Computer and Information Science; Vol. 6, No. 2; 2013
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Program Factory – A Conception of Knowledge Representation of Scientific Artifacts from Standpoint of Software Engineering

Lavrischeva K.[1], Aronov A.[1] & Dzyubenko A.[1]

[1] Taras Shevchenko National University of Kyiv (KNU), Ukraine

Correspondence: Lavrischeva K., Taras Shevchenko National University of Kyiv (KNU), Ukraine. E-mail: lavryscheva@gmail.com; Aronov A. & Dzyubenko A., Taras Shevchenko National University of Kyiv (KNU), Ukraine. E-mail: asmer@asmer.com.ua

Received: November 19, 2012 Accepted: May 2, 2013 Online Published: May 7, 2013

doi:10.5539/cis.v6n3p21 URL: http://dx.doi.org/10.5539/cis.v6n3p21

Abstract

The factory of the programs is created by students of the cybernetics department at KNU. A primary objective of this factory is the electronic presentation of the “Software engineering” textbook, created based on international programs “Curricula-2001” and intended for mastering of this discipline by students and acquiring knowledge on these fundamental aspects. It is used for implementation by the students of laboratorials, diplomas and master’s works. Their algorithms and programs are developed at technological lines (TL) with the specifications in various programming languages (C#, Pascal, etc), WSDL, IDL and saved in repository for the prepared program products (artifacts and programs). Factory helps to study the textbook and create products that can be used by the other students at different universities from ready reusable components in the MS.NET system.

Keywords: artifacts, reuses, programs, applied systems, lines production, programs factory

1. Introduction

Over last decades a huge amount of various programs is stored in the informational world that may be used as end products over complex programs development. The new approach has been formed in programming, namely reusability – reuse of ready-made software resources (reuses, assets, services, components etc.) being from now on referred to as RC (reusable components). This term is used in informational world to represent new knowledge acquired over researches in some fields of e-science. Being needed for somebody, it may be used for some problems solving concerning similar artifact as well as for development of new software systems (SS), applied systems (AS) or families systems (FS).

All software artifacts and RCs may be stored in public warehouses (libraries, repositories) for reuse by various professionals through search to identify the necessary data about them for implementation in their own research activities. That is, reuse of ready-made resources becomes capital-intensive activity in the field of software engineering and therefore it is particularly important that the universities possess so-called factories for scientific artifacts, software and RC development that will be needed for other students and professionals. With these factories, students will participate in industry development of scientific artifacts for mass use (Lavrischeva, 2011; Aronov & Dzyubenko, 2011; Anisimov, Lavrischeva, & Shevchenko, 2011).

Based on such innovative ideas, Prof. K. Lavrischeva proposed fourth-year students at KNU to establish the first programs factory at the cybernetics faculty over the course of theoretical and practical labs in SE. This factory is focused on artifacts and programs development and storing them in factory repository.

1.1 Approach to the SE Studies

The SE educational course must include folded theoretical and practical fundamental positions and achievements in software technology and integration at complex SP development and in the information technologies. Directions of the SE course are as follows:

- Base conceptions, principles and methods, which make a basis of SE knowledge and technology of programming (for example, five SE disciplines, life cycle, project management, quality, configuration, etc.) and which proved productivity in practice;

- Systems analysis of subject domain with the use of elements of theory of algorithms, logic and semantics of programming aimed for the formal design of key notions of subject domain, reflection of their communications and relations in case of formal task of their models and SP architecture;

- General principles and methods of planning the programs, systems of the SP programs and their families FSP from the prepared objects, components, services, aspects and etc;

- Practical modern facilities of presentation of elements of the SP systems, which are regularly and widely used by many professionals in case of planning and constructing appendixes (systems analysis, decompositions, architecture, design, OOP, ontology and etc.) with the use of base principles and the SE methods;

- Achievement methods, measuring quality of software products and grant of technological interfaces of work with them;

- Instrumental systems (MS.Net, IBM, Corba, Protégé, Eclipse and etc.), in which modern conceptions, principles and methods are used to develop different types of SP and FSP, able to function in the modern operational environments.

