MATHEMATICAL AND COMPUTER MODELING

DOI 10.20535/2411-1031.2019.7.1.184324
УДК 004(94+41)

ВОЛОДИМИР СОКОЛОВ

ТОПОЛОГІЇ СХЕМ АКТИВНИХ ДИНАМІЧНИХ СПОЛУК ОБ'ЄКТІВ

Представлено результати досліджень способів побудови схем сполук об'єктів, що використовуються для створення програмного забезпечення за архітектурою на основі інтегральних об'єктів. В основі визначення базових топологій схем сполук покладено класифікацію основних елементів сполук, що включають види об'єктів сполук, види зв'язків та види обчислень, що реалізує сполука. Об'єкти, з яких будуються будь-які сполуки, було розділено на чотири види: дані, функції, комутатори та активатори. Об'єкти-дані представляються змінними та константними об'єктами, які зберігають дані. Об'єкти-функції виконують перетворення вхідних даних у вихідні результати, реалізуючи певну функцію або композицію функцій. Об'єкти-комутатори виконують функцію динамічного управління зв'язками об'єктів в межах фіксованих з'єднань на схемі і представлені класами вимикачів, перемикачів та вибіркових перемикачів. Об'єкти-активатори призначені для побудови схем, що потребують багаторазової або послідовної активації об'єктів схеми, і представлені класами циклічних та покрокових активаторів. Види зв'язків об'єктів сполуки розділені на три види: активні або пасивні, фіксовані або комутовані, прямі або зворотні. Активність або пасивність зв'язків визначається видом вихідних конекторів, що приймають участь у певному з'єднанні. Фіксованість або комутованість зв'язків визначається відсутністю або наявністю комутаторів в схемі сполуки. Зворотній зв'язок дозволяє передавати дані об'єкта від його виходу на його вхід або на вхід попередніх об'єктів чи використовувати рекурсію. Види обчислень, що реалізує сполука розбиті на три види: функціональні, ітераційні або покрокові обчислення. Функціональні обчислення мають місце, коли не використовуються стани об'єктів. Ітераційні обчислення вимагають повторної активації тих самих об'єктів в процесі обчислення результату, а покрокові схеми послідовно активують різні об'єкти схеми в заданому порядку. На основі розробленої класифікації елементів схем сполук було визначено три види базових топологій схем сполук: комбінаційні схеми, схеми зі зворотним зв'язком та схеми з контролером. Комбінаційні схеми реалізують функціональні обчислення. Схеми зі зворотним зв'язком можуть реалізовувати як функціональні, так і ітераційні (рекурсивні) обчислення. Схеми з контролером використовують активатори для реалізації ітераційних та покрокових обчислень. Шляхом комбінації базових топологій можна отримувати складні схеми сполук.

Ключові слова: топології схем сполук; інтегральні об'єкти; сполуки об'єктів; комбінаційні схеми; схеми зі зворотним зв'язком; схеми з контролером.

Постановка проблеми. Програмне забезпечення, в основі якого лежать сполуки інтегральних об'єктів, може використовувати не тільки відносно прості послідовно-паралельні комбінаційні схеми сполучення об'єктів, а й більш складні структури, які потребують зворотного зв'язку, заданої логіки перемикання зв'язків об'єктів в програмі, ітераційних процесів та складних елементів даних. Звісно, що будь-яку послідовність обчислень можна реалізовувати в функціях ядер атомарних об'єктах, але така реалізація буде обмежувати можливості для створення програм з повторним використанням класів, придатних для дослідження їх структурних характеристик та верифікації. Тому розробка способів побудови складних схем програм, базових варіантів топології сполучення об'єктів програм та типових класів управляючих об'єктів є актуальною проблемою розвитку архітектури програмного забезпечення на основі інтегральних об'єктів (АІО).

