УДК 004.43::004.94 https://doi.org/10.15407pp2021.04.082

А.Ю. Дифучин, І.В. Стеценко, Е.В. Жаріков

ГРАМАТИКА МОВИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ ПЕТРІ-ОБ'ЄКТНИХ МОДЕЛЕЙ

Петрі-об'єктні моделі вирішують проблему тиражування фрагментів мереж Петрі з заданими параметрами та конструювання моделі системи з великої кількості елементів. Розроблена мова візуального програмування Петрі-об'єктних моделей дає можливість зменшити кількість помилок при конструюванні моделі за рахунок автоматизації кодування зв'язків між елементами моделі та графічного представлення моделі. Окрім тиражування Петрі-об'єктів, мова реалізує тиражування зв'язків між Петрі-об'єктами. Формалізація граматики мови візуального програмування представлена у вигляді правил виведення та зроблені висновки щодо її властивостей.

Ключові слова: формальна граматика, візуальне програмування, стохастична мережа Петрі, імітаційне моделювання.

Вступ

Програмне забезпечення імітаційного моделювання – це широкий клас програмних продуктів, призначених для побудови моделей, які відтворюють функціонування систем. Моделі використовують для дослідження властивостей систем, пошуку оптимальних умов, прогнозування. Моделі, побудовані з використанням мереж Петрі, рекомендовані для проєктування та аналізу програмного забезпечення стандартом ISO/IEC 15909-1:2004 [1]. Перевагами такого використання є візуалізація, математично формалізований опис, покрокове відтворення обчислювального процесу. Застосування мереж Петрі для розробки алгоритмів і програмного забезпечення з часом може втілитись у парадгму програмування на мережах Петрі [2]. Отже, мережі Петрі як засіб побудови моделей надзвичайно важливі для розвитку програмної інженерії. Недолік, з яким доводиться стикатись у розробці моделей мережами Петрі - це ускладнення сприйняття та побудови моделі при зростанні складності системи, що моделюється (складність може бути оцінена кількістю елементарних подій). Через нагромадження зв'язків та елементів перевага візуального сприйняття моделі зникає, модифікація параметрів моделі потребує значної кількості рутинних дій. Тому розвиток методів, що спрощують і прискорюють розробку моделей на основі мереж Петрі є актуальним науковим завданням.

Петрі-об'єктні моделі вирішують проблему тиражування фрагментів мереж Петрі з заданими параметрами та конструювання моделі системи з великої кількості елементів. Якщо система складається із сотень однотипних вузлів, опис кожного з яких потребує 10 переходів, 20 позицій та 40 дуг, то в термінах звичайної мережі Петрі довелось би створювати 1000 переходів, 2000 позицій, 4000 дуг та щонайменше 100 дуг для зв'язування фрагментів між собою. В термінах Петрі-об'єктної моделі один раз розроблена мережа Петрі, що описує функціонування одного вузла системи, використовується для створення 100 вузлів (із різними значеннями параметрів). Отже, розробка моделі значно спрощується і прискорюється. Скорочується також обсяг графічної інформації, яку потрібно зберігати для відтворення моделі. Широко відомий засіб для моделювання мережами Петрі CPNTools дає можливість використовувати фрагменти мережі Петрі повторно, проте налагоджувати і вбудовувати фрагмент у модель доведеться для кожного фрагменту окремо. Звісно, такий спосіб спрощує створення нового фрагменту мережі Петрі на основі вже існуючого, але не спрощує створення множинних наборів типових фрагментів. По суті, уможливлюється копіювання фрагментів і створення на їх основі нових, але не тиражування за заданим шаблоном з заданими параметрами. Зберігатись у CPNTools будуть усі скопійовані фрагменти

як наново створені, отже, на відміну від Петрі-об'єктного підходу, скорочення обсягів пам'яті на збереження фрагментів моделі не відбувається.

Петрі-об'єктні моделі дають можливість створювати моделі засобами мереж Петрі там, де раніше вони призводили до надмірного ускладнення: моделювання розповсюдження комп'ютерних вірусів [3], передача повідомлень у телекомунікаційних системах [4], моделювання алгоритмів паралельних обчислень [5]. Застосування Петрі-об'єктного підходу для моделювання систем різного призначення показало ефективність цієї технології конструювання моделей, що поєднує переваги об'єктно-орієнтованого підходу та стохастичних мереж Петрі. Проте кодування зв'язків між елементами мережі Петрі є процесом, схильним до помилок. Відшукання помилок у структурі моделі безпосередньо в коді потребує концентрації уваги. Якщо модель містить опис двох десятків подій, із цим ще можна впоратись. При більшій кількості подій процес розробки моделі настільки трудомісткий, що результати моделювання можуть не виправдати зусиль, витрачених на створення моделі.

Рішення цієї проблеми можна шукати у двох напрямках: 1) автоматизація, інтелектуалізація чи інші способи запобігання та виявлення помилок безпосередньо в коді, 2) візуалізація зв'язків та моделі в цілому для поліпшення сприйняття моделі, що будується. Підходи, які використовують виключно візуальне представлення моделі, примушують програміста відкривати та модифіковувати модель навіть зі зміною її параметрів. А якщо потрібна зміна водночас в кількох однотипних елементах, то доведеться повторювати рутинну дію. Такі дії швидше виконати безпосередньо в коді (одночасно потрібно забезпечити, щоб при відкритті моделі такі зміни потрапили у візуальне її представлення). Перевірити правильність зв'язків чи стану моделі в початковий момент часу з використанням інтелектуальних підходів не є можливим, оскільки можливі (допустимі) послідовності подій, які відповідають задуму моделі, є надзвичайно великою множиною навіть для відносно простих моделей.

