УДК 51.681.3

А.С. Заболотна

Метод трансляції *SDL*-специфікацій за допомогою мереж Петрі високого рівня

Рассмотрены *SDL*-спецификации распределенных систем с динамическим порождением и уничтожением экземпляров процессов. Для них предложен метод трансляции и модифицированные раскрашенные сети Петри, в которых используется концепция интервального времени.

SDL-specifications of the distributed systems with dynamic generation end the removing of the examples of the process are considered. A method of the translation and the modified coloured Petri nets in with the concepts of the interval time is used are suggested for them.

Розглянуто *SDL*-специфікації розподілених систем з динамічним породженням та видаленням примірників процесів. Для них запропоновано метод трансляції в модифіковані кольорові мережі Петрі, в яких використовується концепція інтервального часу.

Вступ. В останнє десятиліття прогрес засобів передачі та обробки інформації призвів до стрімкого проникнення телекомунікаційних технологій в усі сфери життя, завдяки чому побудова та верифікація комунікаційних протоколів стала однією з областей формальної верифікації, що найбільш активно розвиваються. Великі обсяги і високий рівень складності розроблених протоколів потребували створення засобів автоматичної або напівавтоматичної верифікації розподілених асинхронних систем, в тому числі і комунікаційних протоколів [1, 2].

Для представлення розподілених систем часто використовується мова виконуваних специфікацій *SDL* (*Specification and Description Language*) [1, 2], прийнята як стандарт *ITU*. Перевага *SDL* в її виразності, проте, саме це і ускладнює аналіз та верифікацію специфікацій цих систем. Один з підходів до досліджень специфікацій *SDL* полягає в автоматичному переведенні специфікацій розподілених систем у моделі, для яких розроблено ефективні методи аналізу. Як моделі вибрано модифіковані кольорові мережі Петрі (КМП), названі ієрархічними часовими типізованими мережами (ІЧТ-мережами) [3]. Вони розширюють безпечні кольорові мережі Петрі за допомогою понять часу (семантика Мерліна), пріоритетів, а також спеціальних місць, що зображують черги фішок.

Коротка характеристика та основні властивості *SDL*

SDL – мову специфікацій та опису розроблено у кінці 70-х років XX століття Міжнародним консультативним комітетом з телеграфії та телефонії (*CCITT*) і призначено для опису структури та функціонування систем у реальному часі, зокрема, мереж зв'язку. *SDL* побудовано на базі моделі скінченного автомата за об'єктно-орієнтованою схемою. Мова має як статичні, так і динамічні конструкції, які описують породження та знищення примірника процесу [2].

Графічна мова опису та специфікацій *SDL* є однією з найбільш відомих, що використовуються для формального опису поведінки реактивних та розподілених систем. Спочатку вона була спрямована на телекомунікаційні системи, а тепер широко застосовується і для опису систем керування процесами, і систем реального часу.

Мова *SDL* містить опис як структурної, так і функціональної частини розроблюваної системи. *SDL*-модель слід розуміти як набір сполучених узагальнених абстрактних скінченних автоматів.

Архітектура *SDL*-моделі є багаторівневою. Найбільший загальний об'єкт (найвищий рівень) називається *системою* (*system*), все, що знаходиться поза системою, називається *оточенням* (зовнішнім середовищем, *environment*) системи і засобами *SDL* не описується. Система взаємодіє зі своїм оточенням, отримуючи від нього і посилаючи йому сигнали. Система сама по собі є не функціональним, а структурним описом моделі.

Коротко структуру системи можна описати так. Вона складається з одного або декількох блоків (*block*), з'єднаних між собою і з навколишнім середовищем каналами, по яких передаються сигнали. У свою чергу, блок є іншим, більш низьким, ніж система, рівнем *SDL*-моде-

лі. На даному рівні систему або зовнішній блок (у разі вкладеності блоків) можна розглядати як середовище, поведінка якого спостерігається тільки через вхідні сигнали для поточного блоку. Останній також містить процеси (*process*), що є вже функціональними компонентами системи в тому сенсі, що саме вони визначають поведінку системи.

Процеси являють собою нижній рівень *SDL*-моделі. Структурно процес завжди знаходиться в блоці, який розглядається як середовище для даного процесу. Сигнали між процесами, процесами і блоками передаються маршрутами (*signalroute*). Маршрути можуть бути з'єднані з каналами верхнього рівня (*connect*). Отже, процес – це об'єкт, який має скінченне число станів, переходи і черги вхідних сигналів. Перебуваючи в деякому стані, процес витягує черговий сигнал і виконує перехід – здійснює ряд дій, зокрема, визначає свій наступний стан.

Під час роботи системи, описаної в *SDL*, можуть створюватися і знищуватися примірники процесів. Для кожного процесу може створюватися кілька примірників або жодного. Для простоти примірники процесів далі називатимемо процесами [2].

Початковий стан процесу описується конструкцією *start*, з якої здійснюється початковий перехід в будь-який стан процесу (*state*). Після цього процес починає функціонувати – обробляти сигнали з вхідної черги (згідно *FIFO*) і здійснювати переходи в інші стани.

Кожен перехід складається з послідовності дій *SDL*-програми – присвоєння, установка і скидання таймера, умовний оператор, створення нового і зупинка поточного процесу, посилка сигналу і перехід в інший стан. Процес може містити оголошення своїх локальних змінних і використовувати значення локальних змінних інших процесів (*export*, *import*), а також викликати процедури, які також мають свої стани, переходи та локальні змінні. За необхідності обробити сигнал, процедура використовує вхідну чергу процесу, яким вона була викликана, а посилання сигналу процедурою інтерпретується як посилання цього сигналу цим процесом. Різні процеси можуть звертатися до одних і тих же процедур.

Процеси *SDL*-системи працюють асинхронно, спілкуючись між собою і середовищем за допомогою сигналів.

Опишемо підхід до верифікації *SDL*-специфікацій.

Метод трансляції *SDL*-специфікацій в мережі Петрі

Алгоритм трансляції *SDL*-специфікацій в мережеві моделі системи *SDLE* (*SDL Editor*) реалізовано методом двопрохідної трансляції [3, 4].

