTING A WAY FOR NICS DEVELOPMENT

All the contemporary systems are built as modular and they have an open architecture. It means that they provide portability to versatile platforms, scalability, interoperability (cooperative work with other open systems) and friendliness of users' interfaces. Basically there are two ways of their construction as a software product.

The first way. A complex system is created from the beginning to the end by such a team of developers that is adequate both by its size and skills. Designing by one team and using the open architecture ensures the reusability of modules, eases testing, debugging, maintenance and modernization and helps to reach logical unity of the system from user's point of view.

Project is supported by grant RAS 13.6

The second way. A small team creates just a kernel of a system and makes available a technology and tools for the construction of its modules. The kernel contains obligatory infrastructural functions that support addition of new modules to the system. A limited set of standard and frequently used modules is built by the developers themselves. Besides a useful collection of functions these modules serve as examples of how users can develop their own similar modules when they need a certain function and want to implement it within the system. If a system seems to be attractive for a sufficient number of users, its set of functions starts to grow by the functions implemented by these users without additional efforts of the developers. This in its turn leads to a further increase of users. The snowball effect starts to work for successfully designed systems.

It is the second way that was chosen for NICS further evolution by its developers.

In order for the system to be developed successfully the following conditions have to be met:

1. User must have access to all the levels of modular structure of the system.

2. A wide range of languages and tools must be available for users when developing modules: proprietary assembly language, DCMS library, general purpose languages such as C, C++, Java, Perl, PHP and such tools as Protege editor, Rasqal RDF Query library, etc.

3. Participants with any kind of skill – from programmers to experts in a certain domain can develop a system, all they have to know clearly is just what a module must do.

These conditions form the guidelines for the construction of complex technology of NICS development.

III. THE DESCRIPTION OF COMPLEX TECHNOLOGY OF THE DEVELOPMENT OF NICS MODULAR ARCHITECTURE

The foundation of the assembly technology, that enables the development of NICS as a system with public access, is formed by three ways (along with their combinations) of external module construction. The scheme of the technology and variants of its application are presented at Fig. 1.