© В. Соколов, 2019

Аналіз останніх досліджень та публікацій. Дослідженню програм, що задаються у формі схем, присвячено багато робіт. Зокрема, в [1] запропоновано метод та програмні засоби розробки послідовних і паралельних адаптивних алгоритмів на основі використання параметрично керованої генерації схем програм. В роботах [2], [3] розглядаються механізми автоматизації генерації програм, включаючи графове представлення об'єктно-орієнтованих програм предметної області, а також виділяються функціональні та інтерфейсні об'єкти, що взаємодіють за об'єктним графом. В роботі [4] розглядаються реактивні шаблони проектування та аналіз схем потоків управління в програмах, що відповідають заданим критеріям надійності в розподіленому середовищі. В зазначених роботах представлення та аналіз програм у вигляді схем носить або загальний характер, незалежно від парадигми програмування, або стосується окремих аспектів представлення програм. Проблема, що вирішується в цій роботі, більше стосується архітектури програмного забезпечення на основі інтегральних об'єктів, що представлена в роботі автора [5], та використовує представлення сполук інтегральних об'єктів, що наведено в роботі [6]. Робота [7] розглядає схеми декомпозиції алгоритмічної природи та графічне представлення програм, що складаються з алгоритмічних фреймів та функціональних ядер, що дозволяю генерувати алгоритми та прототипи програмного коду. В роботі [8] запропоновано інтерактивний графік викликів функцій для аналізу великих програм, що дозволяє досліджувати і переглядати множини функцій з можливістю переходу до вихідного коду функції. В роботі [9] запропоновано підхід до візуалізації програмного забезпечення шляхом представлення класів прямокутниками, методів і атрибутів – колами і трикутниками відповідно, а відношення між класами – стрілкам, що певним чином є альтернативою відомим графічним представленням. Як видно з аналізу публікацій, графічне представлення програм є актуальною науково-технічною задачею.

Метою статті є розробка та дослідження базових топологій схем сполук об'єктів на основі класифікації видів об'єктів, зв'язків та обчислень, що забезпечує розширення області застосування АІО для розробки програм розв'язку широкого класу задач.

Виклад основного матеріалу дослідження.

Класифікаціяелементів сполук. Під топологією сполук будемо розуміти структуру зв'язків між об'єктами сполуки та спосіб організації обчислень. В основі аналізу топології лежить класифікація складових схем сполук, таких як види об'єктів та зв'язків між об'єктами, що суттєво впливає на види обчислень, що реалізують сполуки. В результаті проведених досліджень було визначено наступну класифікацію елементів сполук.

1. Види об'єктів сполук:
   - дані – об'єкти класів простих або структурованих даних (константи та змінні);
   - функції – об'єкти класів, що перетворюють входи у виходи;
   - комутатори – об'єкти класів, що управляють зв'язками об'єктів сполуки;
   - активатори – об'єкти класів, призначених для активації об'єктів сполуки.
2. Види зв'язків об'єктів сполуки:
   - активні або пасивні;
   - фіксовані або комутовані;
   - прямі або зворотні.
3. Види обчислень, що реалізує сполука:
   - функціональні – сполука реалізує функцію або композицію функцій;
   - ітераційні – певні об'єкти сполуки активуються багаторазово;
   - покрокові – різні об'єкти сполуки активуються в заданому порядку.

Розглянемо детальніше зазначені складові сполук.

Види об'єктів сполук. Різноманіття даних може включати безліч класів, основною ознакою яких є відсутність перетворень вхідних значень у вихідні результати (вихід дорівнює входу). Типовими представниками класів даних є прості дані (див. рис. 1), статичні масиви та структури (див. рис. 2), а також динамічні структури даних. При цьому статичні

структури даних можуть використовувати статичні зв'язки в функціональних обчисленнях, а динамічні структури даних потребують, як правило, комутованих зв'язків та ітераційних або покрокових обчислень.

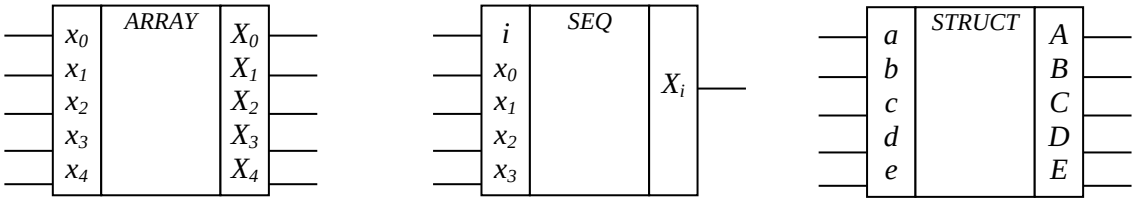
Доцільним для даних є використання додаткового пасивного вихідного конектора на додачу до активного, як показано на рис. 1а. При цьому звертання до активного виходу *X* може призводити до оновлення даних шляхом звертання до входу *x*, а звертання до пасивного виходу X_0 повертає поточне значення без активації входу*x*. Така реалізація дозволяє використовувати попереднє значення даних в схемах зі зворотним зв'язком, що буде продемонстровано нижче.

а) прості дані з пасивним та активним виходом б) проста константа зі значенням 1

Рисунок 1 – Класи простих даних

Масиви, як відомо, містять дані одного типу та допускають паралельний або послідовний (за номером елементу) доступ (див. рис. 2 а, б). Структури містять дані різного типу і, як правило, реалізуються інтегральними класами, в яких об'єднуються об'єкти інших класів даних (див. рис. 2 в).

Класи даних можуть бути різноманітні, в тому числі й представляти зовнішні файли даних, таблиці та реляційні бази даних, що суттєво впливає на вид обчислень.