Мережева модель створюється за допомогою поетапного уточнення. На першому етапі будується мережа, яка розташовується на першій сторінці, відповідає основній структурі системи і містить по одному переходу для кожного блоку. Кожний канал, пов'язаний з блоком, представляється у мережі одним або двома місцями – чергами, залежно від того, був він одно- або двонаправленим. Фішки в отриманих місцях можуть приймати значення з безлічі кольорів, які визначаються сигналами, що передаються по відповідних каналах. Спочатку всі місця, породжені за описом каналів, мають нульову розмітку. З'єднання переходів та місць здійснюється дугами, напрямок яких збігається з напрямком передачі повідомлень [5, 6].

На другому етапі здійснюється трансляція блоку, яка відбувається так само, як і трансляція системи в цілому. Переходи, побудовані на першому етапі, замінюються підмережами, які відповідають розбиттю блоку і розташовуються на пов'язаній з цим переходом підсторінці. При трансляції блоку, що складається з підблоків і внутрішніх каналів, кожному підблоку в підмережі відповідає один перехід, кожному внутрішньому каналу – одне або два місця-черги, залежно від того, одно- чи двонаправлений канал. Трансляція блоку, що складається з процесів, відбувається аналогічно трансляції блоку будь-якого рівня ієрархії. Кожному примірнику процесу відповідає один перехід, кожному маршруту – одне або два місця-черги, залежно від того, який маршрут – одно- або двонаправлений [2, 4].

Транслятор функціонує так. Модуль аналізатора обробляє текстовий файл, що містить *SDL*-специфікацію, тобто здійснює лексичну згорт-

ку та синтаксичний розбір і будує внутрішнє представлення специфікації. У разі відсутності помилок запускається модуль генерації мережевої моделі. Спочатку будується внутрішнє представлення ієрархічної мережевої моделі (ІЧТ)-мережі, а потім здійснюється її візуалізація в системі *SDLE*.

ІЧТ-мережа – це композиція безлічі неієрархічних мереж, званих сторінками. Сторінки можуть містити вершини спеціального типу, які називаються *модулями* і з'єднуються з місцями на сторінці за тим же принципом, що і переходи.

Модуль представляється підмережею, яка розташовується на окремій сторінці і в свою чергу може містити модулі. Така сторінка називається *підсторінкою* сторінки, на якій розташовується модуль. Підсторінка містить копії всіх місць, з якими пов'язаний модуль. Місце–копія може бути вхідним місцем для деякого переходу або модуля на підсторінці тоді і тільки тоді, коли його прототип є вхідним місцем для модуля, що представляє підсторінку. Аналогічно тільки копія вихідного місця-прототипу може бути вихідним місцем деякого переходу або модуля на підсторінці.

Поведінка ієрархічної мережі визначається поведінкою еквівалентної їй неієрархічної мережі, яку отримано в результаті заміщення всіх модулів сторінками, які вони представляють. При цьому кожен модуль разом зі своїми дугами видаляється зі сторінки, а на його місце заноситься підмережа, що розташовувалася на підсторінці. З'єднання мереж відбувається за місцями: кожне місце-прототип склеюється з усіма своїми копіями. Побудова внутрішнього представлення мережі проводиться за кроками, що відповідають етапам в описі алгоритму трансляції. В першу чергу створюється кореневий рівень ієрархічної мережі, що складається з модулів системи *SDLE*, що представляють блоки *SDL*-специфікації. Також на цьому етапі створюються місця-черги, які моделюють канали. Ці модулі та місця з'єднуються дугами згідно з описом *SDL*-системи. Наступні чотири кроки генерації виконуються послідовно для кожного блоку.

На другому проході алгоритм генерує мережу, яка реалізує блок *SDL*-специфікації. Ця мережа, у свою чергу, містить модулі, що відповідають процесам і місцям-чергам, що моделюють маршрути сигналів. Ці модулі та місця з'єднуються дугами згідно з описом блоку.

Далі генерується мережа, що реалізує процес *SDL*-специфікації. Вона містить модулі, що відповідають *SDL*-переходам процесу, місцям, що моделюють змінні, та лічильники, і деяким службовим місцям. На цьому етапі побудови мережі дуги не створюються, а добудовуються на наступному кроці, оскільки не можна заздалегідь вказати, на яких переходах використовується змінна або таймер [7].

На наступному кроці трансляції *SDL*-переходів створюється підмережа, в якій реалізується логіка *SDL*-переходу. В процесі побудови цієї мережі створюються дуги для мережі третього рівня. На завершальному етапі будуються модулі, що реалізують процедури і дії з таймерами, якщо такі є.

При побудові мережі в системі *SDLE* використовуються засоби створення ієрархічних мережевих моделей, що надаються цією системою. Після того як мережа створена, здійснюється розміщення її елементів на площині, тобто кожному елементу приписуються координати на відповідних сторінках системи *SDLE*. При цьому фрагменти мережі, які реалізують дії *SDL*-переходів, розміщуються типовим способом [8].

Приклад. Моделювання системи та блоку

Розглянемо специфікацію системи *S*, текстовий опис якої наведено нижче, графічне зображення – на рис. 1:

```
system S;
signal s1, s2, s3(Integer), s4, s5, s6;
channel C1
    from B1 to env with s1, s2;
endchannel C1;
    channel C2
    from B1 to B2 with s4;
    from B2 to B1 with s5, s6;
endchannel C2;
channel C3
    from env to B2 with s3;
    from B2 to env with s1;
endchannel C3;
block B1 referenced;
block B2 referenced;
endsystem S;
```

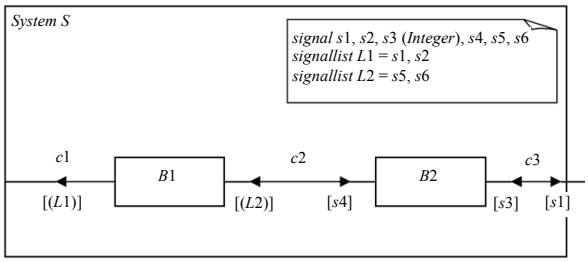

Рис. 1. Опис системи *S*

Описано три канали – *c*1, *c*2, *c*3 і два блоки – *B*1 і *B*2. По каналу *c*1, що з'єднує зовнішнє середовище (*environment*) з блоком *B*1, від блоку можуть передаватися сигнали *s*1, *s*2 до зовнішнього середовища (*environment*). По двонаправленому каналу *c*3, що з'єднує блок *B*2 з оточенням, від блоку може бути передано сигнал *s*1, а від зовнішнього середовища блок може отримати сигнал *s*3. Між блоками *B*1 і *B*2 є двонаправлений канал *c*2. По ньому від блоку *B*1 до блоку *B*2 може передаватись сигнал *s*4, який має параметр цілого сорту, а в зворотному напрямку – сигнали *s*5, *s*6. Блок *B*1 має підструктуру *B*1, опис якої наведено на рис. 3. Кожному підблоку підструктури *B*1 в мережі відповідає модуль, ім'я якого збігається з ім'ям підблоку. Блок *B*2 в системі не описано.

