UDC 004.94

Taras Bivoino[1], Senior Lecturer of the Department of Information Technology and Software Engineering
ORCID ID: https://orcid.org/0000-0001-6914-5441 ResearcherID: B-7478-2017
e-mail: tbivoyno@gmail.com

Dmytro Lysenko[1], Doctor in Technical Sciences, Professor of the Department of Information and Computer Systems
ORCID ID: https://orcid.org/0000-0001-6870-6120 ***e-mail:*** lysenko.d@stu.cn.ua

Pavlo Byvoino[1], PhD in Technical Sciences, Associate Professorof the Department of Information and Computer Systems
ORCID ID: https://orcid.org/0000-0001-8145-8459 ResearcherID: R-7447-2016
e-mail: p.g.byvoino@gmail.com

Nataliia Sokorynska[2], PhD in Technical Sciences, Deputy Head of the Communications Department of the Administration of the State Special Transport Service, Colonel
ORCID ID: https://orcid.org/0000-0001-9713-7289 ***e-mail:*** sokor-nata@ukr.net

[1]Chernihiv Polytechnic National University, Chernihiv, Ukraine
[2]State Special Transport Service, Kyiv, Ukraine

COMBINING OBJECT-ORIENTED PROGRAMMING AND SIMULATION IN EDUCATION

Abstract. *The article discusses the integration of simulation modeling and object-oriented programming (OOP) in IT education. The authors argue that closed-source commercial tools (Simulink, GPSS, Simio) limit the educational process by hiding the internal system architecture. As a solution, the "Simulation" open-source Java framework, developed at Chernihiv Polytechnic National University, is presented.*

The framework is based on discrete-event simulation and the "active object" concept. The core element is the Actor abstract class, which defines object behavior through the rule() method. Model time management and process synchronization are handled by the Dispatcher class, which utilizes a queue-based system to process events and validate logical conditions via lambda functions.

The educational curriculum is divided into stages: studying random number generators (Uniform, Norm, Erlang), statistical data processing, and building queuing system (QS) models. To analyze simulation results, students employ components such as ExperimentManager for automated factor analysis and TransprocessManager for studying transient processes. Special emphasis is placed on the object-oriented analysis of real-world systems, such as soil extraction logistics.

The study demonstrates that the open-source nature of the "Simulation" framework enables students to explore the internal implementation of complex software, design patterns, and pseudo-parallelism mechanisms. This approach ensures continuity in programming training and enhances the development of professional IT competencies. The project is hosted on GitLab.

Keywords: *simulation modeling, object-oriented programming, Java framework, queuing systems, discrete-event simulation, active objects, IT education.*

https://doi.org/10.32347/2411-4049.2026.1.225-234

Relevance of the study. Mastery of simulation tools and the ability to build appropriate models is essential for training developers of modern IT systems, the complexity, dynamism, and scale of which are constantly increasing. Simulation modeling is a research method based on the fact that the system under study is replaced by a simulation model and experiments are conducted with it in order to obtain information about this system without resorting to experiments on a real object. Modeling is widely used in the study of systems of various nature. For IT students, the greatest interest is the simulation modeling of queueing systems (QS) [2], [3]. Developing a simulation model in a programming language "from scratch" is a rather difficult task; however, the models development time for complex systems can be reduced by using existing effective modeling tools.

Target setting. A significant disadvantage of using existing effective modeling tools in the educational process is the lack of access to the source code, which does not allow using them as examples of the large-scale software systems structure and implementation in the training of IT specialists. The ability to solve problems that go beyond the scope of these tools is also a problem. We propose to use an approach in the educational process in which students not only use a closed software product to implement their own models, but also have access to the source code of the package.

Actual scientific researches and issues analysis. For the implementation of modeling, various concepts of formalization and structuring are used, the choice of which significantly depends on the systems under study. MathCad [4] can be used to solve differential equations. Simulink [5] makes the modeling process more understandable and accessible through the use of block diagrams. However, GPSS [6] is better suited for modeling systems consisting of objects interacting in time. A powerful tool is the Simio platform [7], which is used for discrete event modeling with built-in support for both simulation and planning.