а) паралельний масив б) паралельно-послідовний масив в) структура

Рисунок 2 – Приклад класів структурованих даних

Функціональні класи об'єктів, на відміну від класів даних, завжди виконують перетворення даних та можуть бути наскільки завгодно складними. Приклад класу, що реалізує функцію $z = F(x,y)$, представлено на рис. 3.

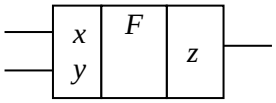
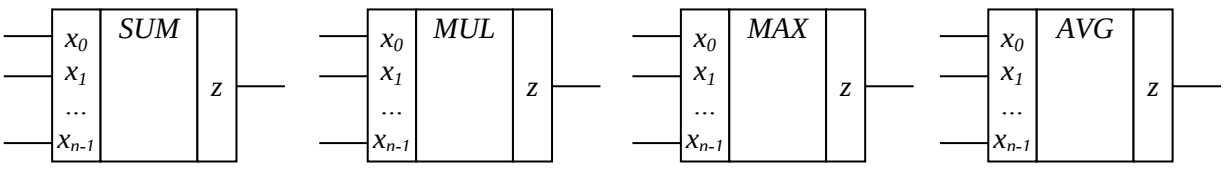


Рисунок 3 – Приклад класу функціональних об'єктів

При обробці структурованих даних, наприклад, корисними є класи агрегатних функцій, що представлені на рис. 4.



а) сума б) добуток в) максимум г) середнє

Рисунок 4 – Приклад класів агрегатних функцій

Класи комутаторів управляють статично-динамічними зв'язками об'єктів сполуки шляхом виконання функцій вмикання/вимикання абоперемикання зв'язків. Ці класи дозволяють реалізовувати різні види динамічно розгалужених процесів в межах заданих зв'язків об'єктів.

Умовне розгалуження реалізується класами елементів управління IF та IFE (див. рис. 5).

а) перемикач IFE (if-else) б) вмикач/вимикач IF

Рисунок 5 – Класи елементів розгалуження

Клас IFE реалізує функцію ядра, яка вихід *z* динамічно зв'язує або з входом *x*, якщо умова *c* є істинною, або з входом *y*, інакше

$$z = \begin{cases} x \mid c = true \\ y \mid c = false \end{cases}$$

Клас IF реалізує функцію ядра, яка вихід *z* динамічно зв'язує з входом *x*, якщо умова *c* є істинною, або повертає NULL, інакше

$$z = \begin{cases} x \mid c = true \\ \varnothing \mid c = false \end{cases}$$

Можлива реалізація групового розгалуження шляхом об'єднання в інтегральному класі декількох об'єктів розгалуження з загальною умовою.

Вибіркове розгалуження (перемикання) реалізується класами комутаторів SWITCH та SWITCHE (див. рис. 6).

Клас SWITCHE реалізує функцію ядра, яка вихід z динамічно зв'язує з одним із існуючих входів, номер якого заданий входом *i*,або з входом *y*, інакше

$$z = \begin{cases} x_i \mid i \in [0, n-1] \\ y \mid i \notin [0, n-1] \end{cases}$$

Клас SWITCH реалізує функцію ядра, яка вихід *z* динамічно зв'язує з одним із існуючих входів, номер якого заданий входом *i*,або повертає NULL, інакше

$$z = \begin{cases} x_i \mid i \in [0, n-1] \\ \varnothing \mid i \notin [0, n-1] \end{cases}$$

а) повна форма б) коротка форма

Рисунок 6 – Класи комутаторів

Класи активаторів призначені для активації виходів певних об'єктів і на відміну від класів об'єктів даних та функцій не виконують передавання даних. Активатори є складовими контролерів в архітектурі MVC.

Клас циклічного активатору FOR (див. рис. 7а) реалізує задану кількість активації власного входу *x* наступним чином: при активації виходу *z* в функції ядра змінна і приймає

початкове значення *a* і далі з кроком *s* збільшується, поки не досягне кінцевого значення *b*, активуючи на кожній ітерації вхід x та формуючи поточне значення на виході i. Вихід z отримує значення стану об'єкту (при нормальному завершенні всіх ітерацій дорівнює 1) тільки після закінчення роботи функції ядра. Клас циклічного активатору FORI (див. рис. 7б) виконує ітерацію змінної i від 0 до *n*-1 з кроком 1.

а) повна форма FOR б) скорочена форма FORI

Рисунок 7 – Клас циклічних активаторів FOR

Клас циклічного активатору WHILE циклічно активує власний вхід *x* в процесі виконання функції ядра, поки істинна умова на вході *c*. Вихід *z* отримує значення стану об'єкту тільки після закінчення роботи функції ядра.