Ідеальним вбачається рішення, коли програміст може застосовувати візуальне представлення і, водночас, використовувати кодоване представлення моделі. Проте перетворення візуального представлення моделі у кодоване відбувається, як правило, з втратою даних про розміщення графічних елементів. Тому перетворення з кодованого представлення моделі у візуальне її представлення потребує інтелектуальної обробки інформації про взаємозв'язки елементів. Ускладнює таке перетворення той факт, що не існує однозначної відповідності між кодованим представленням та візуальним представленням моделі, оскільки одній моделі може відповідати кілька варіантів візуального представлення з різним розміщенням графічних об'єктів.

Конфлікт між графічним представленням і кодованим представленням моделі зникає, якщо припустити, що візуальне представлення є водночас кодованим представленням моделі. Візуальні мови програмування використовують графічні об'єкти як алфавіт мови. Отже структури, складені з таких об'єктів є фактично кодом для представлення моделі.

Метою даного наукового дослідження є зменшення складності процесу конструювання моделі дискретно-подійної системи за рахунок використання мови візуального програмування Петрі-об'єктної моделі.

1. Формальні мови та засоби імітаційного моделювання дискретно-подійних систем

Із появою формальних граматик з'явились машино-незалежні мови програмування. Першою мовою програмування, побудованою на основі формальної граматики, стала мова ALGOL [7]. Мова імітаційного моделювання Simula, побудована на основі мови ALGOL, була задумана розробниками спочатку виключно як мова для цілей імітаційного моделювання дискретно-подійних систем, а стала першою об'єктно-орієнтованою мовою [6]. Її версія Simula 67 була написана вже як універсальна об'єктно-орієнтована мова програмування. Мовою Simula було написано провідне у свій час програмне забезпечення Arena. Ви-

користання блочних діаграм для побудови моделей і надзвичайно зручна розбудова ієрархічних моделей – основні переваги цього програмного забезпечення. Найбільш популярне в наші дні програмне забезпечення з імітаційного моделювання Simio по суті є продовженням концепції Arena з розвинутішими засобами анімації, зокрема 3D, та можливостями завдання розкладів [8].

Вказані програмні засоби та багато інших (AnyLogic, VisualSim, Simul8) ґрунтуються на описі дискретно-подійних систем у термінах блочних діаграм, що описують процес обробки об'єктів. Складні моделі зі специфічною взаємодією елементів важко вкласти у логіку спрацьовування достатньо обмеженого набору блоків, тому кількість блоків у бібліотеках постійно зростає, що негативно впливає на швидкість вивчення програмного забезпечення. Окрім того, логіка блочних діаграм завжди спирається на опис процесу обробки об'єктів, водночас управління ресурсами для обробки обмежено тільки їх кількістю, місткістю та розкладом. Деякі з існуючих програмних продуктів (Simio, Simul8) визнали обмеженість блочної побудови моделей і дають можливість розробникам створювати власні фрагменти коду і вбудовувати їх у моделі. Наприклад, Simio пропонує додавати написані на .NET фрагменти коду для передачі даних, опису специфічних алгоритмів вибору об'єктів з черг, опису серії експериментів [9]. Simula8 дає можливість додаткового опису функціональності блоків мовою Visual Logic[10].

Іншим підходом до побудови симуляції є розробка формалізації у вигляді мережі Петрі і запуск універсального алгоритму імітації для побудованої мережі. Програмне забезпечення CPNTools [11] використовує формалізацію опису моделі у вигляді ієрархічних розфарбованих мереж Петрі та функціональну мову CPN ML для опису даних, змінних та функцій. Мова CPN ML є розширенням мови функціонального програмування Standard ML. Використання функціональної мови програмування дає можливість CPNTools досягти необхідної гнучкості в побудові моделей, проте значно ускладнює сприйняття моделі з графічного представлення, оскільки, окрім графічних об'єктів та їхніх параметрів, опис моделі містить вирази-функції. Якщо CPN ґрунтується на математичному формалізмі розфарбованих мереж Петрі, то при додаванні до опису моделі фрагментів коду функціональною мовою математичний формалізм руйнується [12].

Отже, використання програмних засобів імітаційного моделювання свідчить про переваги використання графічного представлення моделі, такі як наочність представлення, швидкість розробки, зменшення кількості помилок у відтворенні структури. Водночас, забезпечити необхідну гнучкість розробки моделей не вдається без надання розробнику доступу до програмного коду моделі та/або надання дозволу вбудовувати фрагменти програмного коду у код моделі.

2. Поняття мови візуального програмування

Терміни «візуальне моделювання» та «візуальне програмування» суттєво відрізняються. Візуальне моделювання - це застосування графічних об'єктів для створення моделей. У загальному випадку візуальні моделі не призначені для обчислень. Наприклад, моделі UML використовують для опису програмного забезпечення, що проєктується. Візуальне програмування – це застосування графічних об'єктів для розробки програмного коду, який реалізує певний алгоритм обчислень. Відповідно, існує суттєва різниця між мовою візуального моделювання та мовою візуального програмування. Мова візуального моделювання – це набір графічних об'єктів для представлення систем, об'єктів або понять, що проєктуються або існують. Мова візуального програмування – це набір взаємопов'язаних графічних об'єктів (разом з чисельними параметрами), який однозначно може бути трансформований у програмний код.

На відміну від природних мов, мови програмування є формальними, тобто породжуються формальними граматиками. Опис будь-якої формальної граматики складається з алфавіту термінальних символів, алфавіту нетермінальних символів, початкового символу граматики та правил виведення, що використовують для розпіз- навання наборів символів. Алфавіти термінальних та

нетермінальних символів разом утворюють алфавіт мови. Граматика мови програмування породжує множини слів (виведені з термінальних символів за правилами виведення), що можуть бути інтерпретовані (перетворені) в модель обчислень. Приклади розробки формальних граматик, що орієнтовані на використання упорядкованих наборів символів, викладені у [13]. Математична теорія формальних граматик викладена у [14]. Формальне представлення граматики мови важливе для гарантування однозначності інтерпретації мовного виразу та однозначного перетворення його у алгоритм (перелік інструкцій для обчислень), що запускається на виконання.

