УДК 004.415.2:519.876.5

О.П. Доренський

Кіровоградський національний технічний університет, Кіровоград

МЕТОД СИНТЕЗУ ТЕСТОВИХ СТРУКТУР ВЗАЄМОДІЇ ПРОГРАМНИХ ОБ'ЄКТІВ ПІД ЧАС ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ ТЕХНОЛОГІЇ

У статті на основі положень об'єктно-орієнтованої методології, імітаційного моделювання та теорії мереж Петрі розроблено процедуру формалізованого представлення моделі кооперації програмних об'єктів у термінах формального апарату Е-мереж, алгоритм функціонування переходу Е-мережної тестової MUT-моделі взаємодії програмних об'єктів та процедура реєстрації результатів функціонування тестової моделі. За результатами дослідження запропоновано метод синтезу тестових структур взаємодії програмних об'єктів об'єктно-орієнтованого програмного забезпечення, який є основою для автоматизації процесу тестування взаємодії програмних об'єктів із метою виявлення системних й алгоритмічних помилок, а також отримання показників якості, які характеризують такі властивості складних програмних систем, як точність, безвідмовність, функціональна повнота.

Ключові слова: *програмне забезпечення, об'єктно-орієнтована технологія, програмний об'єкт, модель, Е-мережа, тестова структура.*

Вступ

Стрімкий розвиток інформаційних технологій, активне розроблення й впровадження автоматизованих систем обробки інформацій і керування зумовили створення відповідного програмного забезпечення (ПЗ). Воно характеризується багатокомпонентністю й складністю, повинне відповідати ряду жорстких вимог щодо надійності, високопродуктивності, гнучкості, живучості, а також можливості вдосконалення, масштабування, нарощування функціональності тощо [1, 2].

Комплексне врахування вказаних особливостей і характеристик ПЗ досягають шляхом застосування об'єктно-орієнтованої технології (ООТ) розроблення програмного забезпечення [1, 3, 4]. Проте навіть це не забезпечує завчасного виявлення і, відповідно, усунення системних й алгоритмічних помилок, переважна більшість яких виникає ще на етапі проектування складних програмних систем. Адже істотна надлишковість ПЗ, надвеликий об'єм даних, які циркулюють між його елементами, визначають суттєве ускладнення структур програмних об'єктів та їх взаємодії. Як наслідок, зростає кількість помилок, а, отже, збільшується тривалість тестування, розроблення й впровадження програмного забезпечення.

Аналіз останніх досліджень і публікацій показав, що на стадії проектування ПЗ застосовується об'єктна декомпозиція і прийоми представлення логічної (структура класів і програмних об'єктів) та фізичної (архітектура модулів та процесів), а також статичної й динамічної моделей ПЗ, яке розробляється за допомогою ООТ. Моделі створюються на основі предметів і явищ реального світу, описують поведінку програмної системи, пов'язаної зі станами об'єктів предметної області (ПрО) [2-4].

У дослідженні [5] для розроблення ПЗ за допомогою UML побудовано об'єктну модель, яка відображає основні абстракції ПрО, варіанти використання ПЗ, його фізичне уявлення, а також інформаційні потоки, які функціонують в ПЗ.

У роботах [6-8] запропоновано й обґрунтовано трирівневу архітектуру концептуальної моделі, яка відображає розвиток компонент багаторазового використання різного характера (структури моделей ПрО), а також специфікацію ПЗ, яка складається з логічної моделі, моделей використання, реалізації, процесів, розгортання. Крім цього, за допомогою UML побудовано відповідні об'єктні моделі, кожна із яких характеризує певний аспект системи, а в сукупності утворюють відносно повну модель ПЗ, яке розробляється. Однак отримані моделі, які відображають основні абстракції ПрО, фізичне уявлення ПЗ, по суті є набором моделей, що характеризують деякі аспекти програмної системи.

У дослідженні [9] з метою виявлення слабких ланок процесу тестування, аналізу й усунення помилок в ПЗ застосовано діаграму потоків даних як первинний засіб проектування, а задля визначення типу й пріоритету помилки як метод класифікації застосовано наївний класифікатор Байєса та класифікатор на основі нейронної мережі Кохонена. Таким чином, сформовано методологічний базис для

побудови прикладної системи аналізу помилок у програмному забезпеченні.