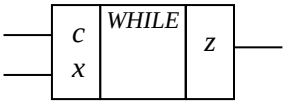


Рисунок 8 – Клас циклічного активатору WHILE

Клас покрокового активатора STEP при активації виходу *z* послідовно активує входи x_i, видаючи на пасивний вихідний конектор *i* номер поточного кроку.

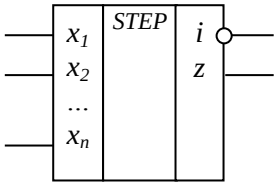


Рисунок 9 – Клас покрокового активатора STEP

Види зв'язків об'єктів сполуки. Види зв'язків об'єктів сполуки визначаються видом конекторів (активні або пасивні), детермінованістю (фіксовані або комутовані) та напрямком (прямі або зворотні), що суттєво впливає як на загальну топологію сполуки, так і на характер обчислень. Детальніше види зв'язків буде розглянуто при розгляді топології.

Види обчислень, що реалізує сполука. Вид обчислень, що реалізує сполука, характеризує спрямованість обчислювальних процесів, порядок активації та повторного використання об'єктів сполуки, що впливає на можливість формального опису та аналізу схеми, використання паралельних процесів.

Функціональні обчислення мають місце, коли сполуку можна розглядати як композицію функцій об'єктів без стану, що піддається формальному опису та аналізу.

Ітераційні обчислення мають місце, коли ті ж самі об'єкти сполуки активуються багаторазово для отримання результату, порядок активації визначається схемою зворотних зв'язків або активаторами і, як правило, використовуються стани об'єктів.

Покрокові обчислення характеризуються тим, що порядок активації об'єктів сполуки визначається контролером.

Базові топології схем сполук. Внаслідок взаємного впливу та обмежень елементів сполук (видів об'єктів, зв'язків та обчислень) визначено наступні базові топології схем сполук.

1. Комбінаційна схема:
   - з фіксованим зв'язком;
   - з комутованим зв'язком.
2. Схема зі зворотним зв'язком:
   - з пасивним зворотним зв'язком;
   - з активним зворотним зв'язком.
3. Схема з контролером:
   - ітераційна схема;
   - покрокова схема.

Шляхом комбінації базових топологій можна отримувати складні схеми сполук.

1. Комбінаційна схема. Комбінаційна схема (КС) сполуки – це схема без зворотних зв'язків, в якій вихід сполуки залежить тільки від входу і не залежить від стану об'єктів. При повторній активації (обчисленні) сполуки при незмінних вхідних даних результат не змінюється. Реалізує функціональні обчислення. В комбінаційних схемах циркулюють виключно дані та відсутні сигнали управління. Порядок активації виходів не впливає на результат. Є чистою моделлю в архітектурі MVC. Приклад КС наведено на рис. 10, а формальне табличне представлення наведено в табл.1.

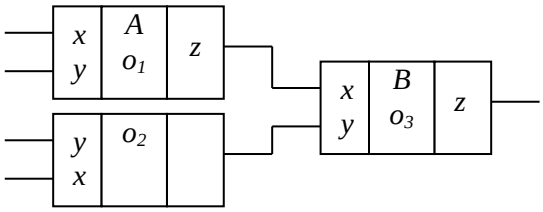


Рисунок 10 – Приклад сполуки S_1 з комбінаційною схемою

Таблиця 1 – Табличне представлення сполуки S_1

S_1	x	y	z	x	y	z	x	y	z
A	-1	-1	1						
A				-1	-1	1			
B							-1	-1	1
=			-1				1		
=						-1		1	

Перевірка КС на ациклічність (відсутність зворотних зв'язків) може бути виконана методом "водоспадного" упорядкування або методом редукції таблиці.

Метод "водоспадного" упорядкування таблиці полягає в тому, щоби шляхом переставлення стовпчиків та рядків привести таблицю до форми, коли в кожному рядку всі входи знаходяться зліва від виходів, а в стовпчиках виходи – вище входів. Якщо це можливо, то схема сполуки є ациклічною.

Таблиця 2 – Упорядкована таблиця сполуки S_1

S_1	x	y	x	y	z	z	x	y	z
A	-1	-1			1				
A			-1	-1		1			
=					-1		1		
=						-1		1	
B							-1	-1	1

Метод редукції таблиці є більш придатний для практичного застосування і полягає в тому, щоби ітераційно викреслювати (“обнуляти”) в таблиці стовпчики “вільних” входів (що містять тільки -1) та “вільних” виходів (що містять тільки 1), а також вироджених внаслідок цього рядків (що не містять жодних виходів) до тих пір, поки не отримаємо нульову (пусту) таблицю, що є підтвердженням ациклічності. Застосування цього методу буде надано при розгляді схем зі зворотним зв’язком.