One of the important tasks of our factory is study of base concepts, principles, methods, and tools of SE. That knowledge is needed at development of real SP, IS and FSP. For studies, there is a textbook that contains description of base conceptions, principles, methods and tools of SE. The textbook is given in electronic representation. Each section of the book has a list of questions and examples. While studying SE, students accomplish simple practical tasks, and create artifacts or programs. The completed products take place in the repository of the factory for use by other students.

Students' programs factory operates on the website (http://programsfactoty.univ.kiev.ua) since December 2011. Website was visited by more than 5,000 people – students, scientists and teachers.

2. Establishing Programs Factory

2.1 History and Backgrounds of USSR

An idea of industry for computers and their software has been proposed by Academician V. M. Glushkov at Cybernetics Institute of NAS of Ukraine in sixties or seventies. Under his guidance, the family of small computers "Mir" (1967-1975) has been developed with the language of analytical and formula transformations for the problems solving of differential, integral and formula calculus. In addition, other computers (Dnepr, Dnepr-2, Kyiv-67, 70, macro-conveyor etc.) have been elaborated with auto code-typed programming languages to develop information processing programs and automated management systems (AMS) for various organizations and enterprises. In order to assist in their development, the problems of software quality and increase of production amount for possible reuse have been investigated at the state institutions with the help of computers (Glushkov, Stogniy, & Molchanov, 1971; Glushkov, 1980; Kapitonova & Letichevsky, 2003).

In 1975 V.M. Glushkov at the scientific seminar of Cybernetics Institute first formulated the concept of assembling conveyor being constituted with technological lines for software products at program factories. The core of this Glushkov's paradigm is to accelerate the transition from the programming art to industrial methods of SP production for automated solving of various economical, business, scientific problems with Automatic Management Systems - AMS (АСУ in Russian).

Then the concept of programs as a part of scientific and technological production, as well as projects requesting for automating SP creation, have been decreed. The first software engineering factory was also established for mass production of various AMS (Kalinin, 1978). But because of lack of ready-made programs and immaturity of programming technology for industry, the factory lasted for two years and had been closed (Lavrischeva, 2008). Nevertheless, the experiments to elaborate programming automation tools were lasting until the collapse of the USSR.

V. Glushkov's idea concerning programs factories is now running in industrial factories explored by the authors theoretically in several papers (Czarnecki & Eisenecker, 2005; Bay, 2003; Greenfield, Short, Cook, & Kent, 2004; Lavrischeva, Koval, Babenko, Slabospitska, & Ignatenko, 2011; Lavrischeva & Grischenko, 2009) and implemented in practice in the experimental KNU factory, devoted to Glushkov (Lavrischeva, 2011; Aronov & Dzyubenko, 2011; Anisimov, Lavrischeva, & Shevchenko, 2011).

The authors of the KNU programs factory consider it as an integrated infrastructure for organizing production of mass usage software products, which are needed for customers and users from the fields of computer science, government, commerce etc. Factory is equipped with Technology Lines (TL) or Product Lines, a collection of products, tools and services needed for automated processes execution on these lines in modern operating environments (MS.Net, IBM, Sun Microsystems and so on) possessing necessary system tools.

2.2 Objectives for Programs Factory Site

The main objectives are: the system for exchange of KNU students' certified software products and scientific artifacts to improve students' skills; increase in quality, interoperability of their systems and SPs; and learning the methods of SS industrial production.

The website presents Life Cycle models, SP building lines, the line for program development with MS.NET platform and an example of a student program. Obligatory requirements are also done for SP's certification to store it in factory's repository (library pool).

The main activities on the factory web site are: organizing programs and artifacts development; students' familiarizing with tools and methods for program and SS development; representing students' SPs in the repository; citing articles and textbook materials concerning various disciplines for their further study by students and professionals. Each artifact is uniformly documented based on WSDL, the IDL standard being used in Grid global project.

