Formalization and Algebraic Modeling of University Economics

Volodymyr Peschanenko [0000-0003-1013-9877], Maksym Poltoratskyi [0000-0001-9861-4438], Karina Pryimak [0000-0001-7922-9534]

Kherson State University, Universytets'ka St. 27, 73000, Kherson, Ukraine
{vpeschanenko, mpoltorackiy, prijmak.karina}@gmail.com

Abstract. The article discusses the approach to modeling economic processes at the university using the methods of algebraic modeling and insertional modeling. The formal model of the eco-nomic processes of the university is presented in the article.

Keywords: insertion modeling, formal methods, economic modeling, formalization of the eco-nomic model.

1 Introduction

Modeling is one of the main methods of knowledge, is a form of reflection of reality and is to ascertain or reproduce certain properties of real objects, objects and phenomena using other objects, processes, phenomena, or using an abstract description in the form of an image, plan, map, a set of equations, algorithms and programs.

The possibilities of modeling, that is, the transfer of the results obtained during the construction and study of the model to the original, are based on the fact that the model in a certain sense reflects (reproduces, models, describes, simulates) some of the object features of interest to the researcher. Modeling as a form of reflection of reality is widespread, and a fairly complete classification of possible types of modeling is extremely difficult, if only because of the ambiguity of the concept of "model", widely used not only in science and technology, but also in art and in everyday life.

2 Overview

Consider the list of software products for modeling of economic-mathematical and mathematical models.

The Maple program is still one of the leaders among universal systems of symbolic calculations. It provides the user with a convenient intellectual environment for mathematical research at any level and is particularly popular in the scientific community. Maple's symbol analyzer is the strongest part of this software [1-2]. Maple provides a convenient environment for computer experiments, during which different approaches to the problem are tried, particular solutions are analyzed, and if programming is necessary, fragments requiring special speed are selected. The package allows you to create integrated environments with other systems and high-level universal programming languages. In this package Maple is not like a traditional programming environment, where a rigid formalization of all the variables and actions with them. Here, the choice of suitable types of variables is automatically ensured and the correctness of operations is checked, so that in the general case, the description of variables and the strict formalization of the record are not required. The Maple package consists of a kernel (procedures written in C and well optimized), a library written in Maple, and a developed external interface. The kernel performs most of the basic operations, and the library contains many commands — procedures performed in the interpretation mode. The Maple interface is based on the concept of a worksheet or a document containing I / O lines and text, as well as graphics [2].

MATLAB is one of the most powerful data processing packages today. The name stands for Matrix Laboratory. MatLab system refers to the average level of products intended for symbolic mathematics. MatLab is one of the oldest, thoroughly developed and time-tested systems for automation of mathematical calculations, built on an expanded view and application of matrix operations [3]. However, the syntax of the system programming language is thought out so carefully that this orientation is almost not felt by those users who are not directly interested in matrix calculations. Despite the fact that MatLab was originally intended solely for computing, in the process of evolution, in addition to computing tools, under the license for MatLab, Waterloo Maple acquired the core of symbolic transformations, as well as libraries that provide functions unique to MatLab for mathematical packages [4]. For example, the well-known library Simulink, realizing the principle of visual programming, allows you to build a logic diagram of a complex control system from standard blocks alone, without writing a single line of code. The MatLab system also has ample opportunities for programming. Its C Math library (MatLab compiler) is object and contains over 300 data processing procedures in C. Inside the package, you can use both the procedures of the MatLab itself and the standard procedures of the C language, which makes this tool a powerful tool for developing applications (using the C Math compiler, you can embed any MatLab procedures into ready-made applications).

The Powersim package is an excellent tool for creating continuous models. However, from the point of view of discrete modeling, it is not effective enough. Powersim is suitable for users who need to build continuous models and who want to learn a rather complex Systems Dynamics notation. The Powersim package stands out among the other packages with the ability to process arrays and support teamwork, as well as the fact that it contains a library with a large number of functions [5]. Arrays are also convenient for creating models, in the construction of which the levels change their state, and the developer wants to follow these changes. Powersim includes more than 150 functions, divided into 16 groups, including financial, mathematical, statistical, graphic and historical. Like other packages, Powersim uses animation tools when running models. Key parameters, charts and tables can be displayed directly on the simulation screen, thereby simplifying the viewing of results. The Multiuser Game feature allows multiple users to simultaneously run a model to work together on it. Powersim contains many standard Windows tools, such as menus and toolbars, and supports Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE) technologies [6]. For example, using OLE, a developer can embed a Powersim model into a document created by a word processor, so that changes to the model are automatically reflected in the document. The simulation language Powersim can be used to build models of both simple and complex systems. Nevertheless, Powersim is quite a powerful tool that allows you not only to quickly and visually build and analyze system-dynamic models, but also to demonstrate in an accessible form the simulation results to a wide range of people who are not necessarily experts in mathematical modeling. Powersim belongs to the family of imitation modeling languages (Dynamo, Stella / iThink, Vensim, Rusim), which rather quickly and efficiently allows you to master the technique of simulation modeling to representatives of not only the natural sciences, but also the humanities.