Приклад КС сполуки з умовним розгалуженням, що знаходить найбільше з двох значень, наведено на рис.11.

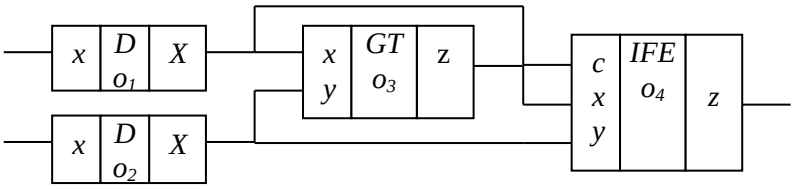


Рисунок 11 – Приклад КС сполуки з комутованим зв’язком IFE

Приклад схеми сполуки з вибірковим розгалуженням, що повертає значення *i*-го елементу паралельного масиву, наведено на рис.12.

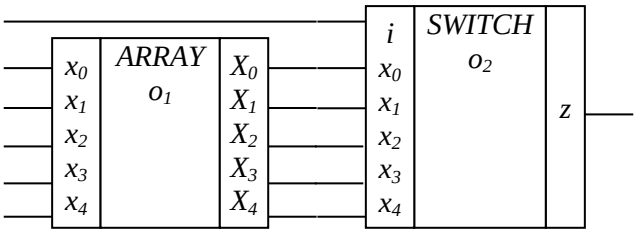


Рисунок 12 – Приклад КС сполуки з комутованим зв’язком SWITCH

2. Схема зі зворотним зв’язком. Схема зі зворотним зв’язком (СЗЗ) містить залежність входів об’єктів від власних виходів (прямо або через інші об’єкти). Застосовується в різних видах обчислень. Наприклад, на рис. 13 наведена сполука S_2 з пасивним зворотним зв’язком, а табличне представлення наведено в табл. 3.

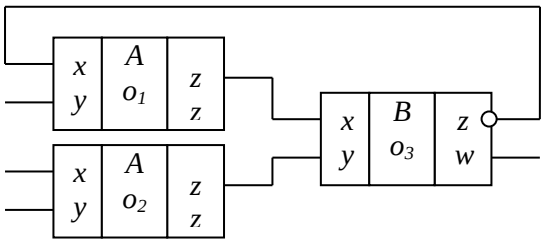


Рисунок 13 – Приклад сполуки S_2 зі схемою зі зворотним зв’язком

Таблиця 3 – Таблиця сполуки S_2

S_2	x	y	x	y	z	z	x	y	z	w
A	-1	-1			1					
A			-1	-1		1				
=					-1		1			
=						-1		1		
B							-1	-1	1	1
=	1								-1	

Для перевірки на наявність зворотних зв'язків застосуємо метод редукції таблиці (див. табл. 4).

Таблиця 4 – Таблиця сполуки S_2 після редукції

S_2	x	y	x	y	z	z	x	y	z	w
A	-1				1					
A										
=					-1		1			
=										
B							-1		1	
=	1								-1	

В СЗЗ для використання зворотного зв'язку можуть застосовуватися пасивні вихідні конектори (вихід *z* об'єкту o_3) для запобігання безкінечної рекурсії, тому що звертання до пасивного вихідного конектора не призводить до виклику функції ядра, яке в свою чергу зверталась би до його входів. Пасивний конектор повинен мати початкове значення і змінюватись ядром при звертанні до іншого активного вихідного конектора того ж об'єкту (вихід *w* об'єкту o_3). Зворотний зв'язок можна розглядати як спосіб передачі даних до попередніх об'єктів.

СЗЗ можна використовувати для обчислень, в яких потрібно виконувати обробку попередніх значень виходу об'єкта, наприклад, $x = x + y$. Приклад такої схеми наведено на рис. 14, яка по суті реалізує формулу $x = x_0 + y$.

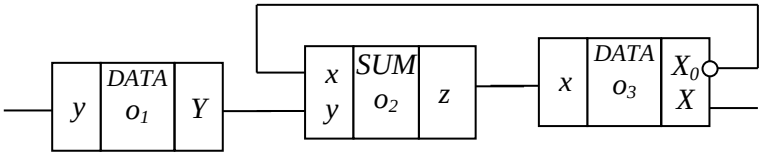


Рисунок 14 – Приклад СЗЗ обчислення виразу виду $x = x + y$

При використанні активних зворотних зв'язків отримуємо рекурсивні схеми, але для запобігання безкінечної рекурсії в рекурсивному ланцюжку потрібно вставляти пасивний зв'язок для отримання контрольованої рекурсії. Наприклад, на рис. 15 наведена сполука з контрольованою рекурсією.

