UDK 621.38

O. Chkalov,
National Technical University of Ukraine «Kyiv Polytechnic Institute», Educational-scientific complex «Institute for applied system analysis»

B. Bulakh,
National Technical University of Ukraine «Kyiv Polytechnic Institute», Educational-scientific complex «Institute for applied system analysis»

O. Beznosyk,
National Technical University of Ukraine «Kyiv Polytechnic Institute», Educational-scientific complex «Institute for applied system analysis»

O. Matsulevych,
Tavria State Agrotechnological University,

THE FEATURES OF DESIGN ROUTE CREATION IN THE GRIDALLTED SYSTEM

This paper is devoted to the features of design route creation in the GridALLTED mathematical simulation complex. Task flow and route creation subsystem's main functions are given, the task route description format used for workflow building-up and executing by means of grid- and web-services is considered. The future investigation direction is denoted.

Keywords - GridALLTED, simulation, design route.

Introduction

Due to the technological progress the requirements to CAD systems' functionality and performance increase. The modern systems should deal with objects compound from hundreds thousand elements in a short time that is impossible to be done on a personal computer or a powerful server. In particular, such systems are rather expensive avoiding their usage in the everyday life by an ordinary user. Prices of some simulation packages can be within some thousand to tens thousand dollars resulting in decreasing of a number of potential users. Moreover, a serious problem when using software for designing systems is its big dependence on hardware, operating system, presence of additional software packages etc. A separate problem is a lack of a single package allowing simulation in different branches. To eliminate this disadvantages, a set of web-services should be created providing remote access to different program tools via a common web-interface involving usage of powerful grid-resources. Such system requires a mechanism which should provide connection and data transfer between different modules to execute tasks. One of such system is a complex for mathematical simulation GridALLTED.

GridALLTED

The development of the complex on a base of service-oriented architecture has a primarily goal to make use of the advantages of modern web- and grid-technologies when realizing a server part at the stage of solving tasks of computational resources search, intermediate data storage, supporting long-term calculations etc.

GridALLTED is built on a base of a client-server architecture with usage of web-services (as well as grid-services, that is web-services having access to the grid-environment) using SOAP+WSDL+UDDI standard group. The essence of this architecture is that the entire computational process has to be divided on a set of independent communicating web-services which can be distributed between different servers

(Fig. 1). An access to web-services takes place through the network using standard protocols as HTTP. As far as in the classic client-server architecture**,** in this case two separate part of the system could be marked out – server and client ones [1, 2].

