

Процик Петро Павлович,
Аспірант кафедри теорії та технології програмування
Київського національного університету ім. Тараса Шевченка
Адреса – Кафедра теорії та технології програмування,
Київський національний університет ім. Тараса Шевченка,
01033, Київ, вул. Володимирська, 69,
E-mail: ppp_extr@acm.org

Композиційно-номінативний аналіз архітектур програмних систем

Процик П. П.

Приводяться означення та пропонується класифікація різних типів програмних систем на композиційно-номінативній основі. Такий методологічний базис дає можливість явно визначити відмінні риси та особливості кожного типу системи, встановити зв'язок між ними та проаналізувати взаємозалежності.

Домінуючою парадигмою у програмній інженерії на даний час є об'єктно-орієнтована парадигма. Згідно неї, архітектура програмної системи визначається компонентами (об'єктами) та зв'язками (взаємодією) між ними, що є важливим, оскільки у композиційно-номінативному підході особливо підкреслюється вагомість зв'язків та необхідність їх дослідження. В залежності від рівня абстрактності компонентами системи можуть бути: функції, об'єкти, модулі, процеси, підсистеми.

Згідно [4] програмна система – це специфічна форма фіксації рівня абстракції. Справді, існуючі типи програмних систем отримані шляхом розвитку поняття обчислюваної функції у загальному розумінні. Яскравим прикладом найбільш загального типу системи, є закрита система: *закрита система* (Blackbox system) – система, про структуру якої нічого не відомо, крім того, що отримуючи вхідні дані вона або обчислює результат, або працює над ними нескінченно довго (заціклюється). Зауважимо, що циклювання зовсім не означає, що система не виконує корисних дій. Навпаки існує цілий широкий клас таких систем – неперервні системи (Continuous systems), до яких входять реактивні системи, системи реального часу.

Для опису даних та станів систем використовується поняття номінативних даних – ND. Вони будуються індуктивно на основі множин імен V і значень W : 1) $ND^{(0)} = W$, 2) $ND^{(i+1)} = ND^{(i)} \cup (V \xrightarrow{m} ND^{(i)})$, $i \in w$. Тоді $ND = \bigcup_{i \in w} ND^{(i)}$. Номінативні дані дозволяють адекватно задавати різні структури даних (дерева, списки, записи, тощо). Номінативні дані можна використовувати у різних підходах та формальних мовах опису, конкретизуючи необхідні аспекти.

Крім уже зазначених типів систем, проаналізовано наступні: послідовні, багатопоточні, розподілені, вбудовані, паралельні, реактивні, реального часу, їх різновиди – синхронні, асинхронні, орієнтовані на події.

Послідовна система (Sequential system) – це система, яка описується послідовністю елементарних кроків обчислення (потокот виконання) деякого обчислювача (ЕОМ, машина Тюрінга, тощо). Для адекватного тлумачення семантики послідовних систем на різних рівнях абстракції побудована ціла низка відповідних програмних алгебр та композиційно-номінативних логік [5,7].

Багатопоточна система (Concurrent system) – система, яка складається з кількох потоків виконання, що виконуються одночасно та можуть взаємодіяти між собою за допомогою різних механізмів (пересилка повідомлень, спільна пам'ять, канали).

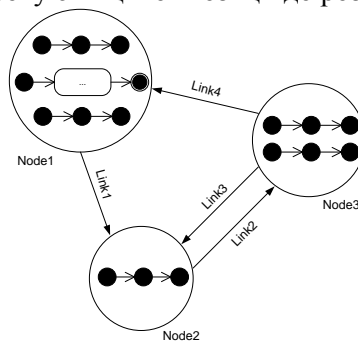
Паралельна система (Parallel system) – система у якій потоки виконуються одночасно та їх взаємодія відбувається через спільну пам'ять. Паралельні системи – це підтип багатопоточної.

Для опису багатопоточних систем та їх підтипів створено велика кількість різноманітних підходів, серед них CSS, CSP, LOTOS, TLA, LTL, CTL (та інші темпоральні логіки), мережі Петрі, тощо. У роботі такі системи розглядаються як транзиційні системи спеціального типу. Тобто вони задаються простором станів та відношенням переходу між ними. Множина станів

подається як номінативна множина, а відношення переходу – як часткова багатозначна функція над станами. При такому тлумаченні прийнято розрізняти зовнішню та внутрішню поведінку систем. Під зовнішньою поведінкою розуміють послідовність станів у яких перебуває система під час роботи та переходи між станами. Коли йдеться мова про зовнішню поведінку вважається, що перехід з одного стану в інший відбувається атомарно. Внутрішню поведінку системи не можливо безпосередньо спостерігати – це те, що при фіксації рівня деталізації не приймається до розгляду. У залежності від рівня абстрактності, системи можуть не відрізнятися зовнішньою поведінкою, а внутрішньою відрізнитись суттєво.

Розподілена система (Distributed system) – це довільний направлений граф у якого вершинам відповідають багатопоточні системи, а ребрам – односторонні канали передачі даних між ними. Вершини цього графу – фізично розділені обчислювачі. Важливо, що кожна вершина розподіленої системи діє у власному адресному просторі.