Окрім алфавіту, неодмінними ознаками формальної мови є синтаксис мови та її семантика. Синтаксис визначає дозволені конструкції мови, семантика - зміст цих конструкцій. Синтаксис визначає набір правил, за якими складаються мовні конструкції. Розбір мовного виразу, складеного за синтаксичними правилами, виконується компілятором мови у відповідності до правил виведення (продукцій), що задані граматикою мови. Семантика – це зв'язок між синтаксисом мови та моделлю обчислення. Формальна семантика задається математичною моделлю, що описує обчислення у відповідності до виразу, заданого мовою. Синтаксичні помилки виявляються автоматизовано під час компіляції програмного коду, а семантичні – під час виконання програми.

Алфавіт мови візуального програмування – це набір графічних елементів, що виконують роль символів. У текстовій мові програмування слово (лексема) – це послідовність символів, утворене за правилами, що визначені синтаксисом мови. Розташування символів у послідовності зліва направо означає фактично їх з'єднання, і це з'єднання інтерпретується граматикою (при розборі виразу). У візуальній мові, що використовує графічне представлення для опису мовного виразу, на розташування елементів у графічному просторі не накладається ніяких обмежень. Тобто послідовності символів у тому розумінні, як вони присутні у традиційних мовах програмування, відсутні. Натомість у графічному представленні встановлюються зв'язки між елементами. Наприклад, якщо кінець дуги співпадає з місцем розташування вершини графу, то елемент дуга поєднаний з елементом вершини. Якщо такі зв'язки можуть утворювати фрагменти, які однозначно інтерпретуються, як лексеми мови, то графічне представлення може використовуватись для візуального представлення виразів програмного коду.

У разі текстової мови програмування текст перетворюється в алгоритм виконання обчислень за правилами, визначеними семантикою мови. У мові візуального програмування зображення, складене з взаємопов'язаних графічних елементів, трансформується у змістові конструкції, що запускаються на обчислення. Опис графічного елементу може супроводжуватись набором чисельних параметрів, що впливають на обчислення. Отже, виразом мови візуального програмування є набір взаємопов'язаних графічних об'єктів, що може бути запущений на обчислення.

Граматика текстової мови породжує послідовності символів термінального алфавіту, виведених з початкового символу. Граматика мови візуального програмування має породжувати графічні зображення, утворені з графічних елементів, що складають алфавіт термінальних символів мови. Далі на прикладі мови візуального програмування Петрі-об'єктних моделей, буде показано, що мова візуального програмування може бути визначена формальною граматикою так само, як і текстова мова програмування, за умови, що визначені зв'язки між графічними елементами.

2. Поняття Петрі-об'єктної моделі

Петрі-об'єктний підхід до розробки імітаційних моделей дискретно-подійних систем комбінує переваги використання стохастичних мереж Петрі та об'єктно-орієнтованої технології для опису моделей. Стохастична мережа Петрі з багатоканальними та конфліктними переходами визначена такими множинами елементів: позиції, переходи, дуги. Для зручності обробки графічного представлення будемо розрізняти вхідні та вихідні дуги переходу. Запуск переходу у мережах Петрі з часовими затримками здійснюється у два кроки: вхід марке-

Таблиця 1. Параметри елементів стохастичної мережі Петрі з багатоканальними та конфліктними переходами

Назва елементу	Параметр	Тип значення	Зміст параметру
Позиція	Кількість маркерів у позиції	Ціле невід'ємне число	Символізує виконання умови для запуску переходу. При достатній кількості маркерів у вхідних позиціях переходу, виконується умова входу маркерів в перехід.
Перехід	Часова затримка	Дійсне невід'ємне число	Затримка між входом та виходом маркерів з переходу, що реалізується в алгоритмі імітації очікуванням просування часу до відповідного значення модельного часу. Затримка відтворює виконання певних дій в реальній системі.
	Пріоритет	Ціле число	При вирішенні конфлікту переходи з найбільшим пріоритетом з рівною ймовірністю можуть здійснити свій запуск. Конфлікт переходів виникає, коли одночасно для більш, ніж одного переходу виконана умова входу маркерів в перехід.
Вхідна дуга	Кількість зв'язків	Натуральне число	Визначає кількість маркерів, що вилучаються при вході маркерів в перехід. Вхід маркерів в перехід здійснюється за умови, що в усіх вхідних позиціях переходу наявна кількість маркерів у кількості, зазначеній у кількості зв'язків відповідної вхідної дуги.
Вихідна дуга	Кількість зв'язків	Натуральне число	Визначає кількість маркерів, що додаються при виході маркерів з переходу. Вихід маркерів з переходу відбувається, коли сплине часова затримка, зазначена Запуск переходу здійснюється за умови, що в усі вихідні позиції переходу додаються маркери у кількості, залежно від кількості зв'язків відповідної вхідної дуги.

рів в перехід та вихід маркерів з переходу (після того, як сплинула часова затримка). Параметри, які визначені для елементів, представлені у таблиці 1.

Петрі-об'єкт за визначенням, введеним у [15], є об'єктом суперкласу, який містить методи для відтворення динаміки об'єкта у відповідності до мережі Петрі, що зберігається у спеціально призначеному для цього полі. Створюючи об'єкт конструктор суперкласу використовує мережу Петрі та, можливо, набір параметрів, тому об'єкт можна представити виразом:

$$o = (net, list),$$

де *o* - Петрі-об'єкт, *net* - мережа Петрі, *list* - список параметрів.