Постановка задачі. Задля зменшення кількості помилок у програмних системах слід забезпечити належне значення показника повноти його тестування: врахування необхідної кількості значущих факторів, які розглядаються під час розроблення ПЗ. У той же час, забезпечення необхідного рівня повноти тестування ПЗ щодо виявлення системних й алгоритмічних помилок є практично неможливим. Адже існуючі методи та засоби або не враховують ряду істотних особливостей об'єктно-орієнтованого програмного забезпечення [10], або спрямовані на розроблення тестів чи тестування вихідного коду програм. Існуючі методи об'єктно-орієнтованого моделювання не мають досить виразних засобів для здійснення детального дослідження динамічних аспектів ПрО на стадії проектування ПЗ з врахуванням семантики й часу. Особливо це стосується моделювання взаємодії елементів комплексної моделі об'єктно-орієнтованого програмного забезпечення (КМ ООПЗ) [2]. Таким чином, не забезпечується ні взаємне відображення структури керування, ні потоки даних програмного забезпечення і, як наслідок, унеможливлюється виявлення алгоритмічних та системних помилок на стадії проектування ПЗ.

Мета роботи полягає у розробленні метода синтезу тестових структур взаємодії програмних об'єктів об'єктно-орієнтованого програмного забезпечення задля досягнення необхідних значень показників функціональної повноти, безвідмовності й точності функціонування програмних систем на стадії їх проектування до стадії програмної реалізації.

Основна частина

На етапі тестування динамічної компоненти КМ ООПЗ [2] розробляються тестові структури, які є основою для прогону тестових випадків та реалізації методик тестування. Метод, що розробляється, ґрунтується на основних положеннях імітаційного моделювання, об'єктно-орієнтованої методології, теорії мереж Петрі [11, 12]. Він складається з сукупності наступних складових [13]:

- процедура формалізованого представлення моделі кооперації програмних об'єктів (МКПО) [2] у термінах формального апарату Е-мереж (синтез Model Under Test – MUT-моделі);
- алгоритм функціонування переходу Е-мережної тестової MUT-моделі взаємодії програмних об'єктів;
- процедура реєстрації результатів функціонування тестової моделі.

З врахуванням прийнятих та обґрунтованих обмежень й допущень [13], позначимо множину тестових випадків (ТВ) через STS (від англ. set of test cases). Тоді результат ТВ можливо представити як бієктивне відображення

$$F_{SidSod} : \sigma(SID) \to \sigma(SOD),$$

де $\sigma(SID)$ є множиною всіх підмножин множини вхідних даних SID (від англ. set of input dates) – екземплярів зовнішніх подій об'єктів, а $\sigma(SOD)$ є множиною всіх підмножин множини вхідних даних SOD (від англ. set of output dates).

Представлення тестової моделі взаємодії програмних об'єктів

Тестову модель взаємодії програмних об'єктів (ТМВПО) представимо в термінах формального апарата Е-мереж. МКПО [2] $MCO_{su} = (SO_{su}, ST_{su}, SL_{su}, SH_{su}, SR_{su}, F_{SotSl}, F_{SoSr})$, де SO_{su} – множина об'єктів, ST_{su} – множина акторів, SL_{su} – множина екземплярів подій, SH_{su} – множина діяльностей, SR_{su} – множина зв'язків, F_{SotSl} – функція породження ($F_{SotSl} : (SO_{su} \cup ST_{su}) \times SL_{su} \to \{0,1\}$, якщо $F_{SotSl}(sot, sl) = 1$, то сигнал керування $sl \in SL_{su}$ породжується об'єктом або зовнішньою сутністю sot), відображення $F_{SoSr} : SO_{su} \to SR_{su}$ визначає можливі зв'язки між об'єктами [2], $F_{NcoSsu}(MCO_{su}) = SSU_{su}$, $F_{Su\sigma(Ssu)}(su) = SSU_{su}$, перетворюється у відповідну Е-мережу N_E^{MVO} за наступним алгоритмом:

1) Кожному елементові множини об'єктів SO_{su} або множини акторів ST_{su} відповідає єдиний елемент множини Z^E описів переходів Е-мережі; тобто, визначено бієктивне відображення $F_{ZSot} : Z^E \to (SO_{su} \cup ST_{su})$.

2) За умови розгляду позиції мережі як деякої події, мітку, яка надійшла у поточну позицію, розглядатимемо як екземпляр події з множини SL_{sc}; оскільки структурно К-мережі відносять до підкласу маркованих графів, то певній події КМ ПЗ може відповідати кілька елементів множини B^E (за кількістю об'єктів та/або зовнішніх сутностей, які породжують екземпляри даної події).