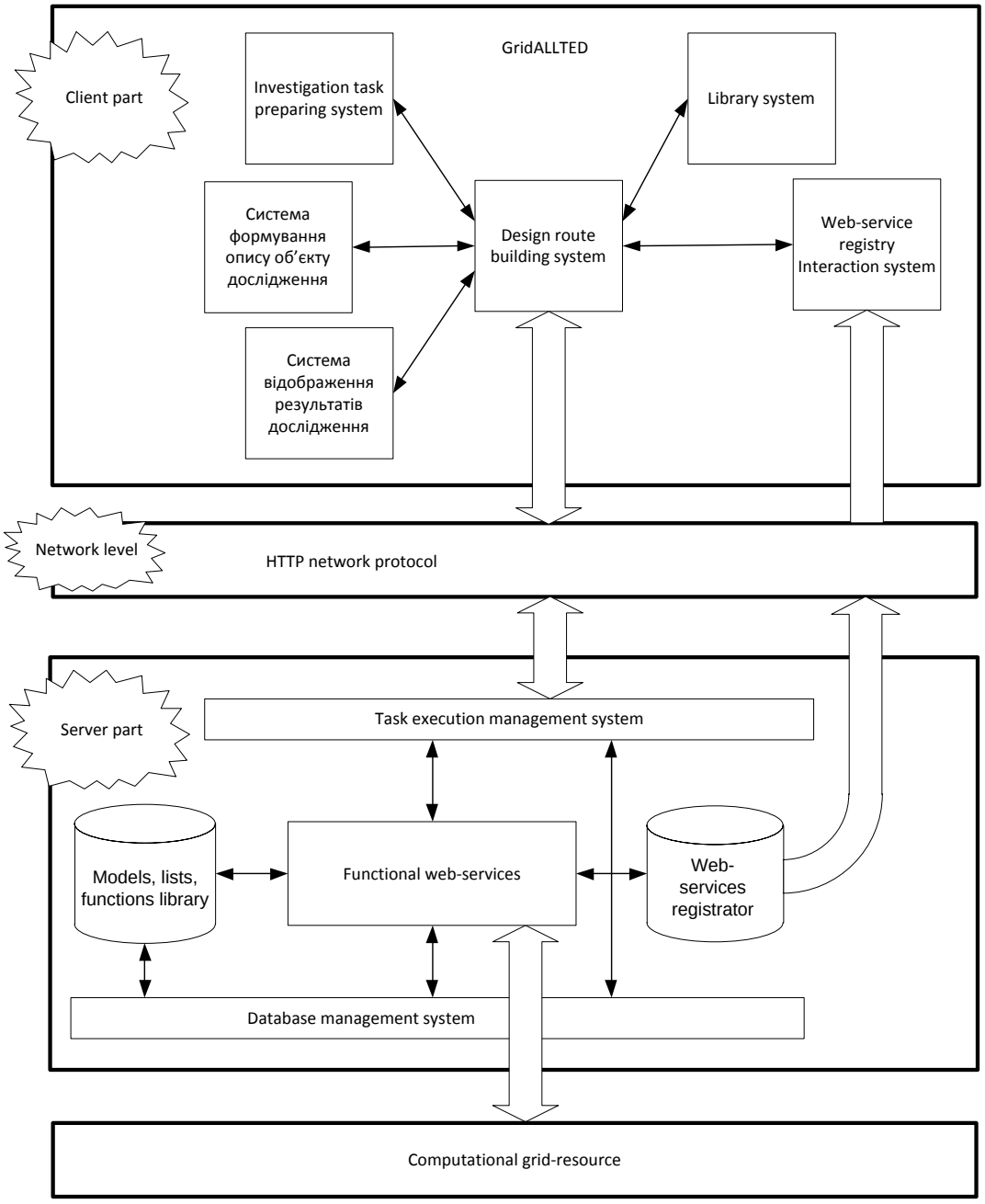

Fig. 1. GridALLTED interdisciplinary design complex architecture

The server part is an independent computational grid-system that not only plays a role of the computational resources for calculations but also provides searching for the resources required to solve a task, an organization of web-services communication and is a queuing system working with client tasks in a multitask mode.

As a platform-independent client part with the access interface to the task setting-up, execution control and displaying results systems a usual browser is used.

GridALLTED system client part is an interface between the «computational» part of the system and a user. The requirements posed to the client part are as follows: execution of the works required to prepare needed for simulation data in a form suitable for a user; execution of the works needed to transform simulation results in a form suitable for viewing and analysis of these data; execution of the works needed

on the components of the lists and functions models libraries; a friendly user interface; a high speed of network work; cross-platformness.

Workflows and design routes

One of the main components of the modern tools for organizing numerical experiments is the systems for organizing calculation cascades, well-known as workflow-systems [3, 4]. Such systems (workflow control systems) allow developing a plan (a route) of the computational experiment, its correcting dynamically according to the goals of an experiment and its intermediate results, controlling a course of the entire complicated computational process.

As a workflow (or flow/cascade/route of the works) let's understand further in a common case a set of steps, each of them represents a certain action (calculation), as well as transitions between them, which organize execution of these steps in a certain order.

Taking into account how the transition semantics is defined, such workflow subtypes as data flow and control flow are distinguished. A real workflow is their integration.

A control flow sets up an order of flow steps execution and is organized by means of various patterns such as branching, parallel execution, conditional transition, loop etc. Additional attention could be paid to the exception handling and steps to compensate actions done (it allows defining actions to cancel multistep transaction in a case of errors at its any stage). In a flow determining ways for data transfer, the main attention is paid just to the problems of data transfer: copying, scope, data formats matching etc (Fig. 2).

