На рис. 6 продемонстровано візуальну інтерпретацію алгоритму для внесення нових блоків в ланцюг та перевірки цілісності бази даних.

Рис. 6. Користувацький інтерфейс для переведення коштів на основі технології блокчейн

Висновки. В роботі було проведено детальний розбір технології блокчейн, за допомогою якої можна досягти децентралізації збереження даних та посилити захисний бар'єр до їх змінення, але в той же час за потреби, зробити процеси прозорими. Структура блокчейн дозволяє зв'язати воєдино всі коли-небудь виконані транзакції. Структура копіюється на всі вузли (комп'ютери) системи, що дозволяє кожному учаснику мати достовірну інформацію про всі транзакції без потреби отримувати її з централізованого джерела. Проведене дослідження дозволяє зробити висновок про те, що технологія блокчейн є досить захищеним та інноваційним засобом збереження інформації.

Список використаних джерел

1. Сайт «The great chain of being sure about things». – [Електронний ресурс]. – Режим доступу: https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable.

2. Децентрализованные приложения. Технология Blockchain в действии. – СПб.: Питер, 2017. — 240 с.: ил. – (Серия «Бестселлеры O'Reilly»). ISBN 978-5-496-02988-9.

METHOD OF TRANSFORMING ABSTRACT MODELS OF SOFTWARE SYSTEMS INTO SIMULATION MODELS

Dorenskyi O., Drieiev O.
Central Ukrainian National Technical University

Abstract. As a result of the dynamic development of information technology and its active implementation in all spheres of human activity it is necessary to create and apply software systems. It is expedient and necessary to simulate them at the early stages of their life cycle. Therefore, the problem of achieving simulation models of software systems on the basis of abstract models is solved in the article. The achievement of simulation models of software systems on the basis of abstract models is solved by transforming state models into equivalent automaton networks. The networks allow simulating asynchrony and non-determinant of parallel independent events, conflicting interactions between processes, describing typical situations in the systems as well as general dynamics of functioning of complex asynchronous systems.

Keywords: software, state model, state-machine net, model of software system.

МЕТОД ПЕРЕТВОРЕННЯ АБСТРАКТНИХ МОДЕЛЕЙ ПРОГРАМНИХ СИСТЕМ В ІМІТАЦІЙНІ

Доренський О.П., Дрєєв О.М.
Центральноукраїнський національний технічний університет

Анотація. Наслідком динамічного розвитку інформаційних технологій та їх активного впровадження у всі галузі людської діяльності є необхідність створення й використання програмних систем. Є доцільним і необхідним їх моделювання на ранніх стадіях життєвого циклу. Тож у роботі розв'язується задача побудови імітаційних моделей програмних систем на основі абстрактних моделей шляхом перетворення моделей станів у еквівалентні їм автоматні мережі, які дозволяють моделювати асинхронність та недетермінованість паралельних незалежних подій, конфліктні взаємодії між процесами, описувати як типові ситуації в системах, так і загальну динаміку функціонування складних асинхронних систем.

Ключові слова: програмне забезпечення, модель станів, мережа, модель програмної системи.

Fast development and implementation of information technology in almost all spheres of human activity has resulted in a dynamic process of developing the corresponding software. It is the main form of IT support. At the same time, software is characterized by complexity, multi-component nature, compliance with a number of severe and standardized requirements for reliability, functional completeness, reactivity, adaptability, as well as the possibility for improvement, scaling, etc. Thus, the software development process requires application of the approach which will meet the established requirements [1]. Therefore, the task of simulation modelling of software systems is very often set [2, 3]. It is carried out at the initial stages of the software lifecycle (in the process of analyzing the requirements for the software, architectural design, or detailed design), at which the software system is presented as abstract (design) models. Consequently, it is necessary to develop the method of achieving simulation models based on abstract (design) models of software systems.

The method of transforming abstract models of software systems into simulation models consists of two stages: 1) representation of the state model of the software system by an initial finite automaton; 2) conversion of the finite automaton to the automatic network which allows realizing the process of simulating the functioning of the software system.