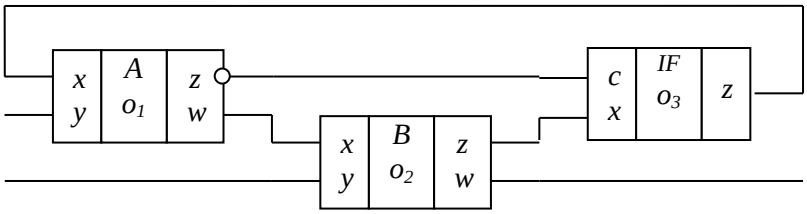


Рисунок 15 – Приклад СЗЗ з контрольованою рекурсією

3. Схема з контролером. Схема з контролером містить, як правило, об'єкти-активатори, що ітераційно або покроково активують об'єкти сполуки в процесі її обчислення. Можуть використовувати зворотній зв'язок та стан об'єктів. Обчислюються таким чином, що спочатку активується контролер для обчислення схеми, а потім використовуються вихідні результати.

Розглянемо приклад КС сполуки, яка виконує добуток двох паралельних масивів, що наведена на рис. 16. В КС використовується чотири об'єкти множення ($o_3 - o_6$) для кожної пари елементів масивів. Результат формується в об'єкті o_7.

Цю КС можна замінити на ітераційну схеми з використанням одного об'єкту добутку та послідовного застосування його до відповідних пар елементів паралельно-послідовних масивів. Приклад схеми сполуки з ітерацією, що виконує добуток двох паралельно-послідовних масивів o_1 та o_2, наведено на рис. 17. Результат формується в об'єкті o_4.