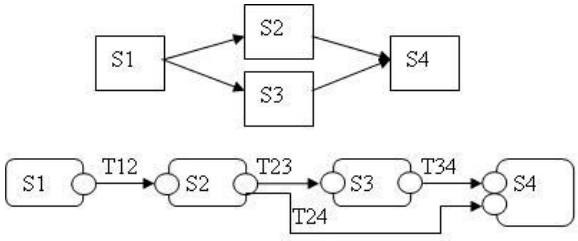

Fig. 2. Subtypes of workflows: control flow (up, first step plays a role of branching or conditional transition) and data flow (below, input-output data ports are shown)

As workflow computational steps, data analysis and handling steps, numerical simulation, optimization, visualization procedures can act making such an approach suitable for realization of engineering and scientific data analysis and simulation systems.

As a rule, the workflow management system building up is impossible without formalization of a flow within some model and its description in a certain language where the allowable flow elements (from the functional, behaviour, informational points of view) and their semantics are clearly defined. Both flow features analysis and their simulation are impossible without binding to the specific flow model as well as a flow designing itself is impossible without binding to the language describing a model selected.

A flow formalization that fixes their semantics and allowable elements is an important condition to create automated environments for flow designing since it allows:

- computer-aided flow design based on the primitives (language elements) in the specialized graphic or text editors;
- computer-aided flow analysis on a common effectiveness, a fault probability, a presence of design defects (simulation);
- computer-aided execution of the flows created due to the semantics defined.

Using workflow-approach in the computer-aided design systems can take the form of «design routes», which consist of a set of data input operations, mathematical algorithms, result handling etc.

In the frame of creating an interdisciplinary complex of mathematical simulation based on the usage of a service-oriented architecture, the computational experiment route (or, design route) building system is one of the key ones. The analogues of such systems can be business-flow building systems but in a case of

using computer-aided design systems the building of such systems has a number of features involving both a composition of route elements and an interaction with other complex's subsystems.

The structure and the functional possibilities of the proposed complex's subsystems are directed just on the creation of «design routes » [5] allowing representing designing process as a set of the interacting web-services and allowing a user to select most successful ways to solve a task.

According to the conception proposed, users of the complex can make and adjust a numerical experiment workflow themselves using existing elements з (which represent different types of analysis, operation on the mathematical models etc), and send it then to the automatic execution.

The specifics of a route comprised by executable modules (web-services) for a circuit design system is that a certain order of actions is possible only on the assumption of the fulfilment of previous steps, for instance, forming a task for investigation mainly has no sense if an investigation object description has not been done before etc.

Route building subsystem

The route and task flow building subsystem is intended for visualization of the task route creation process by the GridALLTED mathematical simulation interdisciplinary complex graphic interface followed by its transformation into a route in XML description language and error prevention in a route description code as far as at the stage of its creation (Fig. 3).

The main functions of the route and task flow building subsystem is:

- investigation object description text generation based on a visually created route;
- execution flow automatic creation;
- route check for non-allowable connections.

The route editor is in effect a «light» editor of directed graphs that could be used as a module as a route and task flow building subsystem. A user is supplied with a route graphic editor allowing building design routes out of registered components; a composition of the route component library could be easily expanded. Adding an element to the route is being made by adding a graphic representation of the respective component and establishing connections to define the passage order. By default, a route element selected is non-active. To make the element active, it is necessary to input the parameters required depending on the element selected. Having all the route's elements connected and activated a route status becomes «possible to be executed ». The route created is keeping in the database, is binding to the certain user project and could be used repeatedly.

The complex, branched routes suppose parallel usage of different route branches, probably, even on the different physical resources. So, even such project procedures could be executed simultaneously, which usually could not be executed sequentially, that is for their execution two independent routes should be composed та executed one after the other.