3. Component Lines

3.1 TL for Development from RC

Technological lines are created at the technological pre-production stage (Lavrischeva, 2008a) that goes before SS production and includes designing TL flowchart from processes and actions that determine processing of SS elements. The basic requirement for TL engineering is to assemble TLs using lifecycle processes meeting problem domain goals, from standard tools, and the regulatory documents. Ready-made RC, tools and instruments are collected in the TL; they generate and implement specific functions or elements and management plan for processes concerning modification of the state of elements and quality evaluation (Lavrischeva, 2008b; Lavrischeva, 2011).

The *RC model* for component-based development has the following specification:

$$RC = \{T. I, F, Imp, \text{and } S\},$$

where T is type, *I* is interface, *F* is functionality, *Im* is implementation, *S* is interoperability service.

Basic operations upon components are:

– Specifying components and their interfaces (including pre- and post-conditions) in languages such as IDL, API, WSDL etc.

– Components, reuses and artifacts maintenance into the component repository for search, change and future integration into AS;

– Integration of RCs into applications, domains, applied systems, etc.

Applied systems is a collection of software means, including general tools (DBMS, protection systems, systems services, etc) containing subsystems or components with marshaling connections between them (Lavrischeva, 2008b).

3.2 Product Lines in SEI

Product lines and Product Family (PF) is defined in the dictionary ISO/IEC FDIS 24765:2009 (E) – Systems and Software Engineering Vocabulary: «*Product line* – a set of products or services that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Synonym: *product family*».

SEI professionals propose the two models to represent engineering activities for SS development, namely engineering and process ones.

Engineering model consists of the three activities, namely RC development, PF development through RC configuration and management of the both aforementioned activities (Lavrischeva & Grischenko, 2009).

Activity for RC development covers PF scoping and production plans accounting for the context of SS usage, production limitations and strategy. Activity for PF development via configuring RCs includes designing each SS implementation based on the set of developed RCs and developing SS according to the PF implementation plan. Management activity to elaborate PF from RCs is based on balancing of activities of RC development and includes both organizational and technical management.

According to the process model, a set of processes is performed at two levels, namely domain (or problem area) engineering, being also referred to as development «for reuse», and application (or SS) engineering – development «with reuse» (Lavrischeva, 2011). The last one is performed at the assembly line using ready-made

RCs to shorten time of development and increase SS availability. Therefore, configuring PF from RC according to specific requirements and needs of a particular market segment is the final activity in the process model cycle.