Identification of previously unsolved parts of the general problem. While the analyzed tools are efficient for engineering tasks, they are insufficient for comprehensive IT education. The gap in current research and practice is the absence of an educational framework that combines practical simulation modeling with deep immersion into the source code implementation. This limits the formation of essential object-oriented thinking and the understanding of real-world software architecture necessary for modern IT specialists.

The research objective. The objective of this research is to enhance the quality of the IT educational process by ensuring the continuity of programming training and the formation of object-oriented thinking. This is achieved through the development and implementation of an open-source simulation framework in the curriculum, allowing students to study the internal structure and implementation of tools simultaneously with their practical application.

Presentation of the main material. The framework "Simulation" was created at Chernihiv Polytechnic National University to support the educational process in the disciplines "System Modeling" and "Object-Oriented Programming", which are closely related in the structure of training specialists in computer engineering. This connection is due to the fact that the department consistently implemented the principle of "to acquaint students not only with the use of packages, but also with their internal structure and implementation". The simulation course was ideally suited for the implementation of this principle, because simulation modeling is organically connected with the object-oriented approach to design and programming.

In accordance with the outlined idea, the developers formulated the following requirements for the framework, which should ensure the solution of the task of mastering the methods and means of simulation modeling in the educational process:

– the ability to create active transactions (which have their own rules of action);

– the ability to delay the execution action rules for a certain time or until the corresponding conditions are met;

– the presence of components that correspond to the main elements of the queueing system model;

– the presence of random variables generating means and statistical data processing;

– the presence of means for automating experiments with the model;

– the presence of special tools for analyzing transient processes;

– the openness of the code.

To implement the requirements, a library of interfaces and classes, including abstract ones, has been developed in the Java programming language. This library is similar in functionality to the Simula language [8, 9]. The library [10] uses the concept of discrete simulation, which assumes that the state of the system can change only at the moments of completion of events, and is based on the description of processes that are focused on transaction processing or simulate the behavior of active transactions.

To model objects that operate in parallel over time, the framework implements the concept of an “active object.” Such an object can be a service device, an agent, or even a transaction provided it exhibits its own behavior over time. To create such objects, the abstract class Actor is used. Subclasses of this class must provide an implementation for the rule() method, which defines the object's logic over time.

The Actor class provides access to methods that provide a delay for some time in the action rules execution, or a delay until the execution of a certain condition, which is described in a lambda function form of the standard BooleanSupplier type. You can also use a combined delay method that provides a delay until the execution of the corresponding condition, but not more than a specified time. The condition and the object's activation time are stored in the corresponding fields of the active object.

Below is an example of the FinishDevice class, which represents objects that complete transaction processing. The action rules include waiting for a transaction to appear in the queue and delaying the transaction processing time.

```
 public class FinishDevice extends Actor {
//Simulation duration
private double finishTime;
//Transaction queue
private QueueForTransaction inputQueue;
//Processing time generator
private Randomable rnd;
//Waiting condition
private BooleanSupplier isTransaction = ()-> inputQueue.size()>0;
//Active object action rules
protected void rule() {
  //A loop while the duration of the simulation
  while (dispatcher.getCurrentTime()<=finishTime) {
    waitForCondition(isTransaction, "must be a transaction");
    //Removing a transaction from the queue
```

```
      queue.removeFirst();
      //Transaction processing delay
      holdForTime(rnd.next());
    }
  }
}
```

Each active object has access to an object of class Dispatcher. This object is responsible for advancing the model time (currentTime) and for synchronizing the action rules of the active objects. The dispatcher has three queues. The Ready Queue stores active objects that are ready to start executing their action rules. The Condition Queue contains objects that have suspended their action rules until the condition they are storing is met. The Time Queue contains objects that have temporarily suspended their action rules and stores the value of the model time when the action rules need to be resumed.

The dispatcher prioritizes the Ready Queue, activating the first available object and pausing its own execution until the activated object yields control or terminates.

The dispatcher then activates again and, if the Ready Queue is empty, scans the Condition Queue. If the condition for any of these objects is met, the dispatcher activates that object and pauses until the activated object stops.

The Time Queue is viewed by the dispatcher last. The dispatcher selects the object with the shortest activation time from the queue and sets the model time value according to the activation time of this object. In this way, the model time is changed from event to event. After the time is changed, the dispatcher activates the selected object.