Петрі-об'єктна модель створюється з множини Петрі-об'єктів, динаміка яких взаємопов'язана через спільні позиції. Поєднання Петрі-об'єктів через спільні позиції гарантує, що динаміка моделі визначається мережею Петрі, яка є об'єднанням мереж Петрі-об'єктів [15]. Програмно спільні позиції реалізуються через присвоєння адрес пам'яті відповідних об'єктів-позицій, що можна представити наступним виразом:

$$o_u.net.p_b = o_v.net.p_a,$$

де o_u, o_v – Петрі-об'єкти, які з'єднуються, *o.net* – мережа Петрі-об'єкта *o*, *o.net.p* – позиція мережі Петрі-об'єкта *o*. Усі пари спільних позицій двох Петрі-об'єктів, що утворюють з'єднання між ними, будемо називати конектором:

$$connector(o_u, o_v) = \{(o_u.net.p_b, o_v.net.p_a)\}.$$

Модель утворюється з Петрі-об'єктів операцією агрегування (в термінах об'єктно-орієнтованої технології), тому модель можна представити наступним виразом:

$$m \longrightarrow\!\diamond \{o_{1,\ldots,} o_n\},$$
$$m.net = \bigcup_j o_j.net,$$

де m – модель, —◇ – символ операції агрегування, o_j – Петрі-об'єкти, які агрегуються, *m.net* – мережа Петрі моделі, *o.net* – мережа Петрі-об'єкта *o*.

Перетворення стану мережі Петрі всієї моделі, як доведено у [15], розпадається на перетворення стану мереж Петрі-об'єктів. Це збільшує швидкодію імітації, оскільки, по-перше, при відтворенні події відбуваються зміни тільки у відповідному Петрі-об'єкті, по-друге, пошук найближчої події відбувається переглядом стану усіх об'єктів моделі замість перегляду стану усіх переходів моделі.

Оскільки при з'єднанні кількох Петрі-об'єктів утворюється знову мережа Петрі, а не структура більш високого рівня, то можуть бути введені операції над Петрі-об'єктами. Операція тиражування *multiply* означає, що мережа Петрі використовується для створення групи Петрі-об'єктів. При цьому списки чисельних параметрів елементів мережі Петрі можуть відрізнятись (окрім кількості маркерів у позиціях, які використовуються для з'єднання з іншими Петрі-об'єктами):

$$g = multiply(net, lists, k) \Leftrightarrow$$
$$\Leftrightarrow \forall i: o_i = (net, list_i), i = 1..k,$$

де g – група Петрі-об'єктів, *net* - мережа Петрі, що використовується для створення групи об'єктів, $lists = \{list_i\}$ - множина списків параметрів, що застосовуються для створення мережі Петрі конкретного об'єкта, k – кількість створюваних об'єктів.

З'єднання Петрі-об'єкта o з групою g Петрі-об'єктів за правилом, сформульованим для пари Петрі-об'єктів, фактично тиражує зв'язок між Петрі-об'єктом o та кожним Петрі-об'єктом з групи g:

$$g.net.p_b = o.net.p_a \Leftrightarrow$$
$$\Leftrightarrow \forall o_i \in g:\ o_i.net.p_b = o.net.p_a.$$

Об'єкти, взаємопов'язані між собою, можуть утворювати колекції , для яких можна здійснювати тиражування за наступними правилами:

де f – група колекцій Петрі-об'єктів, *objects* - Петрі-об'єкти, що використовується для створення колекції, *connectors* – конектори між Петрі-об'єктами, k – кількість операцій тиражування колекції.

Інші об'єкти можуть створювати з'єднання з групою колекцій :

$$f.g_u.net.p_b = o.net.p_a \Leftrightarrow$$
$$\Leftrightarrow \forall u \in g_u: f.u.net.p_b = o.net.p_a,$$

де з'єднання встановлюється між Петрі-об'єктом o та групою Петрі-об'єктів g_u з колекції.

У [16] розглянуто застосування дворівневої структури графічного представлення Петрі-об'єктної моделі на прикладі моделі телекомунікаційної системи і визначені переваги використання мови візуального програмування. У наступному розділі статті для запропонованої мови даний опис у вигляді формальної граматики та зроблено висновки про її властивості.

3. Граматика мови візуального програмування Петрі-об'єктних моделей

Мова візуального програмування визначає набори графічних елементів (символів), що можуть бути перетворені на змістові конструкції Петрі-об'єктної моделі. Графічні елементи, передбачені для конструювання Петрі-об'єктної моделі, є символами мови. Набори елементів, які можуть бути перетворені на імітаційні моделі і запущені на виконання – це лексеми (слова) мови. Мережі Петрі-об'єктів – це лексеми в граматиці. Вирази мови – це фрагменти Петрі-об'єктної моделі.

Введемо алфавіт термінальних символів для опису мережі Петрі: *p* - позиція, *t* - перехід, *a* – дуга. Введемо алфавіт нетермінальних символів: *E* - комбінація термінальних символів, що задає одну вхідну дугу переходу, *Q* - комбінація термінальних символів, що задає одну вихідну дугу переходу, *N* - комбінація кількох нетермінальних символів *E* та *Q*, що може бути інтерпретована в мережу Петрі. Символом *I* позначимо початковий символ граматики. Вважатимемо,

$$f = multiply(objects, connectors, k) \Leftrightarrow$$
$$\forall o \in objects, \forall c \in connectors, \forall u, v \in objects | (u.net.p_b, v.net.p_a) \in c:$$
$$\begin{cases} g_o = multiply(o.net, lists, k), \\ \forall j: c_j = \left\{ (u_j.net.p_b, v_j.net.p_a) \,\middle|\, u_j \in g_{u_j}, v_j \in g_{v_j} \right\} \end{cases}, j = 1..k,$$

що дуга, яка поєднує елементи «позиція» та «перехід», визначає відношення «попередній – наступний» у послідовності символів, що задає лексему мови програмування. Правила виведення граматики для введених символів представлені у таблиці 2. Правила визначають, що усі набори символів інтерпретуються як трійки нетермінальних елементів *pat* та *tap*. У найпримітивнішому випадку мережа Петрі складається з однієї позиції (без переходів). Мережа Петрі може містити позиції, які не з'єднані з переходами (такі позиції зберігають інформацію про стан об'єкта, а зміни в такому стані можуть відбутись тільки в результаті подій, що відбуваються в інших Петрі-об'єктах).