У мовах специфікації та моделювання багатопоточних систем (LOTOS, [1]) передбачені спеціальні композиції: \parallel (незалежне паралельне виконання), \parallel (паралельне виконання з синхронізацією), $[\]$ (вибір). Застосовуючи ці композиції до розмічених

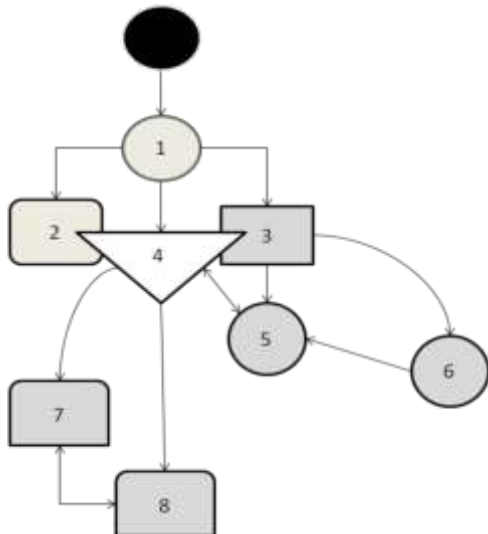


(мал. Схема розподіленої системи)

Відмінність розподіленої системи від багатопоточної проявляється у тому, що її стан – це диз'юнктивна множина, утворена із станів її підсистем. Важливо, що розподілені системи мають композиційну структуру – вони утворені з інших систем, між якими існують певні зв'язки.

Реактивна система (Reactive system) – це система, яка працює із взаємодією (запит-відповідь) з певним середовищем (оточенням) та її виконання продовжується нескінченно довго. До реактивних систем відносяться вбудовані системи, системи реального часу.

Вбудована система (Embedded system) – це комп'ютерна система, яка складається з кількох взаємодіючих між собою компонентів, що є частиною більшої системи (середовища) та виконує для неї деякі функції. Середовищем для вбудованої системи, як правило, служить деякий пристрій (літак, машина, тощо).



- 6. Розподілені системи
- 7. Вбудовані системи

Системи, що взаємодіють з певним середовищем можна уявляти як обчислювачі з певним оракулом. В даному випадку в якості оракула виступає середовище.

Система реального часу (Real-time system) – це реактивна система, яка повинна генерувати відповідь на запит у встановлений проміжок часу.

Схема, яка показує основні зв'язки між різними типами систем, приводиться на наступному малюнку.

- 1. Закриті системи
- 2. Послідовні системи
- 3. Багатопоточні системи
- 4. Реактивні системи
- 5. Паралельні системи

8. Системи реального часу

Зв'язки показують перетин типів систем. Наприклад, система може бути одночасно вбудованою та реального часу (зв'язок 7-8)

Таким чином, у роботі проаналізовані існуючі типи архітектур програмних систем, досліджені їх взаємозалежності, та класифіковані у відповідності до їх композиційної структури.

Для кожного з наведених типів систем існують засоби їх формального моделювання та дослідження. У таблиці наводяться деякі основні властивості (типи систем до яких вони застосовуються) найбільш розповсюджених методів:

Method \ System Type	Concurrent systems	Distributed systems	Embedded systems	Parallel systems	Reactive systems	Real-time systems	Safety-Critical systems	Sequential systems	Synchronous systems
CCS									
CSP									
LOTOS									
Мережі Петрі									
RAISE									
TLA									
VeriSoft									
SPIN									
UNITY									
HyTech									
Esterel									
LUSTRE									
TTM/RTTL									
ACSR									
UPPAAL									
NP-Tools									
VDM									
Z-Notation									
B-Method									
PVS									

ЛІТЕРАТУРА:

1. Howard Bowman, Rodolfo Gomez. "Concurrency Theory. Calculi and Automata for Modelling Untimed and Timed Concurrent Systems" // Springer. – 2006, 444 p. ISBN 1-85233-895-3.
2. Denis Caromel, Ludovic Henrio. "A Theory of Distributed Objects" // Springer. – 2005, 352 p. ISBN 3-540-20866-6.
3. Peter Van Roy, Seif Haridi, "Concepts, Techniques, and Models of Computer Programming" // The MIT Press. – 2004, 931 p. ISBN 0-262 -22069-5
4. Nikitchenko, N.S.: Composition nominative approach to the explication of the notion of program, *Problems of Programming*, No 1, 1999, pp. 16-31.
5. И.А. Басараб, Н.С. Никитченко, В.Н. Редько, «Композиционные базы данных» // «Либідь». – Київ, 1992.
6. E.M. Clarke, E.A. Emerson, A.P. Sistla, "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications" // ACM Transactions on Programming Languages and Systems, Vol.8, No. 2, April 1986, Pages 244-263.
7. Нікітченко М.С. «Інтенціонально-орієнтований підхід до теорії програмування» // Вибрані питання програмології. Праці наукового семінару «Програмологія та її застосування», Київ. – 2007, стор. 22-54