Виходячи з того, що об'єкт або актор може приймати під час свого життєвого циклу не один екземпляр події від одного джерела, для представлення події КМ ПЗ використовуються позиції-черги. Тож, позначимо множину позицій-черг через B_Q^E, де $B_Q^E \subset B^E / B_P^E$, $B_Q^E \cap B_R^E = \varnothing$. При цьому принцип дії позицій-черг полягає в тому, що в

неї може вміщуватись кілька міток – накопичуються екземпляри події – і мітка, яка надійшла першою, першою ж з неї і виходить.

Відповідність елемента множини подій КМ ПЗ одному або кількох елементам множини B_Q^E можна визначити як сюр'єктивне відображення $F_{BSe}: B_Q^E \to SE_{su}$, де $\forall se \in SE_{su}(\exists sl \in SL_{su}, F_{SlSe}(sl) = se)$. Оскільки екземпляр події породжується об'єктом або зовнішньою сутністю, то тільки вирішальні позиції є периферійними; тобто $B_P^E = B_R^E$.

3) Пряма та зворотна функції інцидентності визначаються наступним чином:

$$F_{BZ}^E(b,z) = \begin{cases} 1, \text{ якщо } \exists st \in ST_{su}\, F_{Se\sigma(Sa)}^{ID}(se) = SA_{st}, F_{BSe}(b) = se, \\ F_{ZSot}(z) = st \text{ або } \exists so \in SO_{su}, F_{BSe}(b) = se, \\ F_{Se\sigma(Sa)}^{ID}(se) = F_{Se\sigma(Sa)}^{ID}(F_{SoSc}(so)), F_{ZDot}(z) = so; \\ 0, \text{ якщо умова не є виконаною;} \end{cases}$$

$$F_{ZB}^E(z,b) = \begin{cases} 1, \text{ якщо } \exists sot \in (SO_{su} \cup ST_{su}), F_{SotSl}(sot, sl) = 1, \\ F_{BSe}(b) = se, F_{SlSe}(sl) = se, F_{ZSot}(z) = so; \\ 0, \text{ якщо умова не є виконаною;} \end{cases}$$

Початкове маркування мережі визначається за формулою:

$$M_0^E(b) = \left| \cup_{l \in SL_{su}} \left\{ F_{Se\sigma(Sa)}(F_{SlSe}(sl)) \right\} \right|, \qquad (1)$$

де $\forall b \in B_Q^E(F_{BSe}(b) = se, F_{SlSe}(sl) = se, F_{SotSl}(sot, sl) = 1)$, а $\forall b \in B_P^E(M_0^E(b) = \varnothing)$.

З кожною міткою зв'язаний описувач, в якому задаватиметься черговість надходження мітки до позиції-черги. Тож, результат відповідності описувачів деякої мітки j до позиції-черги $b \in B_Q^E$ даними деякого екземпляра події $sl \in SL_{su}$ формалізується як ін'єктивне відображення $F_{bC=Sa}: \{b^j[i]\} \to F_{Se\sigma(Sa)}(F_{SlSe}(sl))$, де $b^j[i]$ є i-тим описувачем деякої мітки j в позицію-чергу b; $F_{Se\sigma(Sa)}(F_{SlSe}(sl))$ є множиною даних екземпляра подій $sl \in SL_{su}$ та $F_{BSe}(b) = se$, $F_{SlSe}(sl) = se$; $i = 1,...,n$, n – кількість описувачів міток у позиції-черзі b; $j = 1,...,m$, m – кількість міток у позиції-черзі b (кількість екземплярів однієї події, які надійшли від однієї сутності). При цьому вважатимемо, що $\forall b^j \in B_Q^E$ описувач мітки $b^j[0]$ відповідає означеному атрибуту-ідентифікатору класу обробника події, описувач мітки $b^j[1]$ – фіксатор часу надходження мітки до позиції-черги.

Задля забезпечення формальної єдності МКПО та її мережної тестової моделі, вводиться набір описувачів переходу z, який відповідає набору означених атрибутів об'єкта $so \in SO_{su}$. Результат цієї відповідності формалізуємо як ін'єктивне відображення $F_{zSa}: \{z[i]\} \to F_{Sc\sigma(Sa)}(F_{SoSc}(so))$, де $z[i]$ є i-тим описувачем переходу z, $F_{ZSo}(z) = so$, $i = 1,...,n$; n – кількість описувачів переходу z. При цьому надалі вважатимемо, що $\forall z \in Z^E(F_{zSa}(z[0]) = sa_{sc}^{ID}$, $sa_{sc}^{ID} \in SA_{sc}^{v}$, $F_{zSa}(z[1]) = sa_{ss}$, $sa_{ss} \in SA_{sc}^{v}$, $F_{SoSc}(so) = sc)$.