Another important class of the framework is the QueueForTransactions class, whose objects are queues for storing transactions. References to an object of the Dispatcher class, an object of the Diagram class (to display queue changes over time), and a DiscretHisto statistics accumulator can be passed to objects of this class. In this case, the queue will display changes in its state over time on a chart and accumulate statistics about changes in its size.

A variant of such a queue is the Store class, whose objects accumulate their size as a real number, but can also be associated with a statistics accumulator and a chart.

A simulation model building. The framework offers a certain protocol for creating a simulation model. According to this protocol, the model must have a constructor through which a reference to an object of the Dispatcher class is passed and a reference to an object from which the model can obtain settings for its components. Such an object can be, for example, a graphical user interface that provides the ability to configure model parameters and provide access to them. This is the solution presented in Fig. 3, 4, 5, where the QS parameters panel is used to configure the model parameters. The user initiates the modeling process using the Start button. As a result, an object of the Dispatcher class is created. After that, the model is created either directly by the constructor or by the model factory.

The constructor call should ensure the creation and configuration of the model itself and all its components, that are required at the time of start. The model constructor must also ensure the loading of active objects into the dispatcher's Ready Queue through the appropriate method. After the model is created, the modeling process is initiated by calling the dispatcher's start method.

The process of creating a model is discussed in more detail a little later.

The framework includes various classes of random number generators, classes for accumulating statistical data, processing accumulated data, and testing statistical hypotheses.

The visual components of the framework help create an application graphical interface for conducting model experiments. In particular, these are visual components for selecting and configuring random number generators, elements for input/output of data with simultaneous conversion, and diagrams for displaying graphs and histograms.

The ExperimentManager component provides automation of conducting a series of single-factor experiments with a model at one or many levels, as well as variance and regression analysis of the obtained results.

The TransprocessManager component allows you to conduct experiments on parameter estimation and viewing transient processes in the model queues.

The StatisticsManager component allows you to organize the collection of statistical information about the model's operation and convenient viewing of statistical data.

GitLab provides access to the framework via the link [10].

Practical tasks overview. Random number generation. At this stage, first, methods for obtaining uniformly distributed random numbers and their testing are investigated.

Then, methods for creating random number generators for other distribution laws are considered, in particular, uniform (Uniform class), normal (Norm class), triangular (Triangular class), exponential (Negexp class), Erlang (Erlang class), discrete (Discret class) and arbitrary (Linear class). The concept of a “random flow of events” is also investigated.

As practical work, researchers create an application using the tools of the Simulation framework, Fig. 1. This application allows studying the influence of distribution law parameters on the probability density function and the integral distribution function.

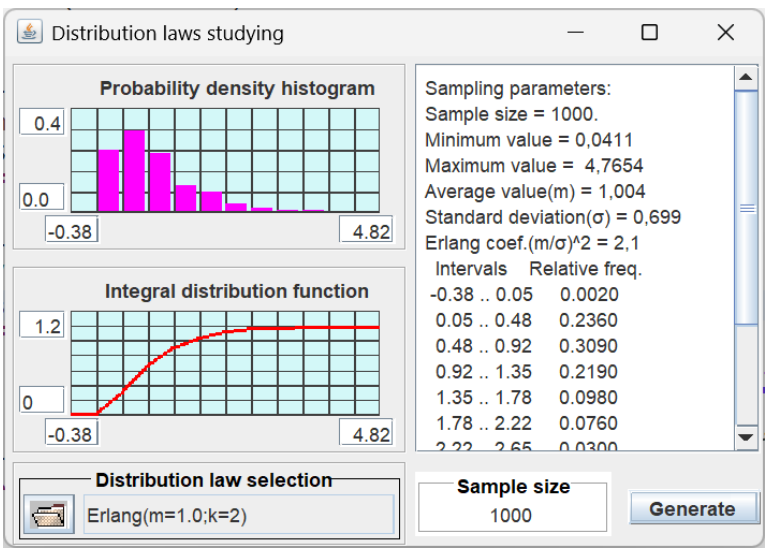

Fig. 1. Application for studying distribution laws
Source: Developed by the authors

Statistical data processing and statistical hypothesis testing. In the bounds of performing this task, statistical processing of the provided samples of numerical data, which are located in text files, is carried out. To implement the task, an application is used that allows you to view the data sample, calculate the main statistical characteristics and test statistical hypotheses about the compliance of the data with the selected distribution law. Fig. 2 shows the results of testing the random number sample and testing the statistical hypothesis for compliance with the selected law.