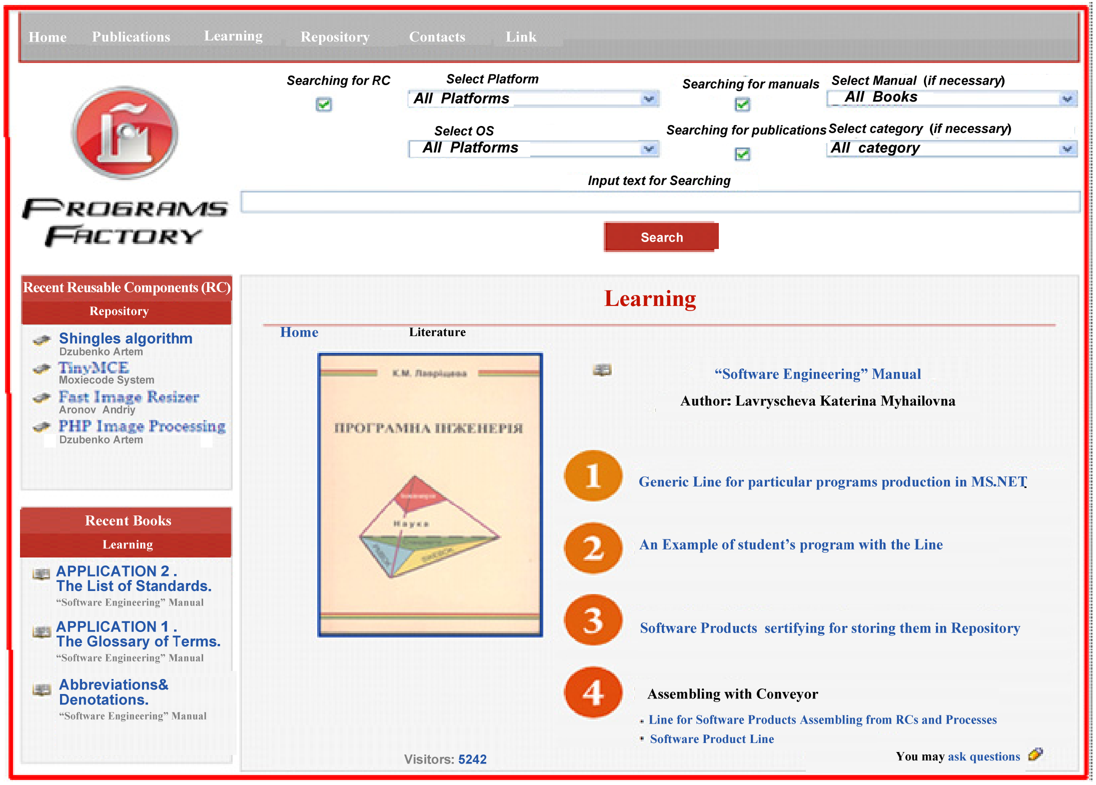
4. Glushkov's Conveyor Factory

From the theoretical standpoint, the factories are based on the assembling conveyor that includes a collection of various more or less complex production lines for software artifacts, programs and RCs. Conveyor lines contain execution of processes with system tools or TM that automate process execution to obtain interim or final result.

From the perspective of information technology, the factory provides a data processing toolset for a transition from individual programming of particular resources to the industry of mass usage SPs. The factory increases SP development productivity of each lifecycle process due to usage of RCs with quality guaranteed by developers. Assembling (composition, configuration) line reduces efforts, too, because of ready artifacts or RC being stored in the repository.

4.1 TL of KNU Programs Factory

The line of products development is as a rule matched with some lifecycle, e.g. implemented in MS.NET environment with guides, frameworks, programming languages, common libraries of templates, system tools that support new subject-oriented languages such as Domain Specific Language, etc. For complex SP development an *assembly* (compositional) line is proposed. It is the tool for composing RC sets using their interfaces from various libraries stored in development environments and in repositories of RC and interfaces (Figure 1). This figure shows the main page of the web site, which illustrates product lines 1, 2, 3, 4, and a text-book on fundamental aspects of software engineering (Aronov & Dzyubenko, 2011).



Recent Reusable Components (RC)
Repository

Shingles algorithm
Dzubenko Artem

TinyMCE
Moxiecode System

Fast Image Resizer
Aronov Andriy

PHP Image Processing
Dzubenko Artem

Recent Books
Learning

APPLICATION 2. The List of Standards.
"Software Engineering" Manual

APPLICATION 1. The Glossary of Terms.
"Software Engineering" Manual

Abbreviations& Denotations.
"Software Engineering" Manual

Learning

Home Literature

К.М. Лаврищева
ПРОГРАМНА ІНЖЕНЕРІЯ
Наука

"Software Engineering" Manual
Author: Lavryscheva Katerina Myhailovna

1 Generic Line for particular programs production in MS.NET

2 An Example of student's program with the Line

3 Software Products sertifying for storing them in Repository

4 Assembling with Conveyor
- Line for Software Products Assembling from RCs and Processes
- Software Product Line

Visitors: 5242

You may ask questions

Figure 1. Main page of the program factory web site (textbook «ПРОГРАМНА ІНЖЕНЕРІЯ» = SE)

Based both on the authors' practice acquired while building heterogeneous programs using multiple PLs and on the experience of modern industrial programs factories abroad (IBM, OMG, Microsoft, Oberon etc. (Greenfield, Short, Cook, & Kent, 2004)) the basics of programs factory may be listed as follows (Lavrischeva, 2011):

– Ready-made program resources (artifacts, programs, systems, reuses, assets, RC etc.);

– Interface as the passport data of ready-made heterogeneous resources written in some interface specification language (IDL, API, SIDL, WSDL, RAS etc.);

– Complex SP development method;

– Operating environment equipped with system software tools and instruments for industrial development of heterogeneous resources;

– Assembling conveyor including TL and Product Lines that enable automated or partially automated PF development.