При створенні мережі відповідної *SDL*-системи використовуються видимі для всіх блоків опису списки сигналів, а також каналів, що з'єднують між собою блоки і навколишнє середовище, та частина інформації з опису процесів. Мережа, що будується, містить по одному модулю на кожний блок *SDL*-системи. Для більшої наочності як ім'я модуля можна використовувати ім'я відповідного блоку. На цьому ж етапі відображуються канали. Кожен канал в *SDL* має асоційовану з ним *FIFO*-чергу, в якій зберігаються сигнали, отримані через цей канал. Цю ж властивість мають і маршрути.

При трансляції *SDL*-системи канали, що не піддаються розбиттю в процесі подальшої деталізації системи, а також маршрути природно представляти місцями спеціального типу – чергами, а сигнали, які зберігаються в чергах, – фішками. Крім того, кожна фішка має містити інформацію про те, який процес її відправив і якому процесу вона призначається.

При початковій розмітці всі місця, відповідні каналам, порожні. Однонаправлений канал в мережі представляється одним місцем. Вхідний канал відображається місцем, вхідним для модуля, що представляє блок, пов'язаний з цим каналом, вихідний канал – вихідним місцем. Двонаправлений канал представляється двома місцями.

Декларації мережі при моделюванні системи *S* (див. рис. 1) доповнюються наступними множинами кольорів:

*color Sig*1 = *with s*1 | *s*2
*color Sig*22 = *with s*5 | *s*6
color Sig = *with s*3
*color Sig*22 = *product Sig* * *Int*
*color Sig*31 = *with s*1
*color Sig*32 = *with s*4
*color C*_1 = *product integer* * *integer* * *Sig*1
*color C*_21 = *product integer* * *integer* * *Sig*21
*color C*_22 = *product integer* * *integer* * *Sig*22
*color C*_31 = *product integer* * *integer* * *Sig*31
*color C*_32 = *product integer* * *integer* * *Sig*32.

Перше поле будь-якої фішки, що приймає значення з множини кольорів *C*_1, *C*_21, *C*_22, *C*_31, *C*_32, отриманого при відображенні опису сигналів, містить особистий ідентифікатор примірника-одержувача, друге – власний ідентифікатор примірника-відправника. Спочатку всі місця, породжені за описами каналів, мають нульову розмітку.

В процесі подальшої побудови мережі декларації доповнюються змінними, які входять у вирази на дугах мережі і в спускові функції переходів, і, можливо, новими множинами кольорів, якщо блоки і процеси містять власні визначення сортів, сигналів і списків сигналів. Мережу для системи *S* наведено на рис. 2 (декларації мережі не позначено). Для більшої наочності як ім'я модуля використовується ім'я відповідного блоку.

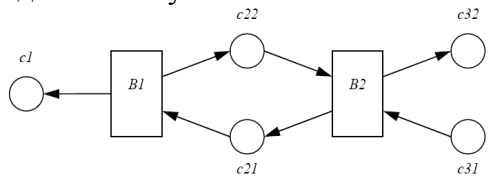

Рис. 2. Мережа для системи *S*

Отже, після завершення першого кроку моделювання маємо мережу, яка складається з модулів (представляються «чорними скриньками»),

що відповідають блокам в системі, і місць, отриманих при відображенні каналів.

Опис системи – це послідовність діаграм, де кожна наступна діаграма здійснює подальшу деталізацію системи. Передбачено діаграми підструктур блоків і каналів. Діаграма підструктури блоку використовується тоді, коли розглянутий блок є складним об'єктом і складається з підблоків і внутрішніх каналів. Канали усередині системи, які з'єднують блоки між собою і з оточенням, назвемо *зовнішніми*. В результаті розбиття блоків виникає структура, аналогічна структурі системи, в якій рамка блоку виконує роль рамки системи.

На кожній сторінці, пов'язаній з модулем, відповідним деякому блоку, відображається внутрішня структура цього блоку. Це відображення здійснюється таким же способом, що і відображення структури системи на першій сторінці. Кожному підблоку на підсторінці відповідає один модуль, кожному внутрішньому каналу – одне або два місця, в залежності від того, який канал діє – одно- або двонаправлений.

Розглянемо блок $B1$, що має підструктуру $B1$, опис якої наведено на рис. 3. Кожному підблоку підструктури $B1$ відповідає модуль, ім'я якого збігається з іменем підблоку.

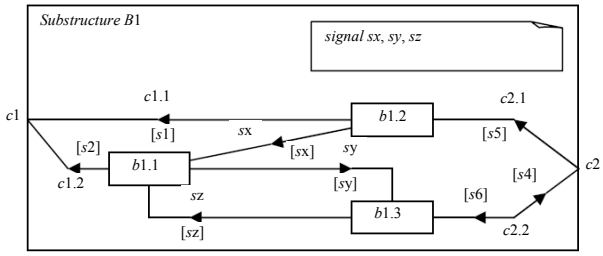

Рис. 3. Опис підструктури $B1$

На рис. 4 показано, як відображуються в мережі внутрішні канали підструктури $B1$. Місце $c1.1$ відповідає внутрішньому каналу $c1.1$ і є вихідним для модуля $b1.2$, місце $c1.2$ відповідає внутрішньому каналу $c1.2$ і є вихідним для модуля $b1.1$, місце $c2.1$ відповідає внутрішньому каналу $c2.1$ і є вхідним для модуля $b1.2$. Каналу $c2.2$ в мережі відповідає два місця: $c2.21$ – вхідне місце і $c2.22$ – вихідне для модуля $b1.3$, що моделює підблок $b1.3$. Нові односпрямовані канали cx, cy і cz в мережі моделюються місцями з такими ж іменами відповідно.