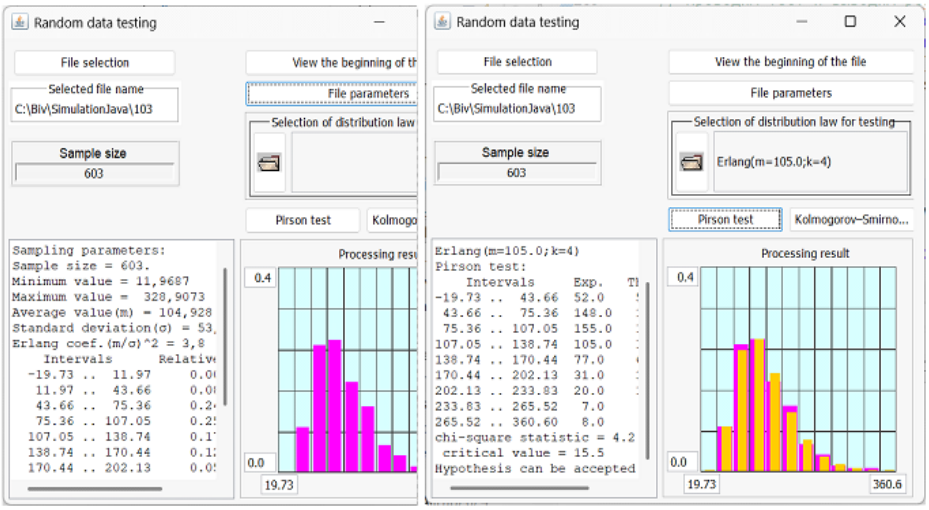

Fig. 2. Processing a sample of random numbers
(a – parameters calculation, b – statistical hypothesis testing)
Source: Developed by the authors

Development of a queueing system simulation model and its research. At this stage, individual tasks are performed to develop and study simulation models for queueing system related to various subject areas. As an example, an object-oriented analysis of soil extraction and transportation works is proposed. The queueing system proposed for modeling includes a bulldozer, a loader, a team of dump trucks and a pile of soil formed by the bulldozer. In the process of completing an individual task, the following steps must be implemented:

– conduct an object-oriented analysis of the system and form a list of objects that will be included in the model;

– determine the list of indicators that characterize the operation of the system, and methods for experimentally determining these characteristics in the modeling process;

– create a simulation model of the system using the tools of the Simulation framework, and program the behavior of active system objects;

– develop a visual application that will allow you to configure model parameters and view the results of statistical processing of modeling results. To simplify this task, the StatisticsManager component is used.

As an example of a software implementation, let's consider an application that models a simple Markov QS. Fig. 3 shows the results of this application.

The active components of the model are represented by the Generator and Device classes, which inherit the Actor class and implement their rule() methods, which are

similar to the rule() method of the FinishDevice class, which was previously considered. The transaction queue is represented by an object of the QueueForTransactions class. The transactions themselves were modeled by objects of the Double class, which contained the time when the transaction was included in the queue. As a storage of statistical data about the queue length, an object of the DiscretHiso class was connected to the QueueForTransactions class object. To accumulate information about the transactions waiting time in the queue, an object of the Hiso class was connected to the Device class object. Another such object was used to accumulate the waiting time of the Device class object.

Lazy instantiation and initialization methods were used to create model objects. When the object was first accessed, the model created this object and initialized it using a reference to the visual part.