The iThink software product was developed specifically for modeling system dynamics, the company ISee systems, inc. The program allows users to run models created as graphical representations of the system using four fundamental building blocks [7]. Ithink is one of the most powerful products we are considering. From the point of view of continuous modeling, it lags behind Powersim, but it is better to support discrete modeling. In addition, the Ithink package is equipped with excellent tutorials and documentation, as well as a large number of blocks for building a model. The package is available in two versions - Basic and Authoring [8]. The version of Authoring, which was compared with other packages, allows the developer to include in the ruler model with engines and other model management tools, as well as enter diagrams and other images directly into the model, so that users can control the modeling process and immediately see its results. Like Powersim, Ithink uses the Systems Dynamics notation, which is mainly focused on continuous modeling. To implement this system, four types of structures are used: stations, streams, converters, and connectors corresponding to the connections. The Ithink package offers the developer a list of 14 valid variables for defining mathematical relationships. Ithink provides a sensitivity analysis of the model by launching it repeatedly with various input parameters. The results of each run are displayed in a separate line of the output diagram. Baseline data are the main types of distributions used for statistical analysis or a chart. When a model is executed, Ithink uses animation tools that move stations located at different levels in accordance with the model logic. Although the choice of formats for outputting results in Ithink is not as wide as in Extend, from this point of view it is superior to both Powersim and Process Charter.

Arena, developed by Systems Modeling Corporation software for simulation, allows you to create mobile computer models.The basis of Arena technologies is SIMAN modeling language and Cinema Animation system. SIMAN, first implemented in 1982, is an extremely flexible and expressive modeling language [9]. He is constantly improving by adding new features. To display the simulation results used animation system Cinema animation. Arena is equipped with a convenient object-oriented interface and has amazing possibilities for adaptation to various subject areas. In general, the system is extremely easy to use. The Arena simulation model includes the following main elements: Create and Dispose, Process and Queue [10]. Create are elements from which information or objects come into the model. The rate at which data or objects are received from a source is usually given by a statistical function. A drain is a device for receiving information or objects. The concept of a queue is close to the concept of a data warehouse - this is the place where objects are waiting to be processed. The processing time of objects (performance) in different processes may be different. As a result, some processes may accumulate objects waiting for their turn. Often, the purpose of simulation is to minimize the number of objects in the queues. The type of queue in the simulation model can be specified. A queue can be similar to a stack — the objects that arrived last in the queue are sent first for further processing (LIFO: last-in - first-out). An alternative to the stack can be sequential processing, when the objects that arrive first (FIFO: first-in - first-out) are sent first for further processing. More complex queue processing algorithms can be specified. Processes are an analogue of work in a functional model. In the simulation model, the performance of processes can be specified. Arena is a simulation system that allows you to create mobile (simulation) computer models, using which you can adequately describe and predict real processes.

Extend + BPR package. The Extend package as a universal modeling tool is convenient for reorganizing various business processes. To create models in a package, a block development environment is used, which is much easier to use than the Systems Dynamics notation for Powersim and Ithink packages. The Extend package, which has the means of building continuous and discrete models, a wide range of preformed blocks, support from third-party suppliers and the possibility of expansion, is a powerful product from the ones we are considering. Initially, it was focused on the convenient user interface of Macintosh computers [11], then transferred to the Windows environment using the Win32 application programming interface, and now even performs the installation of Win32 on systems that do not have Win32. The package is available in four versions: Basic, Extend + BPR (Business Process Reengineering), Extend + Manufacturing and Extend + BPR + Manufacturing [12]. Additional BPR and Manufacturing facilities include a number of features for vertical markets. In addition, there are many third-party products that support Extend and are targeted to specific applications. The basic package of Basic includes more than 90 preformed blocks combined into libraries, of which Discrete-Event, Generic and Plotter are most often used. The Discrete-Event library includes various actions, queues, gateways, and timers. The Generic library contains random number and source data generators, files for input and output information, and blocks for mathematical, boolean, and financial data. The Plotter library consists of blocks for creating output diagrams and tables. The remaining libraries have a special purpose, for example, they collect statistical information. BPR and Manufacturing packages are provided with additional libraries. In addition, Extend has a built-in language Modl, which allows the developer to build specialized blocks. Selecting a block from the Discrete-Event library automatically builds a discrete model; otherwise, a continuous model. The Extend package is equipped with authoring models creation functions, with the help of which the developer includes text, geometric images and control blocks in the model window so that users can independently modify the model. To control the process of modeling and displaying the results on the display, a tool based on the notepad principle is used. The Extend package provides detailed user guidance, a tutorial, and model examples from a wide variety of areas of activity that can serve as the basis for creating new models, which undoubtedly facilitates the modeling process.