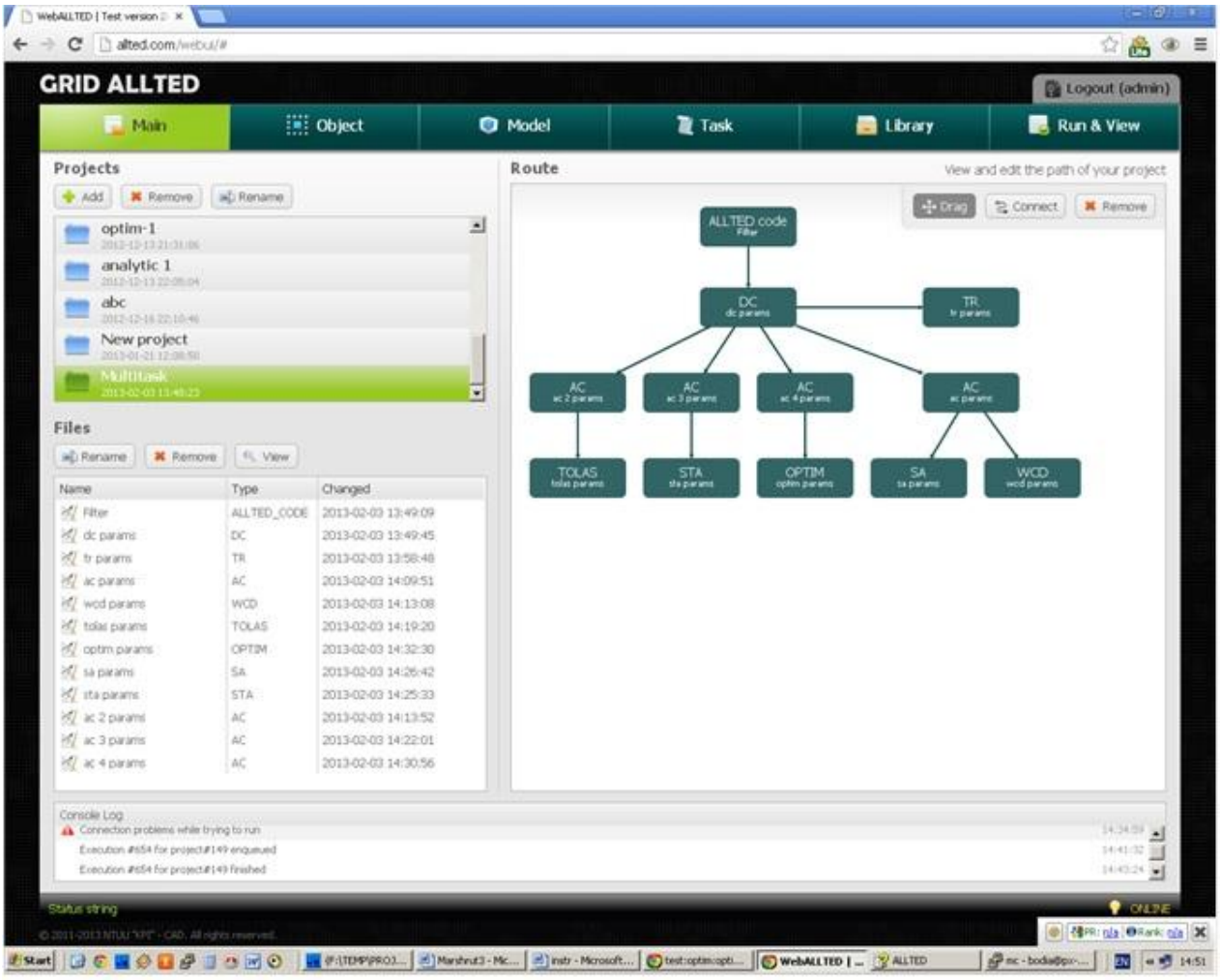

Fig. 3. Calculation experiment route example

Execution task description format

One important issue is data exchange format used in communications between access layer (with graphical workflow editor) and workflow execution layer (based on workflow management system). There are some common requirements to that format:

- it should be platform-independent (because architecture layers may reside on different hosts with different architecture, OS etc.);
- it should be language-independent (because components of the distributed complex can be written by different developer teams in different programming languages);
- it should be compatible with service-oriented architecture of the complex (namely with SOAP web services).

The obvious answer to the requirements stated above is to use extensible markup language (XML) because it is a text format independent of specific language or platform and it is directly used in web services standards (SOAP, WSDL, BPEL are all XML-based formats).

XML strengths are in that: a) being machine processible it remains readable by human; b) it can describe complex data structures (lists, trees etc.); c) there are a lot of parsers for the most of programming languages. XML also has many connected standards devoted to XML structure definition (XSD), data transformation (XSLT) or querying (XPath, XQuery).