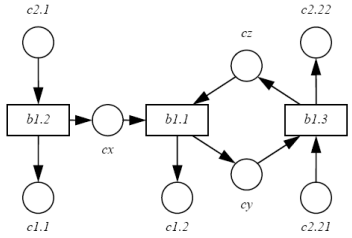

Рис. 4. Мережеве представлення підструктури $B1$

Кожен сигнал, що надійшов на зовнішній канал, може передаватися тільки на один внутрішній канал. За правилами побудови ієрархічної мережі копія кожного з місць, що представляє зовнішній канал, з'являється на пов'язаній з модулем сторінці, яка в свою чергу містить по одному модулю для кожного опису внутрішнього підблоку або процесу. У мережі, що будується, місця, створені на попередньому етапі і відповідні зовнішньому каналу, зливаються з місцями, відповідними внутрішнім каналам, приєднаним до зовнішнього у такий спосіб. Назвемо місця, створені на попередньому етапі і відповідні зовнішньому каналу, *старими* місцями, а місця, створені на етапі відображення підструктури і відповідні внутрішнім каналам – *новими*.

Отже, кожному *старому* місцю, що є вхідним для деякого модуля, що представляє блок, відповідає власний набір *нових* місць, які є вхідними для модулів, що представляють підблоки даного блоку. Аналогічно, кожному *старому* місцю, що є вихідним місцем деякого модуля, відповідає власний набір *нових* місць, які є вихідними для модулів, що представляють підблоки блоку.

У даному прикладі для *старого* місця $c1$, що є вихідним для модуля $B1$, відповідними *новими* місцями будуть $c1.1$ і $c1.2$, які будуть вихідними для модулів $b1.2$ і $b1.1$ відповідно.

Кожне *старе* місце, що є вхідним, моделює блок, зливається з відповідними йому *новими* місцями, які є вхідними для модулів, моделюючих підблоки цього блоку. Аналогічно відбувається злиття *старого* місця – вихідного для модуля, що моделює блок, з відповідними йому *новими* місцями – вихідними для модулів, які моделюють підблоки цього блоку. Зауважимо, що кілька місць, які відповідають внутрішнім

каналам, можуть бути злиті з одним місцем, відповідним зовнішньому каналу.

На рис. 5 показано сторінку для підструктури *B*1, в якій місця, що відповідають внутрішнім каналам, вже злиті з місцями, відповідними зовнішнім каналам. Місце *c*1 злито з місцями *c*1.1 і *c*1.2, місце *c*21 – з місцями *c*2.1 і *c*2.21, а місце *c*22 – з місцем *c*2.22. Надалі «злитим» місцям будемо приписувати ті ж імена, що мають *стари* місця на сторінці, що моделює блок. Наприклад, замість імені *c*22 & *c*2.22 використовуватиметься ім'я *c*22.

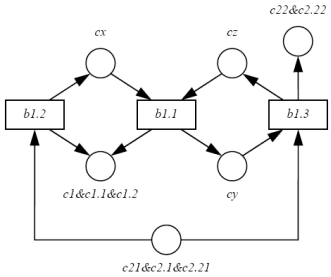

Рис. 5. Кінцева мережа для підструктури *B*1

Якщо опис будь-якого блоку дано за межами опису системи (зазначене посилання на відсутність опису), то подальшого уточнення мережі для цього блоку не відбувається. В кінцевій мережі даному блоку відповідатиме один модуль, який було породжено на попередньому етапі побудови. При моделюванні підструктури каналу на сторінці, пов'язаній з додатковим модулем, відображується підструктура цього каналу аналогічно тому, як відображалася підструктура блоку.

Кожен блок, не розчленований на підблоки, має містити хоча б один процес. Процеси між собою і з рамкою блоку з'єднуються маршрутами. Моделювання блоку, що складається з процесів, аналогічне моделюванню блоку будь-якого рівня ієрархії, при цьому кожному опису процесу зіставляється один модуль. Відмінність полягає в моделюванні точки приєднання маршрутів до каналу. Сигнал, що надходить до блоку вхідним каналом, може передаватися по декількох приєднаних до нього маршрутах.

В описі семантики *SDL* точно не визначається, в які маршрути передаватиметься сигнал. Кожного разу множина маршрутів, які можуть передати цей сигнал, і множина примірників процесу, які можуть отримати сигнал, вибирається довільно.

Система *SDLE*

Ця система являє собою інтегрований програмний комплекс для проектування, аналізу та симуляції моделей ІЧТ-мереж, що складається з транслятора з мови *SDL* та симулятора – блоку імітаційного моделювання.

Цикл роботи користувача в системі *SDLE* має такий вигляд. Будується вихідна мережева модель досліджуваної системи, або ця мережева модель може бути отримана яким-небудь іншим способом, наприклад, як результат трансляції з мови здійсненних специфікацій *SDL*. Потім проводиться симуляція моделі в автоматичному режимі. Система дозволяє візуально контролювати хід симуляції, відстежувати аналіз структурних властивостей поетапно. За результатами симуляції вихідна модель уточнюється, і цикл розробки повторюється до отримання задовільних результатів [9].

Симулятор дозволяє простежити за процесом функціонування мережі при заданій початковій розмітці. При цьому часто вдається виявити семантичні помилки моделі, які проявляються у неочікуваній поведінці мережі.

Висновок. У статті описано процедуру трансляції *SDL*-специфікацій з динамічними конструкціями в кольорові мережі Йенсена, збагачені пріоритетами. Розширення цих мереж пріоритетами дозволяє розвинути засоби симуляції та аналізу.

Спосіб моделювання заснований на тому, що в багаторівневому описі системи в *SDL* позиція кожного примірника процесу в загальній ієрархії системи залишається незмінною, що дозволяє опис системи транслювати в структуру мережі, а примірники процесу моделювати за допомогою фішок.