We propose an algebraic approach to economic modeling that is implemented in the scope of the insertion modeling system (IMS) [13]. Insertion modeling focuses on building models and studying the interaction of agents and environments in complex distributed multi-agent systems [13,14].

3 Insertion Modeling System

Insertion modeling is concerned with the construction of models and the study of the interaction of agents and environments in complex distributed multiagent systems.

Environments is an agent that has a dip function. More precisely, the environment is the set of < E, C, A, Ins > , where E - is the set of environment states (identified with behaviors), C - is the set of actions of the medium, A - is the set of actions of agents immersed in the environment, Ins: $E \times F(A) \rightarrow E$ - immersion function. Here F (A) - is the complete algebra of the behavior of agents with the set of actions A. Thus, every medium E admits the immersion of any agent with the set of actions A.

The behavior of agents is described by the algebra of behavior. The algebra of behavior has several basic operations: the prefixing a.u and the non-deterministic selection u + v, where a is the action, u, and v-behavior. Parallel and sequential composition (u || v, u; v).

It should also be noted that there are two terminal constants in algebra of behavior: successful completion and dead-end behavior 0.

Basic protocols are used to represent the insertion models. The general theory of basic protocols is presented in [14]. The methods of verification of requirements and specifications of distributed systems in the field of telecommunications, embedded systems and real-time systems have been developed with the help of the language of basic protocols.

$\forall x\ (Ux \rightarrow \langle P \rangle\ Vx)$, where U (x) is the precondition that defines the state when the protocol can be used; V (x) is the postcondition that defines the transition to a new state; P is the process that illustrates this transition. The basic idea of the theory of basic protocols is presented in [15].

Further, let's consider the formalization of the University's economy using the methods of algebraic programming and insertion simulation.

4 Formalization of economic model of university

The process of formalizing the economy of a university consists of several stages: the selection and description of agents that are involved in the model, the definition of their attributes corresponding to the required level of abstraction, the definition of agent actions and the design of agent behavior.

Based on the model requirements, we selected the following types of agents: country, university, teachers (professor, associate professor, teacher, assistant), state employees, contract service students.

The list of agents and their interaction with each other can be represented by the following diagram.

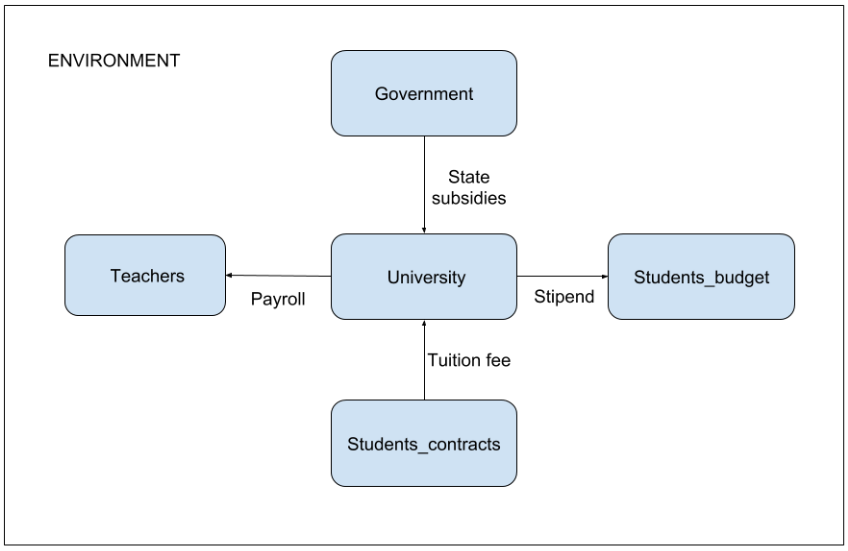

Fig. 1. Diagram of agents and their interaction with each other

The main objectives of the formalization of the university’s economic processes are:

1. The search for modeling errors, such contradictions, deadlocks
2. Search for effective system scenarios in the model, etc.
3. Opportunities to analyze and predict the model;

The initial stage of modeling is the definition of both agents and their attributes. Agents in among inertial modeling can be represented as follows, consider an example of the description of the agents of the university and the state:

```
GOVERNMENT:obj(
score:real
…….
),

UNIVERSITY:obj(
score:(Score_TYPE)->real
…….
)
```

It should also be noted that the score attribute of the university agent has an enumerated type, this is based on the requirements, since the university has two accounts: special and basic, this fact among inertial modeling can be represented as follows:

```
Score_TYPE:(SPECIAL,GENERAL)
```

The interaction between agents is carried out through the language of the basic protocols [15]. The following are examples of the basic protocol formalizations and their description in natural language:

Table 1. The examples of the basic protocol formalizations

Basic protocol	Description
`calcTuitionFee` `= Operator(Forall(i:int,j:int)( (i>=1&i<=2&j>=1&j<=max&(currentMonth=1))-> <"calculation ratio of tution fee by month"> (sumTuitionFee:= sumTuitionFee+(countStudents *tuitionFee(i,j)*rationStd(i,j) )))`	Calculation of the monthly amount of tuition fees for all students of contract. It is important to note that: tuitionFee (i, j) is the cost of tuition for one month according to the distribution "department / course" where, i - department; j - course. tuitionFee(1,1)=1200 tuitionFee(1,2) =1300 …. tuitionFee(2,2) = 1700
`tutionFee` `= Operator( (currentMonth>=1 && (~(currentMonth=11) &&(~(currentMonth=12)) )-> <"Tuition fee"> (univ.score(GENERAL):=univ.score(GRNERAL) +sumTuitionFee*coef; univ.score(SPECIAL):=univ.score (SPECIAL) + sumTuitionFee * coef))`	Tuition fees, the amount that is calculated in the protocol is transferred to the university account. calcTuitionFee. 1. 50% - transferred to a special fund. 2. 50% is transferred to the general fund.
`sumSalary=``Operator(Forall (i:Teachers_TYPE)( (i>=1&i<=4 & (currentMonth>1))-> <"calculation ratio of salary by month">`	The calculation of the monthly amount that the university spends on salaries for faculty members, where: Salary(Professor) = n1 Salary(Docent) = n2 ….. Salary(Assistant)=n3

Basic protocol	Description
`(sumSalary:=sumSalary + countTeachers * Salary(i) )`	It should be noted that Salary (Professor) = n1 - the value that the university spends to pay salaries for all professors of the university.
`salary = Operator( (currentMonth>=1)-> <”payment of wages”> (univ.score(GENERAL):=univ.score(GENERAL) - coef * sumSalary; univ.score(SPECIAL):=univ.score (SPECIAL) - coef * sumSalary; coach.score:=coach.score + sumSalary) )`	Payment of wages.

At the highest level of agent behavior can be represented as follows, and fig. 2. Presents a graphical version of the behavior algebra. Consider each of the subprocesses in more detail:

1. **tutionFee** - the process of paying for tuition; contract-based students pay university tuition fees.
2. **slaryForTeachers** - payroll process.
3. **stipend** - scholarship payment.
4. **otherExpenses** - a process representing unplanned expenses of a university (business trips, organization of conferences, and so on).

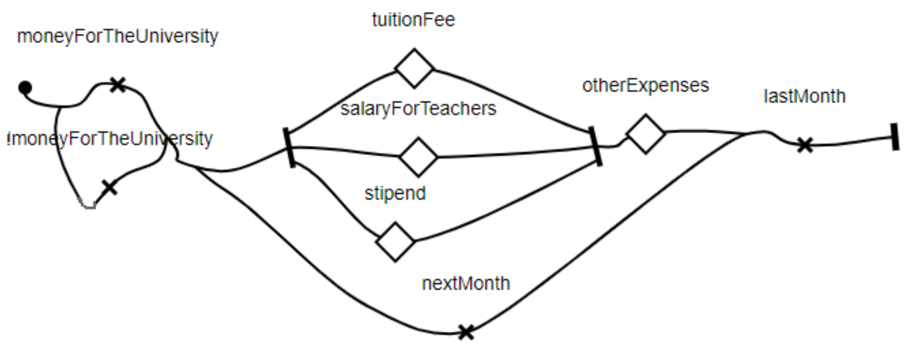

Fig. 2. Graphic representation of behavior algebra

It should be noted that the tutionFee, slaryForTeachers, otherExpenses processes are parallel processes. The diagram presented in fig. 2, can be represented in algebraic form as follows:

```
B0 =moneyForTheUniversity . EB1 + !moneyForTheUniversity
. EB1,
EB1 =(
{tutionFee || slaryForTeachers || stipend}; otherExpens-
es; EB2
),
EB2 = nextMonth . B0 + lastMonth . Delta
```