Для опису Петрі-об'єктної моделі доповнимо алфавіт термінальних символів граматики такими символами: *o* - Петрі-об'єкт, *g* - група Петрі-об'єктів, *f* – група колекцій Петрі-об'єктів, *s* – позиція Петрі-об'єкта, що використовується для з'єднання з іншими Петрі-об'єктами, *l* – ототожнення позицій Петрі-об'єктів. Вважатимемо, що конектор, який поєднує Петрі-об'єкт з іншим Петрі-об'єктом, або Петрі-об'єкт з групою Петрі-об'єктів, визначає відношення «попередній-наступний» у послідовності символів, що задає лексему мови програмування. Доповнимо алфавіт нетермінальних символів граматики такими символами: *Z* – комбінація символів, що визначає пару з'єднаних Петрі-об'єктів, *Y* - комбінація символів, що визначає з'єднання Петрі-об'єкта з групою Петрі-об'єктів, *X* - комбінація символів, що визначає з'єднання Петрі-об'єкта з групою колекцій, *C* - конектор Петрі-об'єктів, що складається з кількох ототожнень позицій Петрі-об'єкта з позиціями іншого Петрі-об'єкта. Правила виведення 7-12 граматики (для введених символів) представлені у таблиці 3.

Таблиця 2. Правила виведення 1-6 граматики (синтаксис мови)

Правило	Зміст правила
1. $I \longrightarrow NI \mid N$	Для визначення Петрі-об'єктів має бути визначена одна або кілька мереж Петрі.
2. $N \longrightarrow EQ \mid QE$	Якщо вказана вхідна дуга, то обов'язково має бути вихідна. Якщо вказана вихідна дуга, то обов'язково має бути вхідна.
3. $N \longrightarrow EN \mid QN$	Може бути задано будь-скільки вхідних та вихідних дуг у довільному порядку.
4. $E \longrightarrow pat$	Трійка термінальних символів, що визначають вхідну дугу переходу
5. $Q \longrightarrow tap$	Трійка термінальних символів, що визначають вихідну дугу переходу
6. $N \longrightarrow p \mid pN$	Одна або декілька позицій в мережі Петрі

Таблиця 3. Правила виведення 7-12 граматики

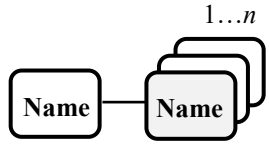
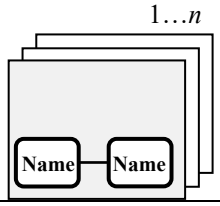
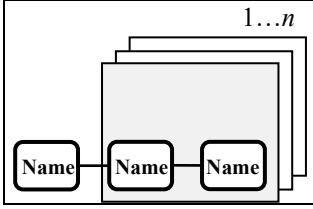
Правило	Зміст правила
7. $I \longrightarrow ZI \mid YI \mid XI$	Модель складається з трійок елементів, що визначають зв'язки між двома Петрі-об'єктами, між Петрі-об'єктом і групою Петрі- об'єктів, або групою колекцій Петрі-об'єктів та мереж Петрі для створення Петрі-об'єктів (див. правило 1)
8. $Y \longrightarrow oCg \mid g$	Трійка символів, що визначають зв'язок Петрі-об'єкта з групою Петрі-об'єктів. Ототожнення позицій, що вказані в конекторі, тиражуються для всіх Петрі-об'єктів групи. Допускається конструювання моделі з групи Петрі-об'єктів, що не мають зв'язків з іншими Петрі-об'єктами.
9. $X \longrightarrow oCf \mid f$	Трійка символів, що визначають зв'язок Петрі-об'єкта з групою Петрі-об'єктів, що входить до складу колекції. Ототожнення позицій, що вказані в конекторі, тиражуються для всіх Петрі-об'єктів групи. Допускається конструювання моделі з групи колекцій Петрі-об'єктів, що не мають зв'язків з іншими Петрі-об'єктами.
10. $Z \longrightarrow oCo \mid o$	Трійка символів, що визначають зв'язок Петрі-об'єкта з іншим Петрі-об'єктом. Допускається конструювання моделі з одного Петрі-об'єкта, що не має зв'язків з іншими Петрі-об'єктами.
11. $I \longrightarrow oI \mid gI$	Один Петрі-об'єкт, або група Петрі-об'єктів в моделі, або кілька Петрі-об'єктів без зв'язків, або кілька груп Петрі-об'єктів без зв'язків.
12. $C \longrightarrow slsC \mid sls$	Трійка символів, що визначають ототожнення позиції одного Петрі-об'єкта з позицією іншого Петрі-об'єкта. Конектор складається з одного або кількох ототожнювань позицій двох Петрі-об'єктів.

Графічні зображення символів мови представлені у таблиці 4. Параметри елементів, які представлені символами мови, мають у графічному представленні спеціально відведені поля. Параметр Name призначений для введення назви відповідного елементу.

Таблиця 4

Графічне зображення символів алфавіту мови візуального програмування Петрі-об'єктних моделей

Графічне представлення символу	Скорочене позначення символу	Інтерпретація
Name *k*	*p*	Позиція мережі Петрі з кількістю маркерів *k*.
Name *d*	*t*	Перехід мережі Петрі з часовою затримкою *d*, зі значенням пріоритету *r* (ширина прямокутника розраховується за формулою $w \cdot (1 + 0{,}5^r)$, де w – значення ширини, що відповідає пріоритету 1)
q	*a*	Дуга з кількістю зв'язків *q*. Значення *q*=1 є таким, що приймається за замовчуванням
1	*E*	Вхідна дуга переходу (з позиції)
1	*Q*	Вихідна дуга переходу (у позицію)
k	*N*	Фрагмент мережі Петрі, що складається з трьох вхідних дуг та трьох вихідних дуг: EEQEQQ.
Name placeName placeName placeName	*o*	Петрі-об'єкт із вказівниками на позиції, що можуть використовуватись для з'єднання з іншими Петрі-об'єктами.
placeName	*s*	Вказівник на позицію Петрі-об'єкта, що використовується для з'єднання
pName pName	*sls*	Ототожнення позиції Петрі-об'єкта з позицією іншого Петрі-об'єкта
	C	Конектор
Name **Name**	*Z*	Спрощене представлення Петрі-об'єкта та з'єднання двох Петрі-об'єктів
1…*n* **Name**	*g*	Група *n* Петрі-об'єктів, створених за однаковим шаблоном Петрі-об'єкта з назвою Name

	*Y*	З'єднання Петрі-об'єкта з групою *n* об'єктів
	*f*	Група *n* колекцій Петрі-об'єктів, створених за однаковим шаблоном. По суті, фрагмент моделі тиражується у кількості *n*.
	*X*	Група *n* колекцій Петрі-об'єктів, створених за однаковим шаблоном. По суті, це фрагмент моделі, що тиражується у кількості *n*.