У результаті роботи алгоритму створюється така мережева модель, в якій в кожному місці міститиметься не більше однієї фішки, що моделює деякий примірник процесу. Тож, якщо під час функціонування системи може існувати *n* різних примірників будь-якого процесу, то в кожному місці мережі, яка його моделює, може міститися не більше *n* фішок, причому ко-

жна з них відповідатиме своєму примірнику процесу. Це дозволяє істотно підвищити ефективність моделювання, оскільки істотно зменшується перебір варіантів зв'язування змінних [9].

1. *Specification* and Description Language (SDL). Recommendation Z.100. – ITU-T, 2000.
2. *Карабегов А.В.*, *Тер-Микаэлян Т.М.* Введение в язык *SDL*. – М.: Радио и связь, 1993. – 272 с.
3. *Jensen K.*, *Christensen S.*, *Wells L.* Colored Petri Nets and CPN Tools for Modeling and Validation of Concurrent Systems // Intern. J. on Software Tools for Technology Transfer. – 2007. – **9**. – P. 213–254.
4. *Peterson J.L.* Petri Net Theory and the Modeling of Systems. Englewood Cliffs. – New Jersey: Prentice-Hall, Inc., 1981. – 290 p.
5. *Jensen K.* Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use. – Berlin: Springer-Verlag, 1997. – **1–3**. – P. 343–416.
6. *Murata T.* Petri Nets: Properties, Analysis and Applications // Proc. of the IEEE. – 1989. – **77**, № 4. – P. 541–580.
7. *Котов В.Е.* Сети Петри. – М.: Наука, 1984. – 157 с.
8. *Кривий С.Л.* Дискретна математика: Вибр. питання: Навч. посібник для студ. вищ. навч. закл. – К.: КМА, 2007. – 572 с.
9. *Заболотна А.С.* Моделювання динамічних конструкцій *SDL*-специфікацій за допомогою мереж Петрі. «Шевченківська весна 2011». Київ, 21–25 березня 2011.

Поступила 29.07.2011

Тел. для справок: (067) 895-7426 (Киев)
E-mail: *ZabolotnaA@gmail.com*

●

А.С. Заболотная

Метод трансляции *SDL*-спецификаций с помощью сетей Петри высокого уровня

Введение. В последнее десятилетие прогресс средств передачи и обработки информации вызвал стремительное проникновение телекоммуникационных технологий во все сферы жизни, благодаря чему построение и верификация коммуникационных протоколов стала одной из наиболее активно развивающихся областей формальной верификации. Большие объемы и высокая степень сложности разработанных протоколов нуждались в создании средств автоматической или полуавтоматической верификации распределенных асинхронных систем, в том числе и коммуникационных протоколов [1, 2].

Для представления распределенных систем часто используется язык исполняемых спецификаций *SDL* (*Specification and Description Language*) [1, 2], принятый в качестве стандарта *ITU*. Преимущество *SDL* – в его выразительности, однако, именно она и затрудняет анализ и верификацию спецификаций этих систем. Один из подходов к исследованию спецификаций *SDL* заключается в автоматическом переводе спецификаций распределенных систем в модели, для которых разработаны эффективные методы анализа. В качестве моделей выбраны модифицированные раскрашенные сети Петри (РСП), названные иерархическими временными типизированными сетями (ИВТ-сетями) [3]. Последние расширяют безопасные раскрашенные сети Петри с помощью понятий времени (семантика Мерлина), приоритетов, а также специальных мест, изображающих очереди фишек.

Краткая характеристика и основные свойства *SDL*

SDL– язык спецификаций и описания, разработанный в конце 70-х годов XX века Международным консультативным комитетом по телеграфии и телефонии (*CCITT*) и предназначен для описания структуры и функционирования систем в реальном времени, в частности, сетей связи. Язык построен на базе модели конечного автомата по объектно-ориентированной схеме, имеет как статические, так и динамические конструкции, описывающие порождения и уничтожение экземпляра процесса [2].

Графический язык описания и спецификаций *SDL* – один из наиболее известных, используемый для формального описания поведения реактивных и распределенных систем. Сначала он был нацелен на телекоммуникационные системы, однако теперь широко применяется и для описания систем управления процессами и систем реального времени.

Язык *SDL* включает в себя описание как структурной, так и функциональной части разрабатываемой системы. *SDL*-модель понимают как набор соединенных обобщенных абстрактных конечных автоматов.

Архитектура *SDL*-модели многоуровневая. Самый общий объект (наивысший уровень) называется *системой* (*system*), все, что находится вне системы, называется *окружением* (внешней средой, *environment*) системы и средствами *SDL* не описывается. Система взаимодействует со своим окружением, получая от него и посылая ему сигналы. Система сама по себе не функциональное, а структурное описание модели.

Коротко структуру системы можно описать так. Она состоит из одного или нескольких блоков (*block*), соединенных между собой и с окружающей средой каналами, по которым передаются сигналы. В свою очередь блок – это другой, более низкий, чем система, уровень *SDL*-модели. На данном уровне система или внешний блок (в случае вложенности блоков) может рассматриваться как среда, поведение которой наблюдается только через входные сигналы для текущего блока. Блок также содержит процессы (*process*), которые представляют собой уже функциональные компоненты системы в том смысле, что именно они определяют поведение системы.

Процессы представляют собой нижний уровень *SDL*-модели. Структурно процесс всегда находится в блоке, который рассматривается как среда для данного процесса. Сигналы между процессами, процессами и блоками передаются маршрутам (*signalroute*). Маршруты могут быть соединены с каналами верхнего уровня (*connect*). Итак, процесс – это объект с конечным числом состояний, переходов и очередью входных сигналов. Находясь в некотором состоянии, процесс извлекает очередной сигнал и выполняет переход – осуществляет ряд действий, в частности, определяет свое следующее состояние.

Во время работы системы, описанной в *SDL*, могут создаваться и уничтожаться экземпляры процессов. Для каждого процесса может создаваться несколько экземпляров или ни одного. Для простоты экземпляры процессов далее будем называть *процессами* [2].

Исходное состояние процесса описывается конструкцией *start*, с которой осуществляется первоначальный переход в любое состояние процесса (*state*). После этого процесс начинает функционировать – обрабатывать сигналы с входящих заявок (согласно *FIFO*) и осуществлять переходы в другие состояния.