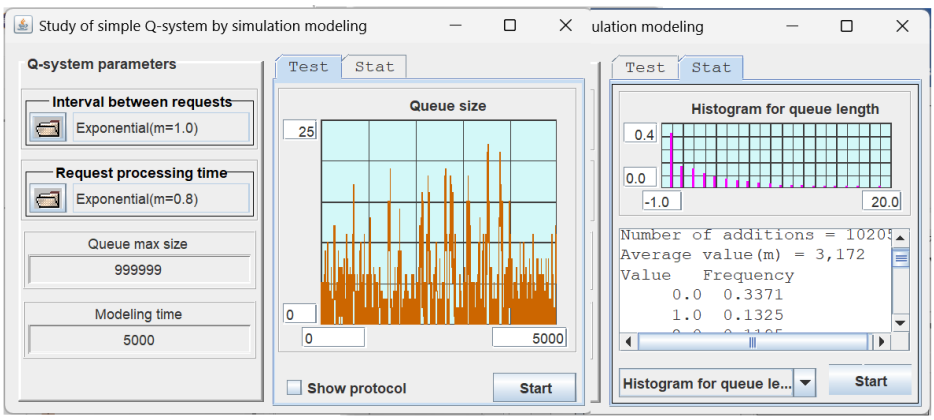

Fig. 3. Application for studying a simple Markov queueing system
Source: Developed by the authors

Automation of single-factor experiments with simulation models. The next stage of practical work is planning and conducting a series of single-factor experiments with the model at one or many levels in order to study the influence of this factor on the results of the system. To implement this task, the ExperimentManager component is provided, to which the own model is connected, by implementing a specific interface for communication between the component and the model.

As an example, an application is considered in which the ExperimentManager works with the model, which was developed at the previous stage. Fig. 4 shows the appearance of this application with an open panel for conducting experiments, which is configured to study the influence of the system load factor on the performance of the system.

The ExperimentManager component not only automates the conduct of a series of experiments with the model, but also provides the ability to conduct variance and regression analysis of the experiments results, as evidenced by Fig. 4.

Study of transient processes in queueing systems queues. At the final stage of practical work, transient processes in the SMO are investigated. For this, the TransprocessManager component is used. The average queue length is used as an indicator that characterizes the state of the system in time. In order to identify patterns that are characteristic of the transient process, the component uses a factory

to create many instances of the model that operate in a single time space. During the operation of these models, the component averages the queue length in time at accumulation intervals and across all implementations. Fig. 5 shows the results of using the TransprocessManager component to analyze the transient process in the queue of a simple queuing system.

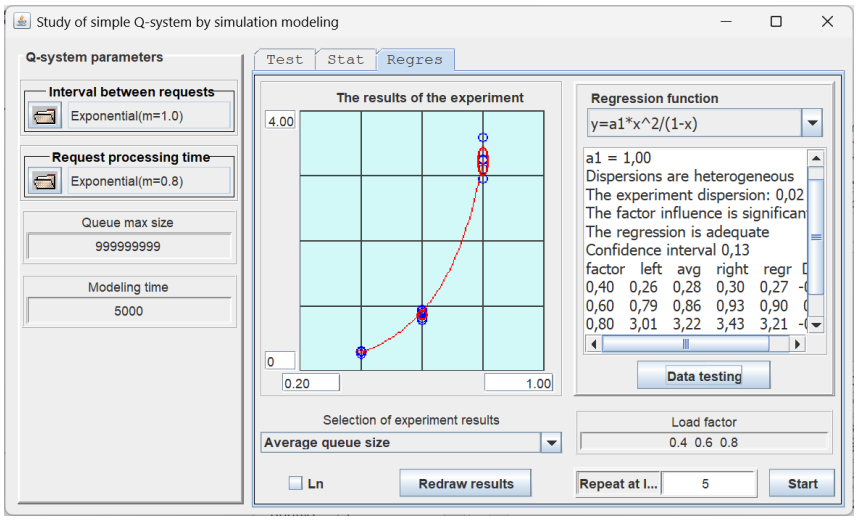

Fig. 4. Results of a series of experiments with a simple QS model and regression analysis of these results