The elements listed are the fundamentals of SP production industry at the factories that as a rule operate at huge foreign companies such as Microsoft, IBM, Intel, Apple, Oberon etc.

At our student programs factory, these fundamentals are implemented as several lines to represent both programs and scientific artifacts; the factory is among the best ones developed by students with standardized documentation and the factory repository.

There are four TL (see Figure 1) in the assembling conveyor (Lavrischeva, 2011a; Aronov & Dzyubenko, 2011) that successfully operate (since 2011) on artifacts and programs. Their structure is formed according to the TL elaboration technology explored by the author (Prof. K. M. Lavrischeva) over 1987-1991 (Lavrischeva, 2008a, 2008b, 2011b). These TL are:

– E-learning C# language in VS.Net environment (see the TL chart depicted in Figure 2);

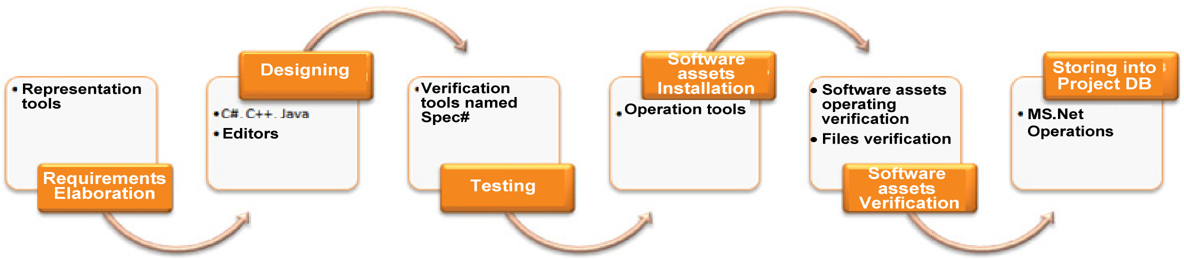


Figure 2. A flowchart for designing components in VS.Net environment

– Selecting RCs from the repository to meet market demands in special-purposed software;

– Configuring RCs into complex program structures;

– E-learning basic knowledge on software engineering by the students with dedicated e-textbook.

Depicted TL for learning C# programming is formed according to ISO/IEC 12207 generic scheme of lifecycle processes, allowing for VS.Net specifics. I is able to perform the following tasks:

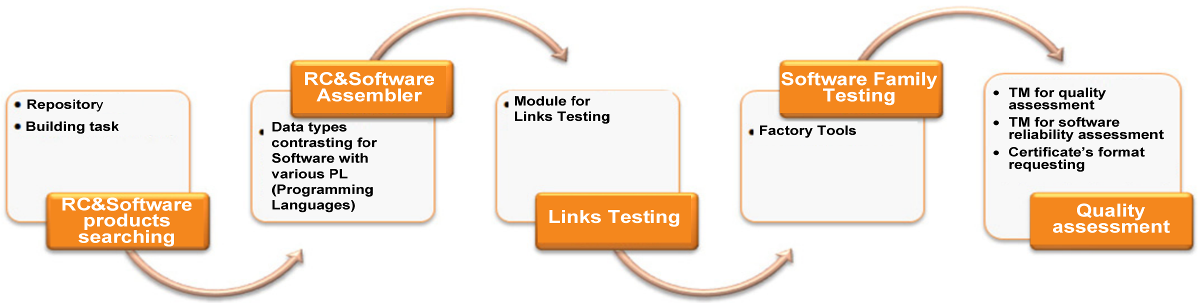
– Correcting requirements on SP functions according to the problem domain;

– Specifying passport data and interfaces for software elements or artifacts with a WSDL-like language;

– Artifacts or programs being designed and stored in the repository.