Умова збудження переходу представляється в наступному вигляді:

$$\forall z \in Z^E\left(\forall b \in F_{BZ}^{\beta}(z) \to \left(M^E(b) - F_{BZ}^{\beta}(b,z) = 0\right)\&\right.$$
$$\left.\forall b \in F_{ZB}^{\beta}(z) \to \left(M^E(b) - F_{ZB}^{\beta}(z,b) = M^E(b)\right)\right),$$

де множина $F_{ZB}^{\beta}(z) \subseteq F_{BZ}(z)$, множина $F_{ZB}^{\beta}(z) \subseteq F_{ZB}(z)$, $b \in B^E$, а склад підмножин $F_{BZ}^{\beta}(z)$ та $F_{ZB}^{\beta}(z)$ залежить від типу Е-мережного переходу β.

Умову збудження переходу представимо наступним чином:

$$\forall z \in Z^E\left(\exists b \in F_{BZ}^{\beta}(z) \to \left(M^E(b) \geq 1\right)\&\right.$$
$$\left.\forall b \in F_{ZB}^{\beta}(z) \to \left(M^E(b) \geq 0\right)\right), \qquad (2)$$

де $F_{BZ}^{\beta}(z) \subseteq F_{BZ}(z)$; $F_{ZB}^{\beta}(z) \subseteq F_{ZB}(z)$; $b \in B_Q^E$; до складу підмножини $F_{BZ}^{\beta}(z)$ входить єдина позиція, адже перехід з одного стану до іншого – єдина подія (екземпляр події).

Розроблення алгоритму функціонування переходу Е-мережної тестової моделі взаємодії програмних об'єктів

Задля побудови ТМВПО об'єктно-орієнтованого програмного забезпечення введемо новий тип переходу $YYXF_{En}$. Він відображає за деякою умовою пріоритетність руху однієї мітки відносно інших міток на вході і водночас дозволяє змінювати напрям руху за тією ж умовою одночасно в декількох напрямах на виході, де $\forall z \in Z^E(\exists so \in SO_{sc}, F_{ZSot}(z) = so)$.

Перехід $YYXF_{En}$ ТМВПО є об'єднання наступних макропереходів:

– Y_{En1} типу Y_{En}, який моделює пріоритетність обробки довільної кількості входів за деякою умовою; у контексті ТМВПО, Y_{En1} впорядковує екземпляри однієї події, які надійшли певному об'єкту з різних джерел;

– Y_{En2}, що визначає відповідність екземпляра події стану, в якому знаходиться заданий об'єкт;

– X_{En3} типу X_{En}, який моделює передавання інформації більш ніж у двох напрямках за деякою умовою; у контексті ТМВПО, X_{En3} визначає екземпляри подій, які породжуються об'єктом у новому стані.

– F_{En4} типу F_{En}, який моделює передачу інформації у понад два напрямки без умов; у контексті ТМВПО, F_{En} здійснює адресне надсилання екземпляра події, породженого об'єктом, у деякому стані.

Отже, алгоритм функціонування переходу типу $YYXF_{En}$ Е-мережевої тестової моделі N_E^{MVO} є наступним. Надходження мітки до вхідної довільної позиції-черги $b \in F_{BZ}^E$ або до кількох вхідних позицій переходу z типу $YYXF_{En}$ відповідає отриманню об'єктом $so \in SO_{sc}(F_{ZSot}(z) = so)$ екземпляра подій $sl \in SL_{sc}(F_{BSe}(b) = se, F_{SlSe}(sl) = se)$ та ініціює процедуру перевірки готовності переходу до спрацювання. Умови, які при цьому перевіряються, включають в себе наступні фази:

– аналізування розмітки вирішальної позиції $M^E(b_r)$, де $b_r \in B_R^E$;

– формування підмножин вхідних $F_{BZ}^E(z)^{in}$ та вихідних $F_{ZB}^E(z)^{out}$ позицій.