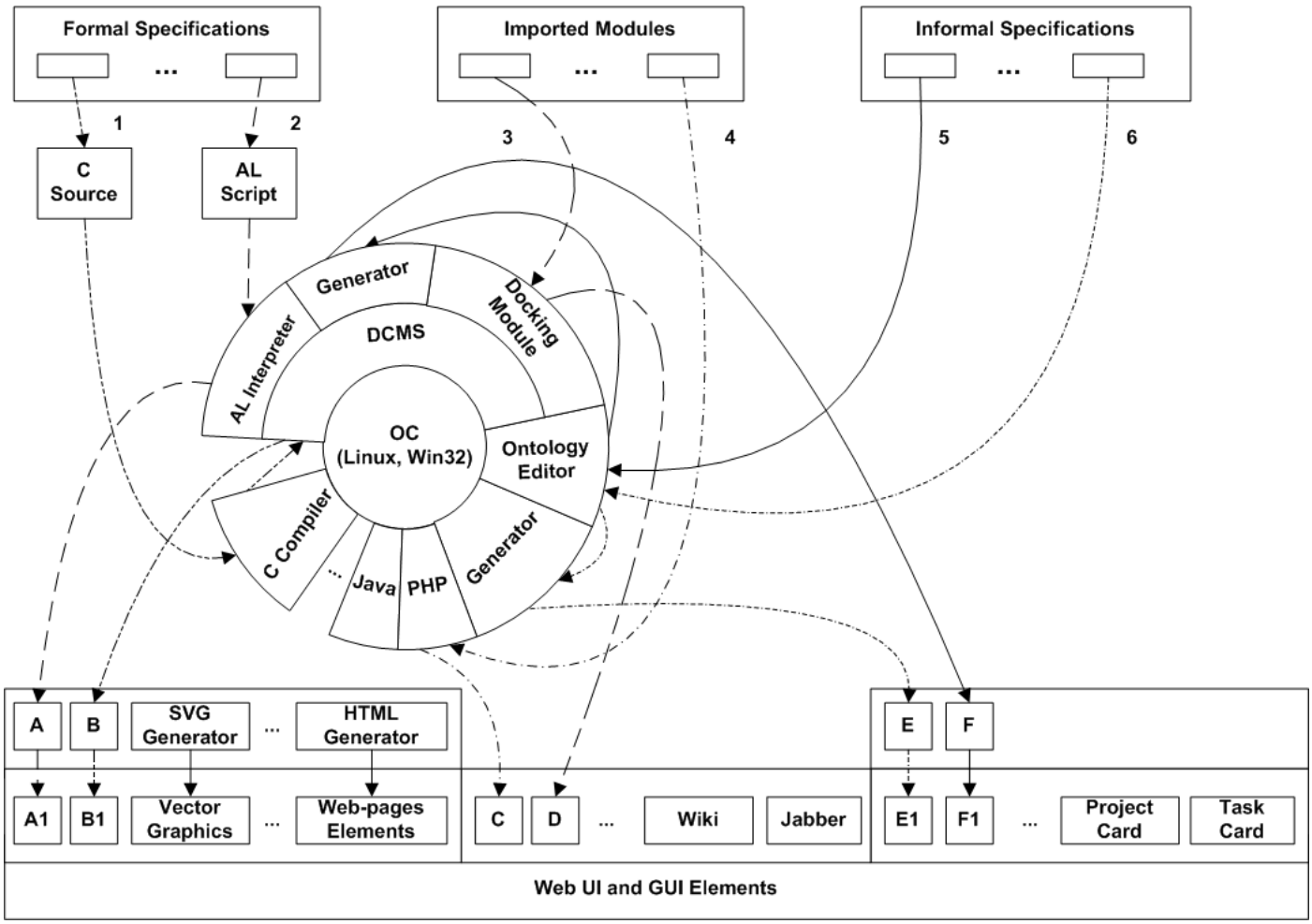

Fig.1 The scheme of the technology and variants of its application

The first variant is fully based on DCMS library. DCMS is a superstructure above an OS (Win32, Linux) and consists of procedures. All the modules built above it contain only event handlers. All the inter-modular interactions are based on events. Event-driven interface makes it possible to embed new modules and alter the existing ones without the need to modify the source text of the kernel. There are two ways of module construction within the first variant. They are denoted by symbols 1 and 2 at the figure. The modules that are built on the basis of DCMS and written in C/C++ are represented by dynamically linked libraries (module B at Fig. 1). The modules written in the assembly language (denoted as AL at the figure) are represented by scripts (module A at Fig. 1). An addition of a new module can be done by any user acquainted with C/C++ or DCMS assembly language. There can be application or system modules. For example, the path 1 can be used to solve the following task. A certain user who uses KOrganizer to keep contacts wants to have a possibility to transfer those contacts to NICS informational space. In order to implement it a user builds a module in C that supports KOrganizer data format for contacts. The module consists of an event handler for the event that occurs when data input in that particular format is about to happen. The DCMS event-driven interface allows integrating this module into NICS without any modification outside this new module (module A). Block B1 is represented by contact cards. Among the samples of system modules that are being added to NICS the following ones can be mentioned: a module for network protocol support for interaction with a remote server, a docking module with Java for the execution of Java procedures in NICS, a module for generation of VBScript code fragments, which improves appearance and interactivity of Web pages when generated for MS Internet Explorer.

The second variant is inclusion of imported modules into NICS (forum, Jabber, Wiki server, etc.). It also has two ways (3, 4). In either case no modifications of imported source text are required. Let us consider the third way. The NICS developers have implemented a number of docking modules and have inserted those into the kernel. A docking module makes the interfaces of the imported module visible within the DCMS environment. The structure of a docking module is typical. That means that an existing docking module can be used with a few modifications instead of writing from the scratch when adding a newly imported module. The fourth way is based on using existing open source solutions that are integrated into the system without modifications. Such solutions are further used in the system as miscellaneous services, for example, for the support of synchronous and asynchronous communications between the users of the system (Jabber, IRC, forums). They are visible as separate elements of GUI or Web interface for the users.