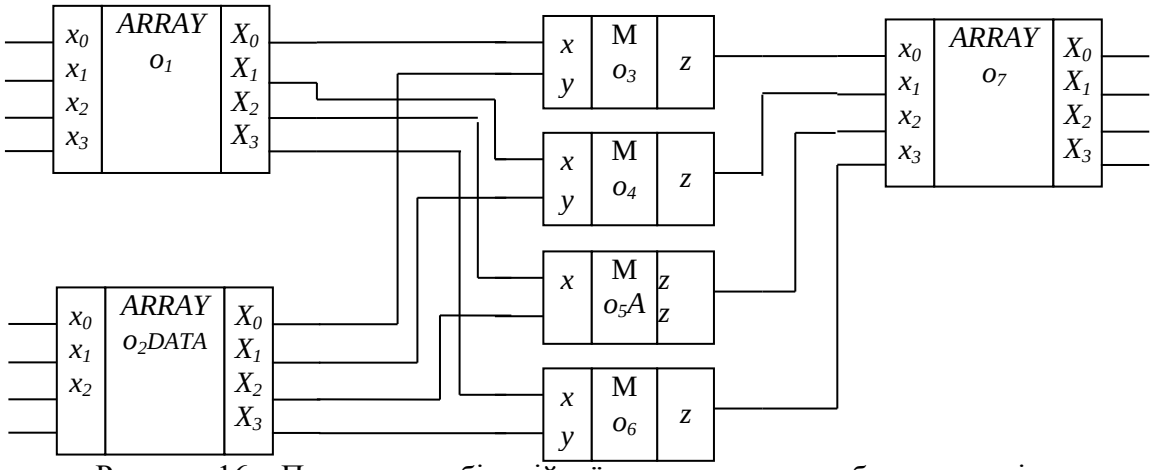


Рисунок 16 – Приклад комбінаційної схеми сполуки добутку масивів

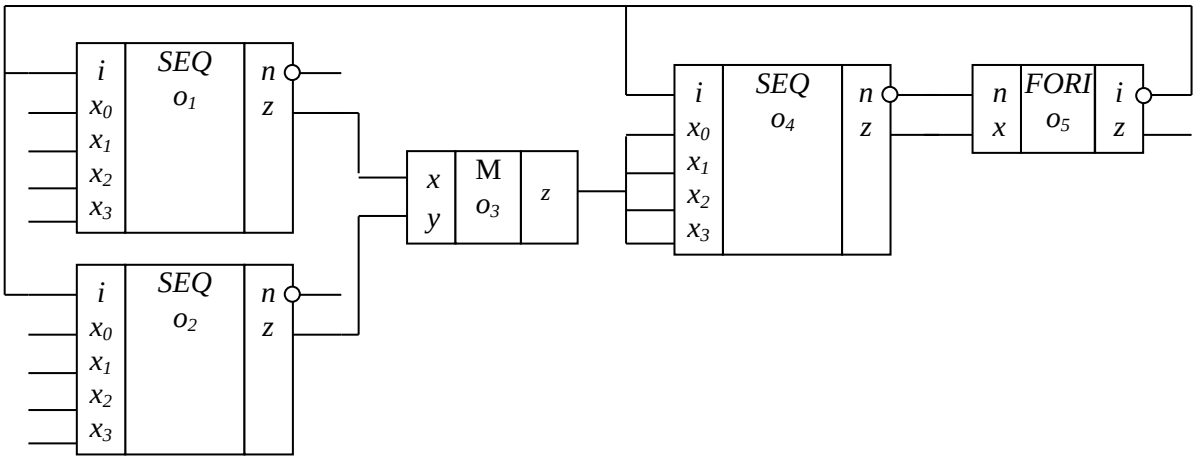


Рисунок 17 – Приклад сполуки з ітераційною схемою обчислень

Покрокові схеми застосовуються, коли послідовність обчислень сполуки неможливо задати схемою зв'язків або важливий порядок активації об'єктів. Наприклад, на рис. 18 наведено схема сполуки, яка реалізує послідовність двох обчислень: $y = y \cdot i; i = i + 1$.

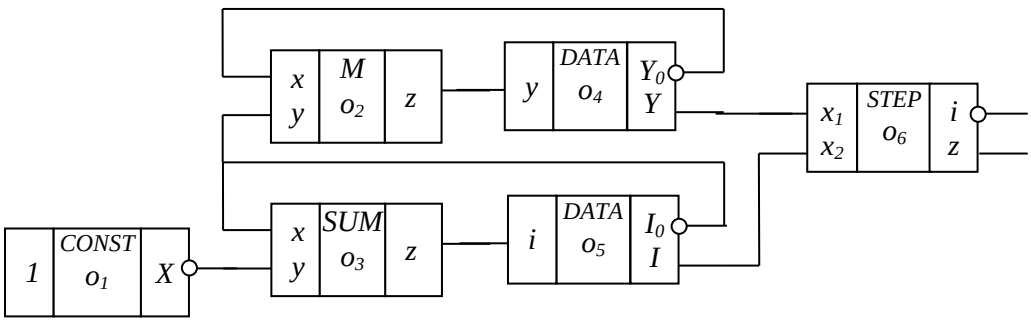


Рисунок 18 – Приклад схеми з покроковим активатором STEP

При активації контролера o_6 спочатку активується його вхід x_1, який в свою чергу активує об'єкти $o_4 - o_2 - o_5$ для обчислення першого виразу, а потім вхід x_2 активує об'єкти $o_5 - o_3 - o_1$ для обчислення другого виразу.

Висновки. Виконано класифікацію елементів схем сполук, визначено основні види об'єктів, зв'язків та обчислень активних динамічних сполук об'єктів, а також розроблено набір базових топологій схем сполук, що задовольняють практичні потреби програмування та розширюють область застосування архітектури програмного забезпечення на основі інтегральних об'єктів.

Перспективами подальших досліджень є формальний опис топологій не комбінаційних схем, визначення метрик та дослідження їх характеристик, а також створення методів проектування класів сполук розв'язку задач предметної області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] А. Ю. Дорошенко та ін., "Формальні та адаптивні методи й інструментальні засоби паралельного програмування", *Проблеми програмування*, № 3, с. 19-30, 2017.

[2] Е. М. Лаврищева, "Парадигмы программирования сборочного типа в программной инженерии", на *Міжнародній науково-практичній конференції з програмування УкрПРОГ*-2014, Київ, 2014, с. 76-92.

[3] П. І. Андон, та О. О. Слабоспицька, "Збіркове програмування компонентних і сервіс-орієнтованих прикладних програмних систем", *Проблеми програмування*, № 3, с. 31-51, 2017.

[4] Р. Кун, Б. Ханафи, и Дж. Аллен, *Реактивные шаблоны проектирования*, Санкт-Петербург, Российская Федерация: Питер, 2018.

[5] В. Соколов, "Архітектура програмного забезпечення на основі інтегральних об'єктів", *Information Technology and Security,* vol. 5, iss. 2, pp. 51-59, July-December 2017. doi: 10.20535/2411-1031.2017.5.1.120559.

[6] В. Соколов, "Структурний аналіз сполук інтегральних об'єктів", *Information Technology and Security*, vol.6, iss. 2, pp. 68-78, July-December 2018. doi: 10.20535/2411-1031.2018.6.2.153491.

[7] В. Г. Колесник, "DS-теория. Представление канонического алгоритма с помощью алгоритмического языка", *Проблеми програмування,* № 1, с. 3-18, 2015.

[8] M. D. Shah, and S. Z. Guyer, "An Interactive Microarray Call-Graph Visualization", in Proc. *IEEE Working Conference on Software Visualization*, Raleigh, USA, 2016, pp.86-90, doi**:** 10.1109/VISSOFT.2016.14.