Правила виведення 1-12 визначають граматику мови візуального програмування Петрі-об'єктних моделей. Наведена граматика є контекстно-вільною або граматикою типу 2 за ієрархією Хомського, оскільки інтерпретація будь-якого нетермінального символу не залежить від символів, які його оточують. Приклад виведення за визначеною граматикою (породження виразу з застосуванням послідовності правил 7, 10, 12, 7, 8, 12, 1, 3, 5, 2, 4, 5, 1, 3, 5, 3, 4, 2, 4, 5):

$$I \longrightarrow ZI \longrightarrow oCoI \longrightarrow oslsoI \longrightarrow oslsoYI \longrightarrow$$
$$\longrightarrow oslsooCgI \longrightarrow oslsooslsgI \longrightarrow$$
$$\longrightarrow oslsooslsgNI \longrightarrow oslsooslsgQNI \longrightarrow$$
$$\longrightarrow oslsooslsgtapNI \longrightarrow$$
$$\longrightarrow slsooslsgtapEQI \longrightarrow$$
$$\longrightarrow ooslsooslsgtappatQI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapI \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattapQN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptapN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptapEN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatN \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatEQ \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatpatQ \longrightarrow$$
$$\longrightarrow oslsooslsgtappattaptappatpattap.$$

Виведення може бути також представлене у вигляді дерева виведення (рис. 1). Наведений ланцюжок відповідає Петрі-об'єктній моделі, що складається з двох Петрі-об'єктів одного типу та групи Петрі-об'єктів іншого типу. Мережа Петрі об'єктів першого типу складається з однієї вхідної дуги та двох вихідних дуг з відповідно визначеними для них позиціями та переходами, а мережа Петрі другого типу складається з двох вхідних дуг та двох вихідних дуг.

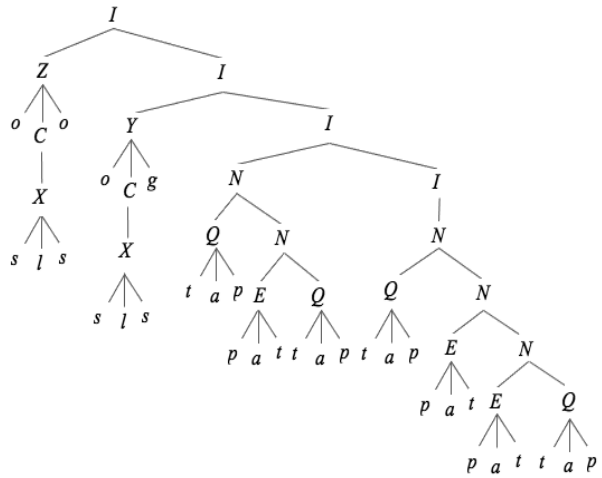


Рис. 1. Дерево виведення для моделі, представленої мовою візуального про-грамування.

При створенні елементів у графічному середовищі елементи одного типу зберігаються в масиві у послідовності створення розробником моделі. Враховуючи однозначність послідовності елементів у масивах, очевидно, що існує тільки одне дерево виведення для будь-якого візуального представлення моделі. Оскільки існує тільки одне дерево виведення для візуального представ-

лення моделі, то введена граматика є однозначною.

Алгоритми пошуку непродуктивних символів та недосяжних символів наведені у монографії [5]. Складаємо список нетермінальних символів, які можуть бути виведені з термінальних символів: *E, Q, N, C, Y, I*. Оскільки усі нетермінальні символи алфавіту потрапили до цього списку, то непродуктивних (зайвих) символів немає. Складаємо список нетермінальних символів, які можуть бути виведені з початкового символу: *I, Z, Y, N, C, E, Q*. Оскільки до списку потрапили усі нетермінальні символи граматики, то недосяжних символів у визначеній граматиці немає. Ланцюгових правил серед правил 1-11 немає. Отже, визначена граматика є приведеною.

Форма Бекуса-Наура – це інший спосіб введення граматики, який використовують, коли символів надто багато і правил виведення відповідно теж багато. Очевидно, що правила граматики 1-11 можна описати у вигляді розширеної нотації Бекуса-Наура.

Петрі-об'єктна-модель, описана візуальною мовою програмування, трансформується в алгоритмічну мову і запускається на виконання алгоритмом імітації. Правила перетворення візуальної моделі у модель обчислень визначають семантику мови.

4. Приклад представлення моделі мовою візуального програмування

Наведемо приклад розробки моделі з використанням розробленої мови візуального програмування. Побудуємо імітаційну модель, що відтворює обробку запитів клієнт-серверним застосунком. Запити надходять від користувачів двох типів: «авторів» та «гостей». Кожний тип користувача має свою логіку роботи з клієнт-серверним застосунком. Користувачі-автори, на відміну від користувачів-гостей, можуть додавати та змінювати інформаційний контент.

Загальною проблемою таких систем може бути довготривала обробка запиту через очікування доступу до бази даних та обробку даних серверним застосунком. Імітація системи на моделі може допомогти розробникам визначитися з параметрами обчислювальних ресурсів, які забезпечують прийнятне значення очікування відповіді на запит, та з вимогами до швидкодії окремих процесів обробки запитів.