Initial values for model initialization are given in a special logical formula. Building a model with specific values includes defining specific values for agents and attributes of the environment.

In contrast to a specific model, the symbolic one allows us to analyze the formal model for stability and stability. To test stability, the first step is to select the indicator of stability of the economic model. Since, the tuition fee is directly related to the number of applicants, and often changes, we have chosen deltaPriceEducation as a parameter of stability. This parameter is very important because a very small average tuition fee will negatively affect the economy of the university, and a very high one will affect the number of applicants. In this project, this fact can be described as follows:

```
2000>=deltaPriceEducation>=1000
```

In the process of modeling, we can get the following formulas; deltaPriceEducation = P (p1, p2, ... pn) - where, p1, p2, ... pn are unknown parameters for the model. Proof that these parameters fall within specified intervals, can confirm or refute the properties of the model.

5 Conclusions

Using the proposed approach gives us the opportunity to study the reliability and stability of economic models to check the safety and properties of models. In the process of formalizing such models, there may be some problems associated with the interdisciplinary level. The article presents a method for analyzing and modeling economic models, provided a formal model of the economy of the university. This approach makes it possible to evaluate the model, and to take into account various scenarios of behavior. Using the methods of algebraic programming, we can analyze models on inconsistencies and seek non-determinism and deadlocks.

References

1. Gander, W., Gander, M. J., Kwok, F.: Scientific computing: an introduction using Maple and MATLAB. Springer, Heidelberg (2014).
2. Biran, A., Breiner, M.: MATLAB for Engineers. Pearson Education, Harlow, Essex (2002).
3. Herniter, M. E.: Programming in MATLAB. Brooks/Cole Publishing Co, Florence, Kentucky, U.S.A (2000).
4. Malczynski, L. A..: Best practices for system dynamics model design and construction with Powersim studio. Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California (2013).
5. Giammarini, M., Orcioni, S., Conti, M.: Powersim: power estimation with SystemC. In: Solutions on Embedded Systems, pp. 285-300. Springer, Dordrecht (2011).
6. Sumari, S., Ibrahim, R., Zakaria, N. H., Ab Hamid, A. H.: Comparing three simulation model using taxonomy: System dynamic simulation, discrete event simulation and agent based simulation. International Journal of Management Excellence 1(3), 54-59 (2013).
7. Murphy, B.P., Randolph, C.M.: Simulation packages today: new user classes and order-of-magnitude breakthroughs in development costs. Computational statistics & data analysis 16(4), 471-479 (1993).
8. Hammann, J. E., Markovitch, N. A.: Introduction to Arena [simulation software]. In: Proceedings of the Winter Simulation Conference, pp. 519-523. IEEE (1995).
9. Batarseh, O., McGinnis, L. F.: System modeling in SysML and system analysis in arena. In: Proceedings of the Winter Simulation Conference, p. 258. IEEE (2012).
10. Krahl, D.: An introduction to Extend [hierarchical simulation modelling tool]. In: Proceedings of the Winter Simulation Conference, pp. 538-545. IEEE (1994).
11. Krahl, D.: Extend: the Extend simulation environment. In: Proceedings of the 34th conference on Winter simulation: exploring new frontiers, pp. 205-213. IEEE (2002).
12. Letichevsky, A. A., Letychevskyi, O. A., Peschanenko, V. S.: Insertion modeling system. In: International Andrei Ershov Memorial Conference on Perspectives of System Informatics, pp. 262-273. Springer, Berlin, Heidelberg (2011).
13. Letichevsky, A., Gilbert, D.: A model for interaction of agents and environments. In: International Workshop on Algebraic Development Techniques, pp. 311-328. Springer, Berlin, Heidelberg (1999).
14. Letichevsky, A., Kapitonova, J., Letichevsky Jr, A., Volkov, V., Baranov, S., Weigert, T.: Basic protocols, message sequence charts, and the verification of requirements specifications. Computer Networks 49(5), 661-675 (2005).