[9] M. Alnabhan, A. Hammouri, M. Hammad, M. Atoum, and O. Al-Thnebat, "2D visualization for object-oriented software systems", in *Proc. International Conference on Intelligent Systems and Computer Vision (ISCV),* Fez, Morocco, 2018, pp. 1-6. doi**:** 10.1109/ISACV.2018.8354085.

Стаття надійшла до редакції 23.03.2019.

REFERENCE

[1] A. Doroshenko et al., "Formal and adaptive methods and tools of parallel programming", *Problems in programming*, no. 3, pp. 19-30, 2017.

[2] E. Lavrishcheva, "Paradigms of assembling kind programming in software engineering", in *Proc. International conf. UkrPROG*, Kyiv, 2014, pp. 76-92.

[3] P. Andon, O. Slabospitska, "Assembling kind programming of component and service-oriented applied software systems", *Problems in programming*, no. 3, pp. 31-51, 2017.

[4] R. Kuhn, B. Hanafee, and J. Allen, *Reactive Design Patterns*, Saint-Petersburg, Russia, 2018.

[5] V. Sokolov, "Architecture of software based on integrated objects", *Information Technology and Security*, vol.5, no. 2, pp. 51-59, July-December 2017. doi: 10.20535/2411-1031.2017.5.1.120559.

[6] V. Sokolov, "Structural analysis of the compounds of integral objects", *Information Technology and Security*, vol. 6, no. 2, pp. 68-78, July-December 2018. doi: 10.20535/2411-1031.2018.6.2.153491.

[7] V. Kolesnik, "DS-theory. Presentation of canonical algorithm by means of algorithmic language", *Problems in programming*, no. 1, pp. 3-18, 2015.
[8] M. D. Shah, and S. Z. Guyer, "An Interactive Microarray Call-Graph Visualization", in Proc. *IEEE Working Conference on Software Visualization*, Raleigh, USA, 2016, pp.86-90, doi**:** 10.1109/VISSOFT.2016.14.
[9] M. Alnabhan, A. Hammouri, M. Hammad, M. Atoum, and O. Al-Thnebat, "2D visualization for object-oriented software systems", in *Proc. International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2018, pp. 1-6. doi**:** 10.1109/ISACV.2018.8354085.

VOLODYMYR SOKOLOV

TOPOLOGIES OF SCHEMES OF ACTIVE DYNAMIC COMPOUNDS OF OBJECTS

The research results of methods for constructing schematics of objects compounds used to create software with architecture based on integral objects are presents in the paper. The basis for determining the main topologies of the compounds schemes is the major elements of compounds classification, including the types of objects of compounds, types of connections and types of calculations that the compounds implements. The objects from which are built compounds were divided into four types: data, functions, switches, and activators. Data objects are represented by variable and constant objects that store data. Function objects perform transformations of input data into output results, implementing a specific function or composition of functions. Switch objects perform the function of dynamic connections control within fixed connections in the scheme and are represented by classes of interrupters, switches, and selective switches. Activator objects are intended for constructing schemes that require multiple or sequential activations of objects and are represented by classes of cyclic and step-by-step activators. Types of connections between objects in compounds are divided into three types: active or passive, fixed or switched, forward or backward. Activity or passivity of connections is determined by the output connector's type, involved in a particular connection. The fixity or commutation of connections is determined by the absence or presence of switches in the compound scheme. Feedback allows transferring object data from its output to its input or to the input of previous objects or to use recursion. The calculations types that implement by compounds are divided into three types: functional, iterative, and step-by-step calculations. Functional calculations take place when the states of objects are not used. Iterative calculations require the reactivation of the same objects in the process of calculating the result, and step-by-step schemes sequentially activate various objects of the scheme in a defined order. Based on the developed classification of the compounds scheme's elements, three types of basic topologies were defined: combinational schemes, feedback schemes and schemes with a controller. Combination schemes implement functional calculations. Feedback schemes can implement both functional and iterative (recursive) calculations. Schemes with a controller used for activators to implement iterative and step-by-step calculations. By combining basic topologies, complex schemas can be obtained.

Keywords: topologies schemes of compounds; integrated objects; compounds of objects; combination schemes; feedback schemes; schemes with controllers.

Соколов Володимир Володимирович, кандидат технічних наук, доцент, доцент кафедри кібербезпеки і застосування інформаційних систем і технологій, Інститут спеціального зв'язку та захисту інформації національного технічного університету "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна.

ORCID: 0000-0002-5779-7167.

E-mail: vsokolov@i.ua.

Sokolov Volodymyr, candidate of technical sciences, associate professor, associate professor at the cyber security and application of information systems and technology academic department, Institute of special communication and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv polytechnic institute", Kyiv, Ukraine.