The life cycle of the software system described the state model transforms into the finite automaton in the following way. To each element of the set of states SS of the state model MS corresponds a single element of the set of states V of the finite Moore automatic machine A; bijective mapping is defined: $F_{VSs}: V \to SS$. To each element of the set of events SE of the state model MS corresponds a single element of a set of input signals X of the finite automaton A; bijective mapping $F_{XSe}: X \to SE$ is defined. To each element of the set of functioning SH of the state model MS corresponds a single element of the set of output signals Y of the A machine; bijective mapping $F_{YSh}: Y \to SH$. To each ordered pair $((se, ss_i), ss_j) \in F_{SesSs}: SE \times \times SS_{sc} \to SS_{sc}$ of the state model MS corresponds a single ordered pair $((\chi, v_i), v_j) \in F_{XV}: X \times V \to V$ of the finite automation A; that is, the condition $F_{SesSs}(F_{XSe}(\chi), F_{VSs}(v)) = F_{VSs}(F_{XV}(\chi, v))$ is fulfilled. To each ordered pair $(ss, sh) \in F_{SsSh}: SS \to SH$ of a state model MS corresponds a single ordered pair $(v, \upsilon) \in F_{VY}: V \to Y$ of the finite automation A; that is the condition $F_{SsSh}(F_{VSs}(v), F_{YSh}(\upsilon)) = F_{YSh}(F_{VSs}(v))$ is fulfilled. To the initial state of creating ss_0 of the model MS corresponds the state $v_1 \in V$ of the finite automaton A; that is, according to the expression $F_{VSs}(v_0) = ss_0$. To each element of the set of states $SS_{sc}^{sac} \subset SS_{sc}$ of the state model

MS corresponds a single element of the set of states $V' \subseteq V$ of the finite automaton A; the bijective mapping $F_{V'Ss} : V' \to SS^{sac}$ is defined.

The final stage consists of achieving a simulation model. Each finite abstract machine can correspond to an ordinary secure Petri net N_A [2, 4]. The received finite-automation state model $A = \left(SS, SE, SH, F_{SeSs}, F_{SsSh}, ss_0, SS^{sac}\right)$, where SS is a set of states, SE is a set of events, SH is a set of functioning, ss_0 is the initial state, $ss_0 \in SS$, SS_{sc}^{sac} is a set of states, is transformed into a corresponding automaton network of the $N_A = \left(B^A, Z^A, F_{BZ}^A, F_{ZB}^A, M_0^A\right)$ form.

Consequently, the suggested model provides the description and presentation of the dynamic processes of software systems for simulation modelling, analysis and semantic verification of their functioning. This will allow, in particular, achieving the appropriate indicators of the reliability of the software systems being developed, reducing the resources for the creation and implementation of the software. On the basis of N_A it is possible to obtain the necessary cases of using the system: any pathname from the initial position from the set B^A to the end position is the case of the system use.

As a result of the analysis of functioning N_{Asc}, the corresponding access tree of reach is formed. Each of its nodes is classified either as boundary, terminal, duplicating node, or as an inner one. Limits are the nodes that are not yet processed by the algorithm of operation N_A. After processing they become either terminal, or duplicating, or internal.

References

1. Weisfeld M. The Object-Oriented Thought Process, Fourth Edition / Matt Weisfeld. – Addison-Wesley, Pearson Education, Inc., 2013. – 306 p.

2. Доренський О. П. Імітаційна модель програмного забезпечення інформаційно-управляючої системи на логічному рівні / О. П. Доренський // Комп'ютерне моделювання та оптимізація складних систем (КМОСС-2015): І Всеукр. наук.-техн. конф. (м. Дніпропетровськ, 3-5 лис. 2015 р.) : матеріали, в 2-х ч. – Дніпропетровськ: ДВНЗ УДХТУ, 2015. – Ч. 1. – С. 64-65.

3. Дрєєв О. М. Імітаційна модель фракталізації мережного трафіку // О. М. Дрєєв, О. П. Доренський // Сучасні інформаційно-телекомунікаційні технології: Міжнар. наук.-техн. конф., м. Київ, 17-20 лис. 2015 р.: матеріали наук.-техн. конф. – К.: ДУТ, 2015. – Т. ІІІ. – Розвиток інформаційних технологій. – С. 38-39.

4. Питерсон Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 270 с.

ПОШУК ЗВ'ЯЗКІВ І ЗАЛЕЖНОСТЕЙ У ДАНИХ З ВЕБ-СТОРІНОК

Катеринич Л.О., Петелько Ю.Ю.
Київський національний університет імені Тараса Шевченка

Анотація. Стрімкий розвиток мережевих технологій, зокрема мережі Інтернет, спричинив значний ріст кількості інформації, що стала загальнодоступною, і яка може бути отримана у будь-який час з будь-якої точки світу. На сьогоднішній день, існує велика кількість пошукових систем, алгоритмів, а також програмних засобів, які так чи інакше справляються з такими задачами, проте використання, наприклад, пошукових систем, часто вимагає здійснення деякої, деколи великої, кількості однакових, механічних дій, які, насправді, є часовитратними і можуть бути автоматизовані. Оскільки кількість результатів може бути дуже великою, а також релевантність