Оскільки $M_0^E(b_r) = \varnothing$, то на цій фазі здійснюється попереднє обчислення вирішальної процедури переходу $pr(z)$ для переведення позиції b_r з невизначеного стану. При цьому визначення вирішальної процедури є наступним: $rp: [\Pi_1 \to M^E(b_r) := F_{SsN}(ss_0); \Pi_2 \to M^E(b_r) := F_{SsN}(ss_1); ...; \Pi_{m+1} \to M^E(b_r) := F_{SsN}(ss_m)]$, де F_{SsN} є ін'єктивним відображенням множини станів SS_{sc} ($F_{SoSc}(so) = sc$, $so \in SO_{sc}$) у множину натуральних чисел, а $\Pi_1, ..., \Pi_{m+1}$ – предикати, які приймають значення поточного стану об'єкта: якщо Π_1 має значення істини, то $M^E(b_r) := F_{SsN}(ss_0)$ і подальше оцінювання не здійснюється; у іншому випадку – аналогічно обчислюється Π_2 і т.д. Після цього із значень $M^E(b_r)$ та відповідно до таблиці переходів відповідної моделі станів екземплярів класу програмних об'єктів [14] визначається підмножина вхідних позицій $F_{BZ}^E(z)^{in}$, до якої входять позиції, що відповідають події-образу, які приймаються у стані ss_i $(M^E(b_r) := F_{SsN}(ss_i))$. Аналізування розмітки вхідних позицій-черг $M^E(F_{BZ}^E(z)^{in})$ здійснюється згідно (2): наявність мітки хоч у одній вхідній позиції-черзі із підмножини $F_{BZ}^E(z)^{in}$ – макроперехід Y_{En2}. Якщо наявні мітки у декількох позиціях-чергах із поточної множини, тоді порівнюються значення описувачів часу надходження відповідних міток та обробляється мітка з тієї позиції, яка має найбільше значення описувача часу надходження – макроперехід Y_{En1}.

Наступним є макроперехід X_{En3}: визначення підмножини вихідних позицій $F_{ZB}^E(z)^{out}$, яке відповідає породжуваним подіям у новому стані, виходячи зі значення $M^E(b_r)$ та відповідно до виразів:

$$F_{Sh\sigma(Sha)}: SH_{sc} \to \sigma(SHA_{sc}),$$

де $\sigma(SHA_{sc})$ є множиною перетинаючих підмножин множини SHA_{sc};

$$F_{SsSh}: SS_{sc} \to SH_{sc};$$
$$F_{SesbSs}(se, ss_i, sbc) = ss_j.$$

Якщо подія може бути адресованою декільком програмним об'єктам, здійснюється аналіз значення описувача мітки $b^j[0]$ відповідної позиції – макроперехід F_{En3}. Аналіз розмітки вихідних позицій $M^E(F_{ZB}^E(z)^{out})$ відповідно (2) можна не здійснювати, оскільки позиції є чергами.

В результаті визначення вхідних та вихідних позицій, які обробляються, можуть бути отримані співвідношення для зміни розмітки вхідної й вихідних позицій переходу.

При досягненні необхідних умов, які відповідають отриманням об'єктом ПЗ деякого екземпляра події у поточному стані та можливості його переведення у новий стан відповідно моделі станів екземплярів класу, перехід z вступає в активну фазу. Її зміст визначається елементами $t(z)$ і $\zeta(z)$ опису переходу. При цьому виконанням процедури переходу $\zeta(z)$ вважатимемо ту частину діяльності у новому стані, яка пов'язана з модифікацією бази даних та/або бази подій: виконання операцій над значеннями атрибутів об'єктів та/або екземплярів класів.

Процедура переходу представляється в Е-мережі наступним чином:

$$\zeta: [\{\Pi_1 \to \{\phi_{li}^b\}\}],$$

де ϕ_{li}^b – операція над описувачем та деякою міткою в позиції b; Π_1 є предикатом, який визначає умови виконання набора операцій.

Черговість виконання процедури є наступною. Спочатку обчислюються предикати Π_1: визначається діяльність, яка асоціюється з новим станом $sh(F_{SsSh}(ss_1) = sh)$, після чого реалізується

необхідна підмножина операцій над описувачами. Розглядаються операції не тільки над описувачами мітки вхідної позиції, але й над описувачами переходу та міток вихідних позицій, які відповідають породжуваним об'єктом у заданому стані екземплярам подій відповідно функції F_{SsSha}.