Каждый переход состоит из последовательности действий *SDL*-программы – присвоение, установка и сброс таймера, условный оператор, создание нового и остановка текущего процесса, посылка сигнала и переход в другое состояние. Процесс может содержать объявления своих локальных переменных и использовать значения локальных переменных других процессов (*export*, *import*), а также вызывать процедуры, которые также имеют свои состояния, переходы и локальные переменные. При необходимости обработать сигнал, процедура использует входную очередь процесса, которым она была вызвана, а ссылки сигнала процедуре интерпретируются как ссылка этого сигнала этим процессом. Различные процессы могут обращаться к одним и тем же процедурам.

Процессы *SDL*-системы работают асинхронно, общаясь между собой и средой с помощью сигналов.

Опишем подход к верификации *SDL*-спецификаций.

Метод трансляции *SDL*-спецификаций в сети Петри

Алгоритм трансляции *SDL*-спецификаций в сетевые модели системы *SDLE* (*SDL Editor*) реализован методом двухпроходной трансляции [3, 4].

Сетевая модель создается с помощью поэтапного уточнения. На *первом* этапе строится сеть, которая располагается на первой странице, соответствует основной структуре системы и содержит по одному переходу для каждого блока. Каждый канал, связанный с блоком, представляется в сети одним или двумя местами – очередями, в зависимости от того, был он одно- или двунаправленным. Фишки в полученных местах могут принимать значения из множества цветов, определяемых сигналами, по соответствующим каналам. Сначала все места, порожденные по описаниям каналов, имеют нулевую разметку. Соединения переходов и мест осуществляются дугами, направление которых совпадает с направлением передачи сообщений [5, 6].

На *втором* этапе осуществляется трансляция блока, аналогичная трансляции всей системы в целом. Переходы, построенные на первом этапе, заменяются подсетями, соответствующими разбиению блока и располагаемыми на связанной с этим переходом подстранице. При трансляции блока, состоящего из подблоков и внутренних каналов, каждому подблоку в подсети соответствует один переход, каждому внутреннему каналу – одно или два места-очереди, в зависимости от того, одно- или двунаправленный канал. Трансляция блока, состоящая из процессов, происходит аналогично трансляции блока любого уровня иерархии. Каждому экземпляру процесса соответствует один переход, каждому маршруту – одно или два места-очереди, в зависимости от того, каков маршрут – одно- или двунаправленный [2, 4].

Транслятор функционирует так. Модуль анализатора обрабатывает текстовый файл, содержащий *SDL*-спецификации, т.е. осуществляет лексическую свертку и синтаксический разбор и строит внутреннее представление спецификации. За отсутствием ошибок запускается модуль генерации сетевой модели. Сначала строится внутреннее представление иерархической временной типизированной сетевой модели (ИВТ)-сети, а затем осуществляется ее визуализация в системе *SDLE*. ИВТ-сеть – это композиция множества неиерархических сетей, называемых страницами. Страницы, содержащие вершины специального типа, называются модулями и соединяются с местами на странице по тому же принципу, что и переходы.

Модуль представляет собой подсеть, располагаемую на отдельной странице, которая в свою очередь может содержать модули. Такая страница называется подстраницей страницы, на которой располагается модуль. Подстраница содержит копии всех мест, с которыми связан модуль. Место-копия может быть входным местом для некоторого перехода или модуля на подстранице тогда и только тогда, когда его прототип – входное место для модуля, представляющий подстраницу. Аналогично, только копия исходного места-прототипа может быть исходным местом некоторого перехода или модуля на подстранице.

Поведение иерархической сети определяется поведенчески эквивалентной ей неиерархической сетью, полученной в результате замещения всех модулей страницами, которые они представляют. При этом каждый модуль вместе со своими дугами удаляется со страницы, а на его место заносится подсеть, располагавшаяся на подстранице. Соединение сетей происходит по местам: каждое место-прототип склеивается со всеми своими копиями. Построение внутреннего представления сети осуществляется шагами, соответствующими этапам в описании алгоритма трансляции. В первую очередь создается корневой уровень иерархической сети, состоящей из модулей системы *SDLE*, представляющих блоки *SDL*-спецификации. Также на этом этапе создаются места-очереди, моделирующие каналы. Эти модули и места соединяются дугами согласно описанию *SDL*-системы.

Следующие четыре шага генерации выполняются последовательно для каждого блока.

На втором проходе алгоритм генерирует сеть, реализующую блок *SDL*-спецификации. Эта сеть в свою очередь содержит модули, соответствующие процессам, и местам-очередям, моделирующим маршруты сигналов. Эти модули и места соединяются дугами согласно описанию блока.

Далее генерируется сеть, реализующая процесс *SDL*-спецификации. Она содержит модули, соответствующие *SDL*-переходам процесса, местам, моделирующим переменные и счетчики, и некоторым служебным местам. На этом этапе построения сети дуги не создаются, а достраиваются на следующем шаге, поскольку нельзя заранее указать, на каких переходах используется переменная или таймер [7].

На следующем шаге трансляции *SDL*-переходов создается подсеть, в которой реализуется логика *SDL*-перехода. В процессе построения сети создаются дуги для сети третьего уровня. На завершающем этапе строятся модули, реализующие процедуры и действия с таймерами, если таковые имеются.

При построении сети в системе *SDLE* используются средства создания иерархических сетевых моделей, предоставляемых этой системой. После того как сеть создана, осуществляется размещение ее элементов на плоскости, т.е. каждому элементу приписываются координаты на соответствующих страницах системы *SDLE*. При этом фрагменты сети, реализующие действия *SDL*-переходов, размещаются типовым способом [8].