4.2 Lines on Website

The line for repository maintenance includes the mechanisms for storing RC with uniform documentation based on WSDL-written passports, as well as tools for extraction of ready-made RC and artifacts from the repository based on their passport data, functions and relevant solutions (see Figure 3).



Repository
Building task
RC&Software products searching
RC&Software Assembler
Data types contrasting for Software with various PL (Programming Languages)
Module for Links Testing
Links Testing
Software Family Testing
Factory Tools
TM for quality assessment
TM for software reliability assessment
Certificate's format requesting
Quality assessment

Figure 3. Flowchart of the line for component search

The line for AS development at the factory includes the tools for engineering specific required components, programs and artifacts using both just developed and selected from repository; it also contains options for multilingual RC development and comparison of non-relevant types of data being exchanged between RCs (Lavrischeva, Koval, Babenko, Slabospitska, & Ignatenko, 2011). The core of this line processes includes standard tools for building or configuring multilingual components, testing both sample components and links between RC, and means for connecting TM for reliability, quality assessment and certification of the product obtained. Such connections are implemented in CORBA, COM, and VS.NET under the supervision of ORB broker or middleware.

The line for remote e-learning with electronically issued dedicated textbook «Software Engineering» (Lavrischeva, 2008a, 2008b) is now actively used in practice when studying the course topics of this discipline by KNU students. This line may also be used for independent studying of the discipline in other high school institutions where «Software Engineering» or «Computer Science» specialties are established.

In the aspect of teaching SE, the first textbook, written in Ukrainian (Babenko & Lavrischeva, 2001), is devoted to the foundations of SE taken from Curricula-2001; the second textbook in Russian teaches the basics of Curricula-2004 (Lavrischeva & Grischenko, 2009); the third textbook is developed for modern approach towards teaching SE (Lavrischeva, 2008b), including topical outlines of some of the above-mentioned disciplines and fundamental aspects such as reliability and quality engineering for AS, SP, FS. In the new textbook, basic elements and engineering tools are presented for the development of various target SE objects and LS processes, methods for design and management of collectives of executors, quality, terms, and cost. The website www.sestudy.edu-ua.net describes new SE disciplines and several fundamental aspects of SE (Lavrischeva, Ostrovski, & Radetskyi, 2012). The structure of the textbooks corresponds to the typical Curricula-2004 program, modern requirements on the subject presented in the program of the Ministry of Education and Science of Ukraine (2007).

At our factory, VS.NET edition licensed for KNU is used as the basis of operating environment for programs factory; MS.Net platform capabilities for multilingual AS development (C #, C++, VBasic etc.) and for team development are utilized, as well. Consequently, an external software developer may not be limited to the choice of a certain PL but may use different PLs over the same AS.

4.3 Designing Factory Programs

The factory Website is developed with popular PHP, and, for its external representation, HTML5, CSS3, and JavaScript languages. The system core is self-dependently engineered by the authors with no well-known counterparts (Aronov & Dzyubenko, 2011).

5. Conclusion

The result of the authors' work is an experimental programs factory website, accessible using address http://programsfactory.univ.kiev.ua/. There are many suggestions about elaboration of student programs factory that will be implemented further. The website is proposed for e-learning of the lines establishing at the high school institutions on the specialties of Informatics, Computer Sciences, Information Systems and Technology.

The following concrete lines are established at the factory:

(1) The repository for RC and artifacts;

(2) Development with C# in VS.NET: console applications, DLL component libraries, and local Windows applications;

(3) Programming Java programs (manual by I. Habibulin);

(4) Assembling ready programs and RC systems in MS.Net environment;

(5) E-learning the software engineering course with dedicated textbooks in Ukrainian and in Russian.