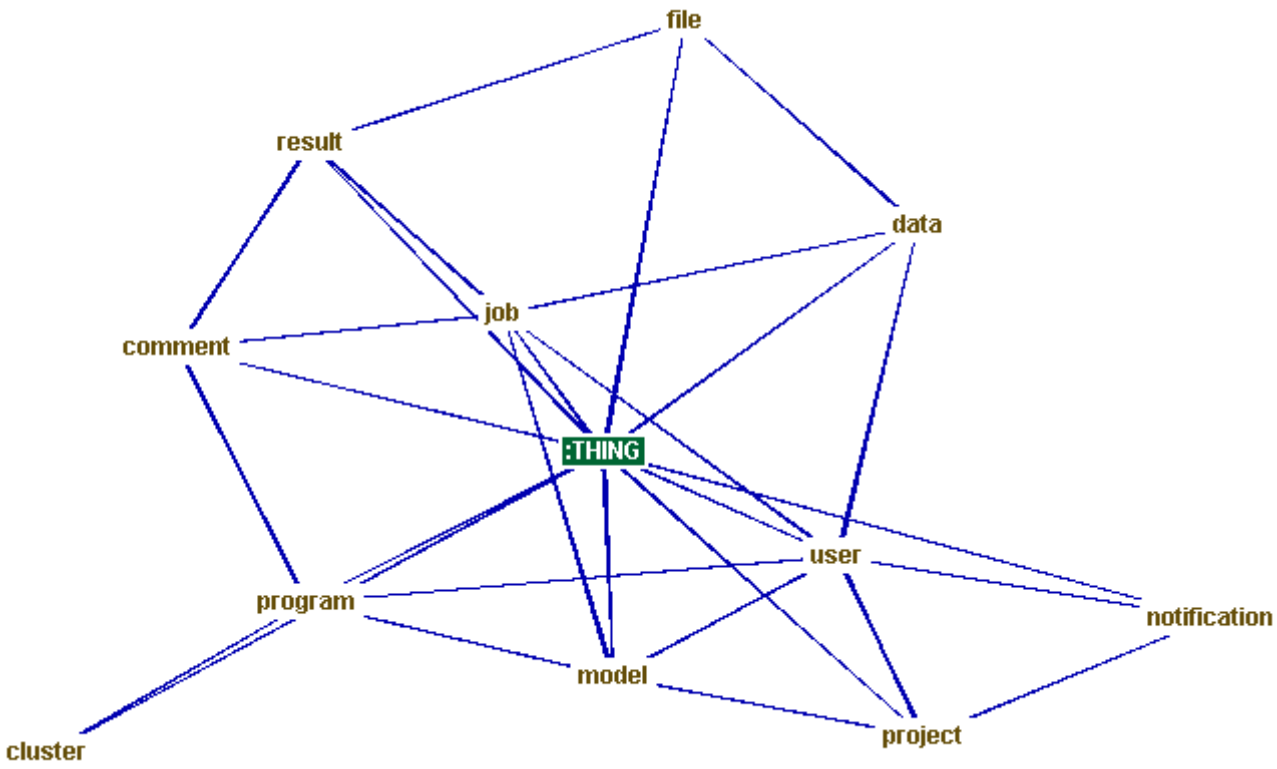

Fig. 2. A fragment of ontology

The variants of module construction and application assembly on their basis that were discussed above can be called “classical”. They follow the most widely spread way of application development, that includes writing of a specification, designing data structures and coding in one of programming languages. After the completion of development the application is ready to be used, but it can only handle a predefined set of kinds of objects. Thus when adding new kinds of objects, the application has to be modified and recompiled. The processing of the existing kinds object can be altered using their descriptions in a form of templates. But in either case a programmer’s assistance is required.

Meanwhile, a problem lies in fact that when developing NICS on one hand it is impossible to describe all the kinds of objects of innovation activity in all domains a priori, while on the other hand it is important not to lose such potential users of the system, who would like to adapt NICS for their problem domain while not being themselves skilled programmers. Thus it seems to be evident that there is a need for one more variant of module construction, the one, which could be used not only by programmers.

This (third) variant is based on the model-oriented approach [8] to the construction of applications. It is directed toward “what to do” rather than “how to do”. The essence of the project is as follows. A high level description of the interface, its model, is built. Then a code is generated by this model using the assembly technology. Such an approach helps to decrease the consumption of time spent for the development of user’s interface and it also eases its further modifications, because a user deals only with a declarative description and doesn’t have to reprogram modules. The generation of code is done by a generator module, that takes high level description in a form of ontology at its input. The result of its work is a text in assembly language (for GUI interface) or in a markup language (for Web interface). Two implementations of the generator are built. The first one is made in a form of an external module based in NICS kernel (the fifth way). The second one is based on Perl and HTML (the sixth way). The third variant of module construction was used for building of NICS user’s interface that unites two ontologies: an ontology of innovation process [6, 7] and an ontology of user’s interface. RDF/RDFS ontology description language and Protégé ontology editor were used. The ontology query processing is based on freeware Rasqal RDF Query Library and RDQL query language.

The ontology of innovation process is built with an orientation to serve as a specification of functions that a system must implement. It is extensible: it can be enriched by sub-ontologies that reflect the specific issues of certain domains. For example, the “simulation” node can be substituted by ontology of access to Siberian Supercomputer Center (SSCC) SB RAS resources, which was developed by NSTU master of science student A. Beloglazov. The objects, events and facts relevant to the simulating process are described in this ontology. The main objective of using ontology is the integration of heterogeneous data and the specification of simulating process. The impression of how the ontology is organized can be got by its fragment presented at Fig. 2. When embedding the ontology of access to SSCC into the ontology of innovation activity, class THING is substituted by a node SIMULATION. The ontology of graphical interface describes graphical representation of system objects. The description is represented by a graph. Its nodes denote primitives (indivisible elements of interface). The nodes denote relations between them. The samples of such primitives are a text string, a text input line, a cell of table, a string of table, a button, a checkbox, a reference, a menu item, a picture. The code generator builds a fragment of description of user’s interface for work with a data object by problem domain ontology and ontology of graphical interface. The blocks E and F at fig. 1 are the generated fragments of Perl and HTML texts and NICS assembly language respectively. The blocks E1, F1 are graphical windows.