У загальному вигляді операцію над i-тим описувачем мітки j у позиції b при істинності l-того предиката, який визначає набір умов, можна представити як $M^E\left(b^j[i]\right) := \phi_{li}^b\left(M^E\left(b^j[i]\right)\right)$, де $F_{bSa}\left(b^j[i]\right) = sa$, $sa \in F_{Se\sigma(Sa)}(se)$, $F_{bSe}\left(b^j\right) = se$. Тоді операцію над s-тим описувачем переходу z за умови істинності l-того предиката можна представити як $z[s] := \phi_{ls}^z(z[s])$, де ϕ_{ls}^z є операцією над описувачем переходу, $FzSa(z[s]) = sa$, $sa \in F_{SsC\sigma(Sa)}(sc)$, $F_{ZSo}(z) = so$, $F_{SoSc}(so) == sc$. Отже, процедура переходу $\zeta(z)\left(\forall z \in Z^E\right)$ матиме наступний вигляд:

$$\zeta : \left[\left\{\Pi_l \to \left(\left\{\phi_{li}^b\right\}^{in} \cup \left\{\phi_{ls}^z\right\} \cup \left\{\phi_{lj}^b\right\}^{out}\right)\right\}\right], \qquad (3)$$

де $\left\{\phi_{li}^b\right\}^{in}$ є множиною операцій над вхідною міткою, $\left\{\phi_{lj}^b\right\}^{out}$ є множиною операцій над вихідними мітками, тобто $F_{\zeta Sh} : \{\zeta(z)\} \to SH_{su}$, де $sh \in SH_{su}$ представляється згідно відповідної моделі діяльності програмних об'єктів.

Операція модифікації описувача переходу $z[l]$ відповідного атрибута поточного стану виконуватиметься не під час процедури переходу, а перед обчисленням та моделюванням часу $t(z)$, а значення $z[l]$ визначається згідно третьої фази перевірки готовності переходу.

Після завершення активної фази перехід вступає у фазу завершення, яка відповідає породженню об'єктом ПЗ нових екземплярів подій та вилученню з бази подій вхідного екземпляра подій як відпрацьованого. Змістом цієї фазі є зміна розмітки вхідної та вихідних позицій переходу: із оброблюваної вхідної позиції-черги вилучається мітка; вирішальна позиція стає невизначеною та виконується переміщення міток в ряд вихідних позицій-черг $F_{ZB}^E(z)^{out}$ відповідно з породжуваним об'єктом у новому стані екземплярами подій.

Завершення спрацювання переходу (за наявності міток у вхідних позиціях) або надходженні нової мітки у вхідну позицію ініціюють процедуру перевірки готовності переходу до наступного спрацювання. Якщо у деякому стані заданий об'єкт не породжує екземплярів подій, то перехід, який відповідає цьому об'єкту, розглядатиметься як перехід-зтік, запуск якого призводить до вилучення міток, не породжуючи інших.

Результат перетворення моделі кооперації програмних об'єктів у Е-мережеву тестову модель представимо таблицею еквівалентних перетворень (табл. 1)

.Розроблення процедури реєстрації результатів тестової моделі взаємодії програмних об'єктів

Реєстрація результатів функціонування тестової моделі кооперації об'єктів ПЗ, яке розробляється на основі ООТ, N_E^{MVO} здійснюється шляхом формування так званого сліду екземпляра каналу керування класу (СЕКК) для кожного переходу мережі.

Таблиця 1

Таблиця еквівалентності перетворень структури взаємодії програмних об'єктів у Е-мережну тестову модель

Елемент об'єктно-орієнтованих структур	Елементи Е-мережних тестових структур
Програмний об'єкт $so \in SO_{su}$, зовнішня сутність $st \in ST_{su}$	Е-мережний перехід $z \in Z^E$
Подія класу програмних об'єктів $se \in SE_{su}$	Позиція-черга Е-мережі $b \in B_Q^E$
Екземпляр події $sl \in SL_{sc}$	Мітка в позиції-черзі Е-мережі b
Атрибути події $se \in SE_{su}$, $se = F_{SlSe}(sl)$, $sl \in SL_{su}$	Множина $\left\{b^j[i]\right\}$ описувачів мітки j у позиції-черзі $b \in B_Q^E$
Атрибути класу програмних об'єктів $sc \in SC_{su}$, $sc \in F_{SoSc}(so)$, $so \in SO_{su}$	Множина $\{z[i]\}$ описувачів Е-мережного переходу $z \in Z^E$
Функція породження $F_{SotSl} : (SO_{su} \cup ST_{su}) \times SL_{su} \to \{0,1\}$	Зворотна функція інцидентності F_{ZB}^E
Взаємозв'язки програмних об'єктів $F_{SoSr} : SO_{su} \to SR_{su}$	Пряма та зворотна функції інцидентності F_{BZ}^E і F_{ZB}^E
Діяльність класу програмних об'єктів $sh \in SH_{su}$	Процедура Е-мережного переходу $\zeta : \left[\left\{\Pi_l \to \left(\left\{\phi_{li}^b\right\}^{in} \cup \left\{\phi_{ls}^z\right\} \cup \left\{\phi_{lj}^b\right\}^{out}\right)\right\}\right]$