Пример. Моделирование системы и блока

Рассмотрим спецификацию системы *S*, текстовое описание которой приведено ниже, а графическое изображение – на рис. 1:

```
system S;
signal s1, s2, s3(Integer), s4, s5, s6;
channel C1
    from B1 to env with s1, s2;
endchannel C1;
    channel C2
    from B1 to B2 with s4;
    from B2 to B1 with s5, s6;
endchannel C2;
channel C3
    from env to B2 with s3;
    from B2 to env with s1;
endchannel C3;
block B1 referenced;
block B2 referenced;
endsystem S;
```

Здесь описаны три канала – *c*1, *c*2, *c*3 и два блока – *B*1 и *B*2. По каналу *c*1, соединяющему внешнюю среду (*environment*) с блоком *B*1, от блока могут передаваться сигналы *s*1, *s*2 к внешней среде (*environment*). По двунаправленному каналу *c*3, соединяющему блок *B*2 с окружением, от блока может быть передан сигнал *s*1, а от внешней среды блок может получить сигнал *s*3. Между блоками *B*1 и *B*2 есть двунаправленный канал *c*2. По нему от блока *B*1 в блок *B*2 может передаваться сигнал *s*4, имеющий параметр целого сорта, а в обратном направлении – сигналы *s*5, *s*6. Блок *B*1 имеет подструктуру *B*1, описание которой приведено на рис. 3. Каждому подблоку подструктуры *B*1 в сети соответствует модуль, имя которого совпадает с именем подблока. Блок *B*2 в системе не описан.

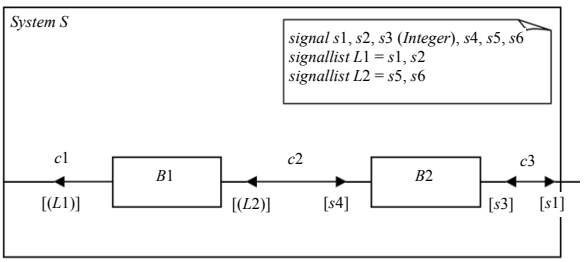

Рис. 1. Описание системы *S*

При создании сети соответствующей *SDL*-системы используются видимые для всех блоков описания списки сигналов, а также каналов, соединяющих между собой блоки и окружающую среду, и часть информации из описания процессов. Строящаяся сеть содержит по одному модулю на каждый блок *SDL*-системы. Для большей наглядности в качестве имени модуля можно использовать имя соответствующего блока. На этом же этапе отображаются каналы. Каждый канал в *SDL* имеет ассоциированную с ним *FIFO*-очередь, в которой хранятся сигналы, полученные через этот канал. Этим же свойством обладают и маршруты.

При трансляции *SDL*-системы, каналы, не поддающиеся разбивке в процессе дальнейшей детализации системы, а также маршруты естественно представлять местами специального типа – очередями, а сигналы, которые хранятся в очередях, – фишками. Кроме того, каждая фишка должна содержать информацию о том, какой процесс ее отправил и какому процессу она предназначается.

При начальной разметке все места, соответствующие каналам, пустые. Однонаправленный канал в сети представляется одним местом. Входной канал отображается местом, входным для модуля, представляющим блок, связанный с этим каналом, выходной канал – исходным местом. Двунаправленный канал представляется двумя местами.

Декларации сети при моделировании системы *S* (см. рис. 1) дополняются следующими множествами цветов:

```
color Sig1 = with s1 | s2
color Sig22 = with s5 | s6
color Sig = with s3
color Sig22 = product Sig * Int
color Sig31 = with s1
color Sig32 = with s4
color C_1 = product integer * integer * Sig1
color C_21 = product integer * integer * Sig21
color C_22 = product integer * integer * Sig22
```

*color C*_31 = *product integer * integer * Sig*31
*color C*_32 = *product integer * integer * Sig*32.

Первое поле любой фишки, принимающее значения из множества цветов *C*_1, *C*_21, *C*_22, *C*_31, *C*_32, полученных при отображении описания сигналов, содержащих личный идентификатор экземпляра-получателя, второе – личный идентификатор экземпляра-отправителя. Сначала все места, порожденные по описаниям каналов, имеют нулевую разметку.

В процессе дальнейшего построения сети декларации дополняются сменными, входящими в выражения на дугах сети и в спусковые функции переходов, и, возможно, новыми множествами цветов, если блоки и процессы содержат собственные определения сортов, сигналов и списков сигналов. Сеть для системы *S* приведена на рис. 2 (декларации сети опущены). Для большей наглядности в качестве имени модуля используется имя соответствующего блока.

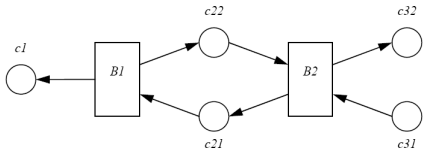

Рис. 2. Сеть для системы *S*

Таким образом, после завершения первого шага моделирования имеем сеть, состоящую из модулей (представляются «черными ящиками»), соответствующих блокам в системе, и мест, полученных при отражении каналов.

Описание системы – это последовательность диаграмм, где каждая следующая диаграмма осуществляет дальнейшую детализацию системы. Предусмотрены диаграммы подструктур блоков и каналов. Диаграмма подструктуры блока используется в тех случаях, когда рассматриваемый блок представляет собой сложный объект и состоит из подблоков и внутренних каналов. Каналы внутри системы, соединяющие блоки между собой и с окружением, назовем *внешними*. В результате разбиения блоков возникает структура, аналогичная структуре системы, в которой рамка блока выполняет роль рамки системы.

На каждой странице, связанной с модулем, соответствующим некоторому блоку, отображается внутренняя структура этого блока. Это отображение осуществляется таким же способом, что и отображение структуры системы на первой странице. Каждому подблоку на подстранице соответствует один модуль, каждому внутреннему каналу – одно или два места, в зависимости от того, какой канал функционирует – одно- или двунаправленный.

Рассмотрим блок *B*1, имеющий подструктуру *B*1, описание которой приведено на рис. 3. Каждому подблоку подструктуры *B*1 соответствует модуль, имя которого совпадает с именем подблока.

На рис. 4 показано, как отражаются в сети внутренние каналы подструктуры *B*1. Место *c*1.1 соответствует внутреннему каналу *c*1.1 и есть исходным для модуля *b*1.2, место *c*1.2 соответствует внутреннему каналу *c*1.2 и есть исходным для модуля *b*1.1, место *c*2.1 соответствует внутреннему каналу *c*2.1 и есть входным для модуля *b*1.2. Каналу *c*2.2 в сети соответствует два места: *c*2.21 – входное и *c*2.22 – исходное для модуля *b*1.3, моделирующего подблок *b*1.3. Новые однонаправленные каналы *cx, cy* и *cz* в сети моделируются местами с такими же именами соответственно.