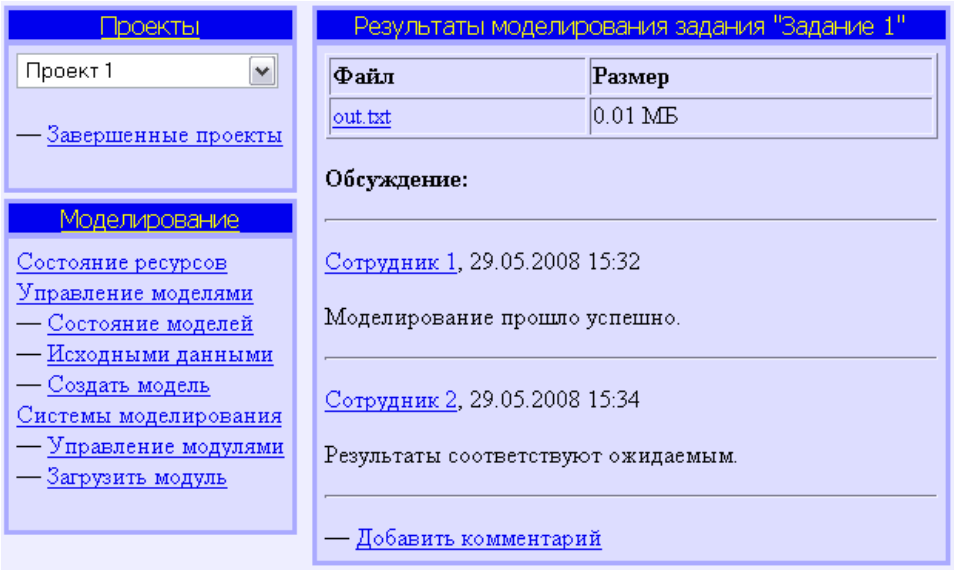

Fig. 3 Simulating window screenshot

From a practical point of view using the third variant means that a user gets a capability to form her work place, that includes a desktop, subwindows of object browsers and their catalogues in versatile modes (list, tree, calendar), subwindows that help to organize interaction with the resources of SSCC: subwindows of projects, uploading of simulating system modules, module recompilations, module management, etc. For example, one can construct a window with the results of a simulation, such as that depicted at Fig 3.

IV. CONCLUSION

The proposed assembly technology for the construction of NICS and especially its part that is based on the model-oriented approach, provide a user with a capability to install the system on own hardware platform, adapt it for own particular problem domain, information environment, use the resources of supercomputers for simulating single phases of innovation project, etc. A user can store modules developed by her to the module repository on the developer's server, enrich system's informational space by own information, etc.

V. REFERENCES

[1] Alekseev A.S., Ostapkevich M.B., Piskunov S.V. About the project of informational computational system for the support of innovation activity // Optimization. Control. Intelligence. – 2005 – vol. 2(10) – P. 203-210. (in Russian)

[2] Alekseev A.S., Kratov S.V., Ostapkevich M.B., Piskunov S.V. Siberian network system of innovation activity support //«Scientific service in Internet: multicore computer world. 15th anniversary of RFBR». The proc. of Russian scientific conf. (September 24-29, 2007, Novorossijsk). – M.: MSU, 2007. – P. 352-354. (in Russian)

[3] Piskunov S.V., Kratov S.V., Ostapkevich M.B., Veselov A.V. Distributed system of information and computational support of innovations // The Third International Forum on Strategic Technologies, IFOST 2008. Novosibirsk-Tomsk, Russia. - P. 325-328.

[4] Ostapkevich M.B. The DCMS library for open architecture application // Bull. Nov. Comp. Center, Comp. Science, 2004, iss. 21. - P. 99 - 111.

[5] Veselov A.V., Kratov S.V., Ostapkevich M.B., Piskunov S.V. The development of users' interfaces of network informational computational system on the basis of model-oriented approach. // Proc. ICM&MG SB RAS. Series: Informatics. Iss. 9. Novosibirsk. 2009.– P. 34 - 39. (in Russian)

[6] Veselov A.V., Ostapkevich M.B., Piskunov S.V. Automated generation of users' interfaces for network informational computational system // Proc. 7th Intl. Conf. Acad . A.P. Ershov «Perspectives of Informatics Systems». Seminar «Scientifically oriented software ». Novosibirsk, «Siberian Scientific Publishing», 2009. - P. 84–90. (in Russian)

[7] Veselov A.V. Using ontologies for the development of graphical users' interfaces. // Proc. Conf. Young Sci. – Novosibirsk: ICM&MG SB RAS, 2008. – P. 18-28. (in Russian)

[8] Gribova V.V., Kleshev A.S. Concept of the development of user's interface based on ontologies. // Messenger FEB RAS, 6, pp. 123-128, 2005. (in Russian)