Фрагмент моделі, що містить обробку запитів, які надходять від користувачів-гостей, представлений на рисунку 2. Мережа Server імітує обробку запитів на сервері. Оскільки запити надходять на пошук інформації або на перегляд, то використовуються два Петрі-об'єкти, створені з мережею Server, зі спільними позиціями-ресурсами Limit queue, Threads, DB access. Використання двох Петрі-об'єктів (замість одного) дозволяє відслідковувати обробку різних видів запитів окремо, а також використовувати в кожному фрагменті часові затримки, характерні для обробки відповідного запиту.

Позиції мереж Петрі, що можуть використовуватись для з'єднання з іншими Петрі-об'єктами, утворюють список позицій-сокетів Петрі-об'єкта. Такі позиції позначені сірим кольором на зображенні мережі Петрі (див. рис. 2) та представлені у списку досутпних для з'єднання позицій у зображенні Петрі-об'єкта (див. рис. 3). Три з'єднаних Петрі-об'єкти User, Server, Server утворюють фрагмент моделі, що імітує обробку запитів одного користувача. Щоб створити 50 користувачів, побудований фрагмент тиражуємо та з'єднуємо утворену групу колекцій з Петрі-об'єктами Server конектором Server-Server.

Модель, побудована з графічних елементів, перетворюється на обчислення алгоритмом імітаційного моделювання стохастичної мережі Петрі. Результати імітації Петрі-об'єктної моделі інформаційної системи наведені у публікації [17].

Висновки

Розроблена формальна граматика мови візуального програмування Петрі-об'єктних моделей. Синтаксичні правила складання мовних виразів визначені правилами виведення (продукціями), з яких випливає, що визначена граматика є контек-

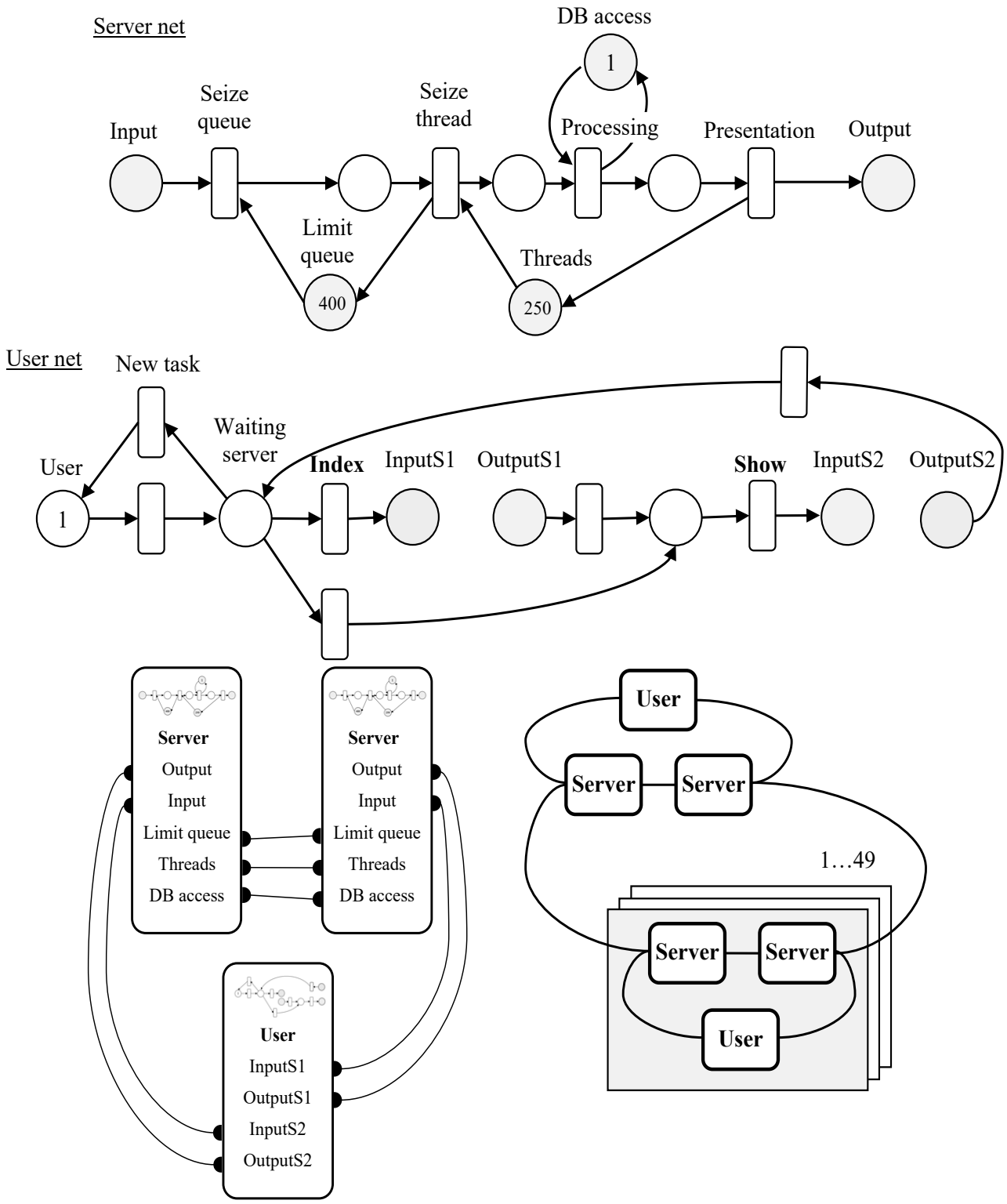


Рис. 2. Фрагмент Петрі-об'єктної моделі інформаційної системи, розроблений мовою ві-зуального програмування

стно-вільною і приведеною. Петрі-об'єктні моделі, породжені граматикою, можуть бути перетворені на обчислення, а саме запущені на виконання алгоритмом імітації. Семантика мови програмування визначається правилами перетворення мовних виразів на обчислення.