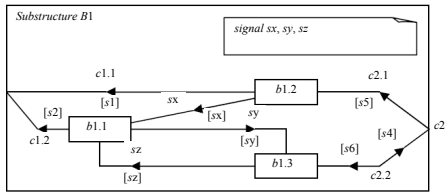

Рис. 3. Описание подструктуры *B*1

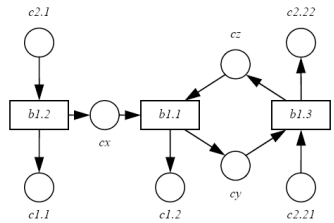

Рис. 4. Сетевое представление подструктуры *B*1

Каждый сигнал, поступивший по внешнему каналу, может передаваться только по одному внутреннему каналу. По правилам построения иерархической сети копия каждого из мест, представляющих внешний канал, появляется на связанной с модулем странице, которая в свою очередь содержит по одному модулю для каждого описания внутреннего подблока или процесса.

В сети, которая строится, места, созданные на предыдущем этапе и соответствующие внешнему каналу, сливаются с местами, соответствующими внутренним каналам, подключенным к внешнему таким способом. Назовем места, созданные на предыдущем этапе и соответствующие внешнему каналу, *старыми* местами, а места, созданные на этапе отражения подструктуры и соответствующие внутренним каналам – *новыми*.

Итак, каждому *старому* месту, которое служит входным для некоторого модуля, представляющего блок, соответствует свой набор *новых* мест, которые служат входными местами для модулей, представляющих подблоки данного блока. Аналогично, каждому *старому* месту, которое есть исходным для некоторого модуля, соответствует свой набор *новых* мест, которые служат выходными для модулей, представляющих собой подблоки данного блока.

В примере для *старого* места *c*1, исходного для модуля *B*1, соответствующими *новыми* местами будут *c*1.1 и *c*1.2, которые будут выходными местами для модулей *b*1.2 и *b*1.1 соответственно.

Каждое *старое* место – входное место, моделирующее блок модуля, сливается с соответствующими ему *новыми* местами, входными для модулей, моделирующих подблоки этого блока. Аналогично происходит слияние *ста-*

рого места – исходного для модуля, моделирующий блок, с соответствующими ему *новыми* местами – выходными для модулей, моделирующих подблоки этого блока. Заметим, что несколько мест, соответствующих внутренним каналам, могут быть слиты с одним местом, соответствующим внешнему каналу.

На рис. 5 показана страница для подструктуры *B*1, в которой места, соответствующие внутренним каналам, уже слиты с местами, соответствующими внешним каналам. Место *c*1 слито с местами *c*1.1 и *c*1.2, место *c*21 – с местами *c*2.1 и *c*2.21, а место *c*22 – с местом *c*2.22. В дальнейшем *слитым* местам будем приписывать те же имена, имеющие *старые* места на странице, моделирующей блок. Например, вместо имени *c*22 & *c*2.22 будет использоваться имя *c*22.

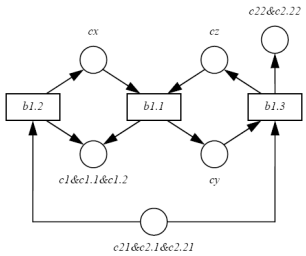

Рис. 5. Конечная сеть для подструктуры *B*1

Если описание любого блока дано за пределами описания системы (указана ссылка на отсутствие описания), то дальнейшего уточнения сети для этого блока не происходит. В конечной сети данному блоку будет соответствовать один модуль, порожденный на предыдущем этапе построения. При моделировании подструктуры канала на странице, связанной с дополнительным модулем, отражается подструктура этого канала аналогично отражению подструктуры блока.

Каждый блок, не расчлененный на подблоки, должен содержать хотя бы один процесс. Процессы между собой и с рамкой блока соединяются маршрутами. Моделирование блока, состоящего из процессов, аналогично моделированию блока любого уровня иерархии, при этом каждому описанию процесса сопоставляется один модуль. Отличие заключается в моделировании точки присоединения маршрутов к каналу. Сигнал, поступающий в блок по входному каналу, может передаваться по нескольким подсоединенным к нему маршрутам.

В описании семантики *SDL* точно не определяется, в какие маршруты будет передаваться сигнал. Каждый раз множество маршрутов, которые могут передать этот сигнал, и множество экземпляров процесса, которые могут получить сигнал, выбирается произвольно.

Система *SDLE*

Система представляет собой интегрированный программный комплекс для проектирования, анализа и симуляции моделей ИВТ-сетей, состоящая из транслятора с языка *SDL* и симулятора – блока имитационного моделирования.

Цикл работы пользователя в системе *SDLE* выглядит так. Строится исходная сетевая модель исследуемой системы или эта сетевая модель получается каким-либо другим способом, например как результат трансляции с языка выполнимых спецификаций *SDL*. Затем проводится симуляция модели в автоматическом режиме. Система позволяет визуально контролировать ход симуляции, отслеживать анализ структурных свойств поэтапно. По результатам симуляции исходная модель уточняется, и цикл разработки повторяется до получения удовлетворительных результатов [9].

Симулятор позволяет проследить за процессом функционирования сети при заданной начальной разметке. При этом часто удается выявить семантические ошибки модели, которые проявляются в неожиданном поведении сети.

Заключение. В статье описана процедура трансляции *SDL*-спецификаций с динамическими конструкциями в цветные сети Йенсена, обогащенные приоритетами. Расширение этих сетей приоритетами позволяет развить средства симуляции и анализа.

Способ моделирования основан на том, что в многоуровневом описании системы в *SDL* позиция каждого экземпляра процесса в общей иерархии системы остается неизменной, что позволяет транслировать описание системы в структуру сети, а экземпляры процесса моделировать с помощью фишек.

В результате работы алгоритма создается такая сетевая модель, в которой в каждом месте будет содержаться не более одной фишки, моделирующей некоторый экземпляр процесса. Таким образом, если во время функционирования системы может существовать *n* различных экземпляров любого процесса, то в любом месте моделирующей его сети может содержаться не более *n* фишек, причем каждая из них будет соответствовать своему экземпляру процесса. Это позволяет существенно повысить эффективность моделирования, так как существенно уменьшает перебор вариантов связывания переменных [9].