Якщо перехід спрацьовує, то формуємо для цього переходу рядок сліду його спрацювань. При

цьому початкове маркування мережі N_E^{MVO} задається відповідно до формули (1). Це відповідає породженню програмним об'єктом деякого екземпляра події та встановлюються необхідні значення описувачів міток і переходів мережі згідно з відповідними значеннями атрибутів в базі даних й базі подій КМ ООПЗ [2].

Під слідом екземпляра каналу керування програмного об'єкта розумітиметься послідовність змін розмітки вхідних і вихідних позицій переходу. Для представлення СЕКК використовуватимемо відповідну таблицю (таблиця 2), яка встановлює критерії для вимірювання покриття MUT-моделей, які досліджуються. Вона також забезпечує розподіл результатів тестування за категоріями (за типами елементів моделі, які тестуються) та служить основою для формування каталогів помилок й звітів з описом помилок.

Таблиця 2

Слід екземпляра каналу керування об'єкта об'єктно-орієнтованого програмного забезпечення

Вхідна позиція / подія, яка приймається		Значення описувача переходу	Значення функції часу переходу	Вихідна позиція / подія, яка породжується	
Мітка / екземпляр події	Значення описувачів мітки			Мітка / екземпляр події	Значення описувачів мітки
–	–	$z[0]..z[m]$	–	–	–
b_l^1	$b_{l[0]}^1,...,b_{l[p]}^1$	$z[0]..z[m]$	t1	b_{j+1}^s	$b_{j+1[0]}^s,...,b_{j+1[q]}^s$
–	–	–	t2	b_{j+h}^{φ}	$b_{j+h[0]}^{\varphi},...,b_{j+h[\upsilon]}^{\varphi}$
b_i^k	$b_{i[0]}^k,...,b_{i[u]}^k$	$z[0]..z[m]$	t3	–	–
…					
b_{i+g}^k	$b_{i+g[0]}^k,...,b_{i+g[r]}^k$	$z[0]..z[m]$	tn	b_{j+1}^s	$b_{j+1[0]}^s,...,b_{j+1[q]}^s$

Для кожного переходу виконується наступна процедура формування СЕКК:

1) перший рядок таблиці СЕКК є завжди заповненим; причому заповнюється виключно четвертий стовпчик, який визначає значення описувачів переходу на момент початку виконання аналізу МКПО відповідно до бази даних комплексної моделі програмного забезпечення або тестового випадку;

2) після спрацювання переходу до другого стовпця таблиці СЕКК записується мітка вхідної позиції, яка наразі обробляється, відповідно до результатів виконання вирішальної процедури й аналізу розмітки вхідних позицій;

3) до третього стовпця таблиці СЕКК записуються значення описувачів мітки, яка надійшла у вхідні позицію та наразі обробляється;

4) до четвертого стовпця таблиці СЕКК заносяться значення описувачів переходу, які отримані в результаті виконання процедури переходу;

5) до п'ятого стовпця таблиці СЕКК записується результат виконання функції часу переходу;

6) до шостого стовпця таблиці СЕКК записуються мітки вихідних позицій, які наразі обробляються, відповідно до аналізу результатів виконання вирішальної процедури переходу; при цьому для кожної вихідної позиції використовується новий рядок;

7) до сьомого стовпця таблиці СЕКК вносяться значення описувачів міток, які надійшли до відповідної вихідної позиції й наразі обробляються.

В результаті повторного спрацювання переходу формується новий рядок таблиці СЕКК і т.д.

Після завершення або під час відстежування екземплярів каналів керування, які утворені в наслідок функціонування тестової моделі ВПО N_E^{MVO}, здійснюється контроль отриманих результатів з погляду коректності та несуперечності MUT-моделі. Його функції полягають у наступному:

– контроль поточних і кінцевих значень описувачів міток і переходів з погляду взаємної коректності, несуперечності й точності оброблення даних відповідних екземплярів подій та об'єктів, які беруть участь в МКПО;

– контроль послідовностей змін розміток вхідних позицій кожного переходу, який відповідає об'єкту, задля перевірки функціональності програмних об'єктів і можливості подальшої перевірки коректності синтезу моделі станів класу об'єктів, екземпляром якого є даний об'єкт;