For a short time quite a lot of various users, foremost students, addressed the factory site.

The prospects of future factory evolution are its further adjunction with new materials in the field of software engineering being prepared by the students, namely:

(1) Description of generative development of complex programs and SS with DSL language (Eclipse-DSL, Microsoft DSL Tools);

(2) Transformation of general data types into fundamental data types from the perspective of the standard ISO/IEC 11404-2007 generation tools;

(3) Ontological representations of new disciplines for study (e.g. computational geometry, life cycle domains, verification etc.);

(4) New applied product lines for business designed using SEI Product Lines approach and so on.

Acknowledgments

The article is devoted to the 90th anniversary of Academician V. M. Glushkov. He founded the Institute of Cybernetics of National Academy of Sciences of Ukraine (1962), the cybernetics faculty in Taras Shevchenko National University of Kyiv (1969). He passed away in 1982.

References

Anisimov, A., Lavrischeva, E., & Shevchenko, V. (2011). *On Scientific Software Industry* [in Ukrainian]. Technical report, Conf. Theoretical and Applied Aspects of Cybernetics.

Aronov, A., & Dzyubenko, A. (2011). Approach to Development of the Students' Program Factory [in Ukrainian]. *Problems in Programming, 3*, 42-49.

Babenko, L., & Lavrischeva, K. (2001). *Foundation of Software Engineering* [in Ukrainian] (p. 269). Znannia, Kiev.

Bay, Y. (2003). *Applications Interface Programming Using Multiple Languages: A Windows' Programmer's Guide*. Prentice Hall Professional.

Czarnecki, K., & Eisenecker, U. (2005). *Generative Programming: Methods, Tools, and Applications*. Addison Weasley.

Glushkov, V. M., Stogniy, A., & Molchanov, I. (1971). *Algorithmic small electronic digital computer.* MIR. Vol. 11. Naukova dumka.

Glushkov, V. (1980). *Basic research and technology programming* [in Russian]. *Programming, 2*, 3-13.

Greenfield, J., Short, K., Cook, S., & Kent, S. (2004). *Software Factories: Assembling Applications*. Wiley.

Kapitonova, U., & Letichevsky, A. (2003). *Paradigms and ideas of Academician Glushkov* [in Russian]. Naukova dumka. p. 355.

Lavrischeva, E. (2008a). Classification of Software Engineering Disciplines. *Cybernetics and Systems Analysis, 44*(6), 791-796. http://dx.doi.org/10.1007/s10559-008-9053-5

Lavrischeva, E. (2008b). *Software Engineering* [in Ukrainian]. Akademperiodika, p. 319.

Lavrischeva, E. (2008c). *Formation and Development of the Modular-Component Software Engineering in Ukraine* [in Russian]. Institute of Cybernetics after V. Glushkov., p. 31.

Lavrischeva, E. (2011a). Concept of Scientific Software Industry and Approach to Calculation of Scientific Problems [in Ukrainian]. *Problems in Programming, 1*, 3-17.

Lavrischeva, E. (2011b). Theory and practice of software factories. *Software–Hardware Systems, 47*(6), 961-972.

Lavrischeva, E., & Grischenko, V. (2009). *Assembly Programming. Basics of Software Industry* [in Russian] (p. 371). Naukova Dumka (2nd ed.), Kiev.

Lavrischeva, E., Koval, G., Babenko, L., Slabospitska, O., & Ignatenko, P. (2011). *New Theoretical Foundations of Production Methods of Software Systems in Generative Programming Context* [in Ukrainian]. Electronic monograph. In DIUK-2011, p. 277. Retrieved from http://www.nbuv.gov.ua/

Lavrischeva E., Ostrovski A., & Radetskyi, I. (2012). *Approach to E–Learning Fundamental Aspects of Software Engineering*. In 8th international Conf. ICTERI. Retrieved from http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR- WS-paper-17-p-176-187

!&