Source: Developed by the authors

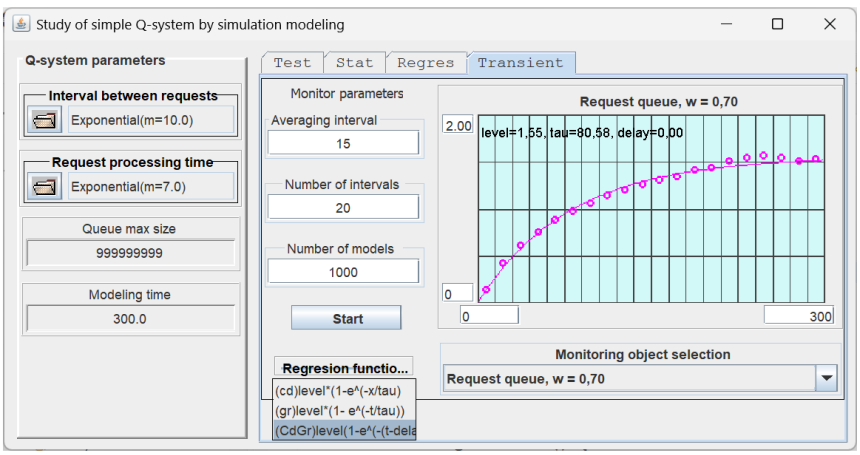

Fig. 5. Results of the transition process study in a simple queueing system

Source: Developed by the authors

Methodological support. While solving practical problems, researchers can use methodological guidelines for developing a simulation model [11] and a study guide on developing simulation models in the Java programming language [12]. Development examples are provided in a software package containing an open-source implementation of the cases discussed. The software package is provided as a .zip file that can be imported into the Eclipse IDE for code examination. Access to the application is provided via GitLab at the link [13].

Conclusion. The proposed organization of the "Systems Modeling" course successfully integrates simulation modeling with advanced OOP principles. The practical implementation of this approach is centered around the Simulation framework, which was developed to bridge the gap between theoretical models and software engineering.

The effectiveness of the framework is demonstrated through its application in students' training projects. Key outcomes of this integration include:

– Deepened OOP Understanding: Students gain practical mastery of core OOP concepts. The framework explicitly showcases polymorphism through class inheritance (e.g., in random number generators) and interfaces (e.g., connecting universal experiment components). The Actor class serves as a practical example of implementing polymorphism via abstract classes.

– Practical Mastery of Design Patterns: Students actively use patterns such as Observer and the Factory design pattern for the dynamic creation and management of models during experiments. – Versatility and Scalability: The framework has proven its versatility across 35 diverse assignment types in fields such as logistics, computer networks, banking, retail, and military science. These models effectively manage between 1 and 5 interacting queues.

– Advanced Features & Collaborative Development: The approach extends to complex topics like real parallelism using thread pools and synchronization mechanisms for multi-level experiments. The open-source nature on GitLab [див. список літератури] fosters student contribution to the modernization of the core code.

We invite readers and the academic community to cooperate in the further improvement of this framework.

REFERENCES

1. Law, A. M. (2014). *Simulation modeling and analysis*. McGraw-Hill Higher Education.
2. Gross, D., Harris, C. M., Shortle, J. F., & Thompson, J. M. (2018). *Fundamentals of queueing theory* (5th ed.). Wiley Series in Probability and Statistics. John Wiley & Sons.
3. *Basic queueing theory*. (n.d.). yzr95924.github.io.
4. *User's guide Mathcad® 15.0 M010*. (2011). lmal.zut.edu.pl.
5. Dabney, J. B., & Harman, T. L. (2004). *Mastering Simulink*. Prentice Hall.
6. Schriber, T. J. (1991). *An introduction to simulation using GPSS/H* (2nd ed.). Wiley.
7. Simio. (n.d.). *Digital twin simulation software*. www.simio.com.
8. Dahl, O. J., & Nygaard, K. (1967). *SIMULA - A language for programming and description of discrete event systems: Introduction and user's manual*. NCC.
9. Pooley, R. (2018). *An introduction to programming in Simula*. portablesimula.github.io
10. *SimulationFramework*. (Version 23) [Software]. (2023). GitLab. gitlab.com.
11. *System modeling: Methodological guidelines for performing calculation and graphic work for higher education students in the educational program "Computer Engineering"*. (2024). [Modeliuvannia system: Metodychni vkazivky do vykonannia rozrakhunkovo-hrafichnoi roboty dlia zdobuvachiv vyshchoi osvity osvitnoi prohramy "Kompiuterna inzheneriia"]. Chernihiv National University of Technology. ir.stu.cn.ua.
12. Byvoino, P. H., Byvoino, T. P., & Pavlovskyi, V. I. (2025). *Development of simulation models in the Java programming language: A study guide for higher education students in "Computer Engineering"* [Rozrobka imitatsiinykh modelei na movi prohramuvannia Java: Navch. posib. dlia zdobuvachiv vyshch. osvity spets. 123 "Kompiuterna inzheneriia"]. Chernihiv National University "Chernihiv Polytechnic". ir.stu.cn.ua.
13. *SimulationAllLab.* [Software]. (2024). GitLab. gitlab.com.