Anyway, XML has several weaknesses: its syntax is redundant (XML files usually bigger then binary ones or even files in other text formats like JSON), it is not well suited for transferring binary data (binaries should be converted to text, all such encodings like Base64Binary, HexBinary etc. increase file

size up to 4/3~2 times) and so on. But as soon as CAD/CAE data flow is mainly document-flow, XML can be considered as preferable choice for data exchange format [6, 7].

During the implementation of the GridALLTED complex it was decided that workflow description must be clearer and more laconic than WS-BPEL used for workflow execution. So internal simple XML-based format was developed to describe both the control and data flows and contain any necessary data attached inside. It is based on few basic workflow concepts depicted on Fig.4 (dashed lines belong to data flow, solid arrows describe control flow).

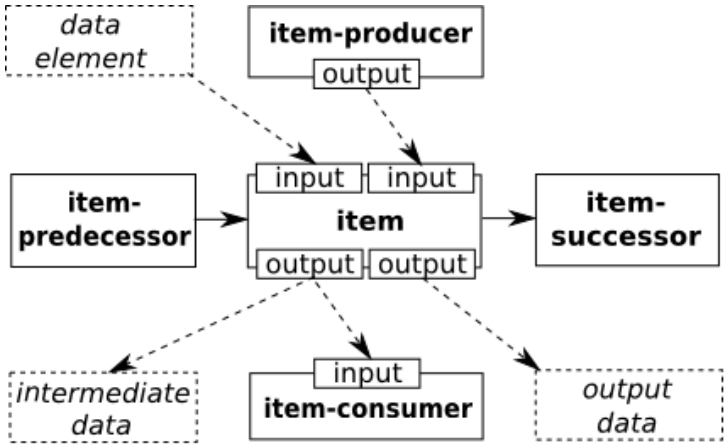

Fig.4. Basic entities and types of relationship in a workflow

Workflow-like XML-based task description format [8] provides unified description of the text and binary data exchange order as well as the action execution sequence in the engineering workflows. An XML model is being created at the stage of launching task on execution at the server side of a web-service. This file contains all the needed information on a route composition and configuration allowing interpreting definitely a route to be executed at the grid-environment.

Brief description of the format proposed and used in GridALLTED is presented below (“*” means “any number”, “?” means “optional single element”, “+” means “single or more”):

```
<task>
  <workflow>
    <item
      id="item_id"
      name="item_type_name">+
      <after
        item="pred_item_id"/>*
      <input
        name="input_name">*
        <from
          item="source_item_id"?
          output="source_output_name"?
          element="data_element_id"?/>?
        <from/>
      </input>
      <output
          name="output_name"
          check="true"?
          return="true"?/>*
    </item>
  </workflow>
  <data>
    <input
      item="item_id"
      name="input_name"
      encoding=”base64”?>*
```

```
        <![CDATA[input_data]]>
    </input>
    <element
      id="data_element_id"
      encoding=”base64” ?>*
        <![CDATA[input_data]]>
    </element>
  </data>
</task>
```

Elements in this basic structure have the following meaning. Each file describes single *task*. Task description consists of *workflow* and *data* parts. Workflow part describes involved workflow items and their interdependencies: execution order and data transfers. Each *item* represents an instance of registered item and has unique (for current workflow) identifier (*id* attribute) and the *name* of registered item. Execution order dependence is specified with *after* element (meaning that current item cannot be started before items which id's are in *after* block). *Input* and *output* ports should be specified for each item (but it is necessary to specify correct registered *names* for item's ports). Input ports can contain *from* element which is used to specify data source: if data should be copied from other item's output then *item* attribute must contain id of item-producer, *output* – its output to copy data from; if data should be copied from data element then only *element* 's id should be specified. Output XML element have several optional flags: *return* specifies whether data from this port should be returned as the part of workflow's output, *check* allows to return data before workflow finish (to control intermediate results). Data block can either contain initial data for specific items' *input*s or data *elements* that can be consumed by more than one item. Binary data should be base64-*encoded* and corresponding attribute should be set.

