**Bardus Irina. Determination of the regularities of the software evolution of computer engineering as the condition of the fundamentalization of professional training of future specialists in the field of information technologies.**

Improving the quality of the training of future IT professionals to productive professional activities requires fundamentalization of computer disciplines on the basis of general methodological knowledge and skills. One of the conditions for the fundamentalization of professional training of IT professionals is to determine the laws of software development. All software, depending on its purpose, can be divided into four categories: system, instrumental, middleware and applicative.

In order to effectively training of future IT professionals for productive professional activities, it is necessary to develop a unified system approach to presenting the teaching material of various computer disciplines, which will allow students to develop the fundamental methodological knowledge and skills in creating new software.

For the systematic description of the concepts of computer discipline it is expedient to apply a universal hierarchical model of the technical object P = {R, S, D, H} based on semantic features (purpose (R), structure (S), operating principle (D) and characteristics (H)). As the software is a subsystem of the technical system "computer", so its improvement is based on the principle of progressive development of technical systems. Each new generation of systems is improved by improving the criteria (H). It is advisable to determine the laws of software evolution on the basis of comparison of quantitative and qualitative indicators of functional, technological, economic and anthropological criteria.

Retrospective analysis of the software should be based on the causes of each generation appearances (indicators to be improved), indicating the new structure, the operating principle (mathematical algorithms, laws and theories), or the purpose. Comparing the dynamics of the indicator over the entire time of evolution will determine the laws of software development.

A retrospective analysis of all software allowed to determine the general laws of software evolution, namely:

* increasing the amount of data processed by the software;
* increase functionality as the product develops;
* source code complication, due to the gradual increase of the software product functionality, as well as the support of new paradigms and software engineering technologies;
* increasing software volumetric capacity by expanding the functionality and increasing the amount of data processed;
* increasing the execution time of programs (when comparing old and new software on a single hardware platform), due to the increase in data that needs to be processed, the growth of software code volumes, and increasing requirements for the hardware.

Retrospective analysis and software evolution patterns will be used in developing the mathematical foundations for the fundamentalization of the professional training of future IT professionals.

**References**

1. Batovrin, V.K. (2012), Tolkovyy slovar' po sistemnoy i programmnoy inzhenerii [Explanatory dictionary on system and software engineering], DMK Press, Moscow, Russian Federation, 280 p. [in Russian].
2. Vasil'yev, V.G. (2009), Vvedeniye v sistemnoye programmnoye obespecheniye [Introduction to system software], TGTU, Tver, Russian Federation, 160 p. [in Russian].
3. Golubenko, A.L. (2004), Teoriya tekhnicheskikh sistem [The theory of technical systems], Aristey, Kyiv, Ukraine, 239 p. [in Ukraine].
4. Ditrikh, Y. (1981), Proyektirovaniye i konstruirovaniye: sistemnyy podkhod [Projecting and constructing: the system approach], Mir, Moscow, Russian Federation, 456 p. [in Russian].
5. Zayonchik, V.M. (2004), Osnovy tvorchesko-konstruktorskoy deyatel'nosti: Metody i organizatsiya [Fundamentals of creative and design activity: Methods and organization], Izdatel'skiy tsentr “Akademiya”, Moscow, Russian Federation, 256 p. [in Russian].
6. Lazaryev, M.I. (2003), Polisystemne modelyuvannya zmistu tekhnolohiy navchannya zahalʹnoinzhenernykh dystsyplin [Polysistemic modelling of content of training technologies of all-engineering subjects], Vyd-vo NFaU, Kharkiv, Ukraine, 356 p. [in Ukraine].
7. Lipayev, V.V. (1990), Proyektirovaniye programmnykh sredstv [Designing software], Vysshaya shkola, Moscow, Russian Federation, 302 p. [in Russian].
8. Soroka, K.O. (2004), Osnovy teoriyi system i systemnoho analizu [Basic theory of Systems], KHNAMH, Kharkiv, Ukraine, 291 p. [in Ukraine].
9. Khubka, V. (1987), Teoriya tekhnicheskikh sistem [The theory of technical systems], Mir, Moscow, Russian Federation, 202 p. [in Russian].
10. Selyutskiy, A.B. (1991), Shans na priklyucheniye [The chance of adventure], Kareliya, Petrozavodsk, Russian Federation, 304 p. [in Russian].
11. [“What is Middleware?”](https://web.archive.org/web/20120629211518/http:/www.middleware.org/whatis.html) [Electronic resource]. – Access mode: web.archive.org/web/20120629211518/http:/www.middleware.org/whatis.html
12. Workshop on Software Development Tools for Petascale Computing, 1-2 August 2007, Washington, DC [Electronic resource]. – Access mode: http://www.csm.ornl.gov/workshops/Petascale07/