Поняття Петрі-об'єктної моделі розширено поняттями групи об'єктів, колекції об'єктів та групи колекції об'єктів, що дає змогу здійснювати тиражування об'єктів та зв'язків між об'єктами. Реалізація введених понять у мові візуального програмування уможливлює швидке створення та групу-

вання елементів моделі, динаміка яких визначається однаковим набором подій. За рахунок такого тиражування досягається також компактність візуального представлення для складних моделей.

Імітація побудованої моделі виконується алгоритмом імітаційного моделювання Петрі-об'єктної моделі.

References

1. ISO/IEC 15909-1:2004 Systems and software engineering — High-level Petri nets — Part 1: Concepts, definitions and graphical notation. [Online] – Available from: https://www.iso.org/standard/38225.html, last accessed 2020/08/28.
2. Zaitsev D.A.(2014) Paradigm of Computations on the Petri Nets, Automation and Remote Control, Vol. 75, No. 8, 1369–1383, https://doi.org/10.1134/S0005117914080025
3. Stetsenko I.V., Lytvynov V. (2020) Computer Virus Propagation Petri-Object Simulation. In: Palagin A., Anisimov A., Morozov A., Shkarlet S. (eds) Mathematical Modeling and Simulation of Systems. MODS 2019. Advances in Intelligent Systems and Computing, vol 1019, 103-112. Springer, Cham. https://doi.org/10.1007/978-3-030-25741-5_11
4. Shmeleva T.R., Stetsenko I.V. (2021) Modeling Unconditional Forwarding Decision Within Switching Lattice. In: Vorobiyenko P., Ilchenko M., Strelkovska I. (eds) Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212, 171- 186. Springer, Cham. https://doi.org/10.1007/978-3-030-76343-5_10
5. Stetsenko I.V., Pavlov A.A., Dyfuchyna O. (2021) Parallel algorithm development and testing using Petri-object simulation. International Journal of Parallel, Emergent and Distributed Systems. Taylor & Francis. 1-16. https://doi.org/10.1080/17445760.2021.1955113
6. Dahl, O.-J., Myhrhaug, B., Nygaard, K. (1970). Simula information. Common base language. (Report).NorwegianComputingCenter.[Online] – Available from: https://bibsys-k.userservices.exlibrisgroup.com/view/delivery/47BIBSYS_UBO/12216823070002204
7. Johnson M., Zelenski J. Formal Grammars. (2012) [Online] – Available from: https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/handouts/080%20Formal%20Grammars.pdf
8. Prochaska, K., Thiesing R. M. Introduction to Simio. (2016). Proceedings of the 2016 Winter Simulation Conference T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.
9. Simio. The future of Simulation, Growing with you. [Online] – Available from: https://www.simio.com/about-simio/why-simio/simio-is-the-future-of-simulation-software-growing-with-you.php
10. Simul8. Visual Logic Tutorial. [Online] – Available from: https://www.simul8.com/support/help/doku.php?id=features:visual_logic:tutorial
11. CPNTools. [Online] – Available from: http://cpntools.org/, last accessed 2020/04/26.
12. Jensen, K., Kristensen L. M. (2015) Colored Petri nets: a graphical language for formal modeling and validation of concurrent systems. Communications of the ACM 58(6), 61-70. DOI: 10.1145/2663340
13. Формальні мови, граматики та автомати: Навчальний посібник/ Гавриленко С.Ю. – Харків: НТУ « ХПІ », 2021. – 133 с.
14. Формальні мови та автомати / підручник для студ спец. 124 Системний аналіз / І.Я.Спекторський, В.М.Статкевич; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2019. – 167 с.
15. Стеценко И.В. (2011) Теоретические основы Петри-объектного моделирования систем. Математичні машини і системи, 136-148.
16. Stetsenko I.V., Dyfuchyn A. (2021) Petri-object Simulation Two Level Visual Programming Language. In: Palagin A., Anisimov A., Morozov A., Shkarlet S. (eds) Mathematical Modeling and Simulation of Systems. MODS 2020. Advances in Intelligent Systems and Computing, vol 1265. Springer, Cham. P. 266-276. (Scopus) https://doi.org/10.1007/978-3-030-58124-4_26
17. Stetsenko, I.V., Dyfuchyn, A.: Petri-object Simulation: Technique and Software. Information, Computing and Intelligent Systems 1, 51-59 (2020). https://doi.org/10.20535/2708-4930.1.2020.216057

Одержано 18.11.2021

Про авторів:

Дифучин Антон Юрійович,
аспірант 4 року навчання кафедри
інформатики та програмної інженерії
НТУУ “КПІ імені Ігоря Сікорського”.
Кількість наукових публікацій
в українських виданнях – 3.
Кількість наукових публікацій
в іноземних виданнях – 3.
Індекс Хірша – 1.
https://orcid.org/0000-0002-1722-8840

Стеценко Інна Вячеславівна,
доктор технічних наук, професор,
професор кафедри інформатики та
програмної інженерії НТУУ
“КПІ імені Ігоря Сікорського”.
Кількість наукових публікацій в
українських виданнях – понад 100.
Кількість наукових публікацій
в іноземних виданнях – 12.
Індекс Хірша – 2.
http://orcid.org/ 0000-0002-4601-0058

Жаріков Едуард В’ячеславович,
в.о. зав. кафедри інформатики та
програмної інженерії НТУУ “КПІ
імені Ігоря Сікорського”.
Кількість наукових публікацій в україн-
ських виданнях – понад 30.
Кількість наукових публікацій в іноземних
виданнях – 29.
Індекс Хірша – 4.
http://orcid.org/ 0000-0003-1811-9336

Місце роботи авторів:

Національний технічний
університет України
«Київський політехнічний
інститут імені Ігоря Сікорського»,
03056, м. Київ,
проспект Перемоги 37.
Тел.: (044) 236-9651
e-mail: difuchin@gmail.com
stiv.inna@gmail.com
zharikov.edward@gmail.com