Significant attention by route design subsystem creation was paid to the development of the rules which define acceptability of service execution in a certain order. They have to be created by a system administrator by adding new services to provide interconnection between modules (an example of permissible connections between inputs and outputs of different services are shown in Table 1).

Table 1.

An example of the relationships allowed between inputs and output of different services

Input/output	Service 1	Service 2	Service …	Service N
Service 1	X	+	-	+
Service 2	+	X	+	-
Service …	+	-	X	+
Service N	-	-	+	X

At the practice, sometimes a problem appears to include new functional of third-part developers in a design route following to the impossibility to add input and output data representation in the formats required directly in a program module due to the closed source code of a program. In this case, a service has to be created that will be activated implicitly (that is, a graphic block corresponding to this service will be absent at a route scheme), which should contain rules to transfer output data of one module into an input data format of other module.

A perspective direction of the further design route creation functional improvement is a possibility to connect different branches of a route into a common element, in a case a further route execution should continue taking into account the best results out of that obtained at the previous stages.

Conclusion

A presentation of the design route as separate functional services is reasonable in the cases when there is a complicated calculation task (from a time-consuming point of view), or a service provides an additional (absent in the complex) functional (for instance, when interdisciplinary investigations take place) since using a large number of «little» services increases significantly an execution time.

The route description format developed contains all the needed information to provide successive (or, if needed, parallel) execution of separate services and data exchange between them and could be granted to third-part software developers to realize interaction interfaces of their program products with the existing services.

Involving into the complex new services with closed-to-changes codes is possible using implicit («auxiliary») services which should provide matching of new and existing services by input and output data. To eliminate conflicts connected with the possibility of the different services' interaction when their adding to a design route, new services have to be added with the rules of their interaction with other services.

1. Петренко А.І. Дослідження архітектури комплексу схемотехнічного моделювання GridALLTED / Петренко А.І., Ладогубець В.В., Фіногенов О.Д., Булах Б.В. // Вісник університету «Україна» : Інформатика, обчислювальна техніка та кібернетика. – К. : Університет «Україна», 2011. – № 2. – С. 65–70. 2. Петренко А.І. Архітектура мереженого комплексу схемотехнічного проектування ALLTED / Петренко А.І., Ладогубець В.В., Воєвода О.О. // Комп'ютерні системи проектування. Теорія і практика: вісник НУ «Львівська політехніка». – 2005. – № 522. – С. 30–33. 3. Yu J. A Taxonomy of Workflow Management Systems for Grid Computing / Yu J., Buyya R. // Journal of Grid Computing. – 2005. – 3, N 3. – P. 171–200. 4. I.J. Taylor. Workflows for e-Science. Scientific Workflows for Grids / I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (Eds.). – Springer. – 2007. – 530 p. 5. Petrenko A. Simulation in GridALLTED Complex / Anatolii Petrenko, Mykhailo Lobur, Volodymyr Ladogubets, Oleksii Finogenov, Bogdan Bulakh, Tetyana Ladogubets // The Experience of Designing and Application of CAD Systems in Microelectronics : 12-th Intern. Conf. «CADSM'2013», 19-23 February 2013, Polyana-Svalyava (Zakarpattya), Ukraine : proc. – Lviv, 2013. – P. 422–424. – ISBN 978-617-607-393-2. 6. Denysyuk P., Teslyuk V., Khimich I., Farmaga I. XML application for microfluidic devices description. //Proc. of IXth International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics, February, 20-24, Polyana, Ukraine, pp.567-569. 7. Kiselev G. Workflow Task Description Format for GridALLTED Complex / Gennadiy Kiselev, Bogdan Bulakh, Oleksandr Beznosyk, Vitalii Chekaliuk // The Experience of Designing and Application of CAD Systems in Microelectronics : 12-th Intern. Conf. «CADSM'2013», 19-23 February 2013, Polyana-Svalyava (Zakarpattya), Ukraine : proc. – Lviv, 2013. – P. 425–426. – ISBN 978-617-607-393-2. 8. Лобур М.В. Управлінням процесом проектування в середовищі розподілених САПР / Лобур М.В., Лебедєва О.О., Матвійків О.М. // Вісник НУ «Львівська політехніка»: Комп'ютерні системи проектування. Теорія і практика. – 2007. – № 591. – С. 16-21.