The article was received 09.01.26, received after revision 02.02.26, accepted 27.02.26

Т. Бивойно, Д. Лисенко, П. Бивойно, Н. Сокоринська

ПОЄДНАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ ТА МОДЕЛЮВАННЯ В ОСВІТІ

Анотація. У статті розглянуто інтеграцію імітаційного моделювання та об’єктно-орієнтованого програмування (ООП) в IT-освіті. Автори стверджують, що комерційні інструменти із закритим вихідним кодом (Simulink, GPSS, Simio) обмежують освітній процес, приховуючи внутрішню архітектуру системи. Як рішення представлено Java-фреймворк «Simulation» з відкритим вихідним кодом, розроблений у Національному університеті «Чернігівська політехніка».

Фреймворк базується на дискретно-подієвому моделюванні та концепції «активного об’єкта». Основним елементом є абстрактний клас Actor, який визначає поведінку об’єкта через метод rule(). Управління модельним часом та синхронізація процесів здійснюються класом Dispatcher, який використовує систему черг для обробки подій та перевірки логічних умов за допомогою лямбда-функцій.

Навчальна програма розділена на етапи: вивчення генераторів випадкових чисел (рівномірний, нормальний, Ерланга), статистична обробка даних та побудова моделей систем масового обслуговування (СМО). Для аналізу результатів моделювання студенти використовують такі компоненти, як ExperimentManager для автоматизованого факторного аналізу та TransprocessManager для вивчення перехідних процесів. Особлива увага приділяється об’єктно-орієнтованому аналізу реальних систем, наприклад, логістиці видобутку ґрунту.

Дослідження демонструє, що відкритий вихідний код фреймворку «Simulation» дозволяє студентам досліджувати внутрішню реалізацію складного програмного забезпечення, патерни проєктування та механізми псевдопаралелізму. Такий підхід забезпечує безперервність навчання програмуванню та сприяє розвитку професійних IT-компетенцій. Проект розміщено на GitLab.

Ключові слова: імітаційне моделювання, об’єктно-орієнтоване програмування, Java-фреймворк, системи масового обслуговування, дискретно-подієве моделювання, активні об’єкти, IT-освіта.

Стаття надійшла до редакції 09.01.26, надійшла після рецензування 02.02.26, прийнята 27.02.26

Бивойно Тарас
старший викладач кафедри інформаційних технологій та програмної інженерії, Національний університет “Чернігівська політехніка” (м. Чернігів, Україна)
ORCID ID: https://orcid.org/0000-0001-6914-5441 ResearcherID: B-7478-2017
e-mail: tbivoyno@gmail.com

Лисенко Дмитро
доктор технічних наук, професор кафедри інформаційних та комп’ютерних систем, Національний університет «Чернігівська політехніка» (м. Чернігів, Україна)
ORCID ID: https://orcid.org/0000-0001-6870-6120 ***e-mail:*** lysenko.d@stu.cn.ua

Бивойно Павло
кандидат технічних наук, доцент кафедри інформаційних та комп’ютерних систем, Національний університет “Чернігівська політехніка” (м. Чернігів, Україна)
ORCID ID: https://orcid.org/0000-0001-8145-8459 ResearcherID: R-7447-2016
e-mail: p.g.byvoino@gmail.com

Сокоринська Наталія
доктор філософії, заступник начальника управління комунікацій Адміністрації Державної спеціальної служби транспорту, полковник (м. Київ, Україна)
ORCID ID: https://orcid.org/0000-0001-9713-7289 ***e-mail:*** sokor-nata@ukr.net