– контроль послідовностей змін розміток вихідних позицій кожного переходу, який відповідає заданому об'єктові, задля визначення бажаної реакції у відповідь МКПО як наслідок породження одного первісного екземпляра події зокрема, так і

послідовності екземплярів подій загалом, з метою перевірки функціональності програмних об'єктів та визначення переходів, які не працювали;

– контроль виконання операцій переходів мережі (точності обробки даних) з можливістю подальшої перевірки коректності побудови відповідних моделей діяльностей класу програмних об'єктів, екземпляром якого є даний об'єкт;

– контроль часу, необхідного для того, щоб МКПО у вигляді примірника події, який очікується у відповідь, відреагувала на початковий екземпляр події в цілому або ж на будь-який екземпляр події зокрема.

Контроль коректності та несуперечності здійснюється шляхом порівняння результатів тестування, представлених у відповідних таблицях СЕКК, з очікуваними результатами із множини $\sigma(\mathrm{SOD})$.

У процесі фіксації й обробки результатів моделювання є можливим здійснення їх статистичного оброблення з метою отримання статистично стійких оцінок характеристик. Наостанок оцінюється ефективність тестування (повноти MUT-моделі) шляхом визначення у відсотковому співвідношенні покриття MUT-моделі тестовими випадками з погляду об'єктів та/або подій.

Висновки

За результатами дослідження запропоновано метод синтезу тестових структур взаємодії програмних об'єктів об'єктно-орієнтованого програмного забезпечення, який ґрунтується на основних положеннях об'єктно-орієнтованої методології, імітаційного моделювання та теорії мереж Петрі. Застосування синтезованих за допомогою зазначеного методу тестових моделей взаємодії програмних об'єктів ПЗ забезпечує досягнення необхідних значень показників функціональної повноти, безвідмовності й точності функціонування складних програмних систем на стадії їх проектування до стадії програмної реалізації.

Побудова й використання тестової моделі взаємодії програмних об'єктів дозволяє оцінити отримані MUT-моделі не тільки кількісно, а й забезпечує якісною оцінкою властивостей мережної моделі з погляду безвідмовності функціонування ПЗ, досяжності щодо оцінювання маршрутів виконання програми. Крім цього, повнота MUT-моделі дає можливість сформувати оцінку ефективності тестування, тобто визначити у відсотковому співвідношенні покриття MUT-моделі тестовими випадками.

Запропонований метод дозволяє перейти безпосередньо до автоматизації процесу тестування взаємодії програмних об'єктів програмного забезпечення, яке проектується та розробляється на основі об'єктно-орієнтованої технології, задля виявлення системних й алгоритмічних помилок, зокрема отримання показників якості, які характеризують такі властивості програмного забезпечення як точність, безвідмовність, функціональна повнота тощо.

Перспективою подальших розвідок є дослідження процесу створення тестових моделей поведінки програмних об'єктів під час тестування об'єктно-орієнтованого ПЗ на стадії його проектування. Цим можливо забезпечити комплексне тестування динамічної компоненти КМ ООПЗ, автоматизувати процес синтезу логічних та тестових моделей поведінки програмних об'єктів.

Список літератури

1. *Чабанюк Я.М. Оптимізація моделі тестування програмного забезпечення з показником величини проекту / Я. Чабанюк, В. Кукурба, Л. Гнатів і ін. // Комп'ютерні науки та інформаційні технології: Вісник Нац. ун-ту "Львівська політехніка". – 2011. – № 694. – С. 226-232.*
2. *Доренський О.П. Синтез структури комплексної моделі об'єктно-орієнтованого програмного забезпечення / О.П. Доренський // Системи обробки інформації. – Т. 2, Випуск 2(118). – 2014. – С. 68-72.*
3. *Орлов С.А. Технологии разработки программного обеспечения / С.А. Орлов. – СПб.: Питер, 2002. – 464 с.*
4. *Booch G. Object-Oriented Analysis and Design with Applications: 3rd edition / G. Booch. – Addison-Wesley Professional publisher, 2007. – 534 p.*
5. *Ємельянов В.О. Об'єктна модель програмного забезпечення захисту конфіденційних даних користувача / В.О. Ємельянов // Системи обробки інформації.– 2012. – Випуск 3 (101), Т.2 – С. 156-159.*
6. *Маевский Д.А. Априорная оценка количества дефектов в программном обеспечении информационных систем / Д.А. Маевский, С.А. Яремчук // Радіоелектронні і комп'ютерні системи. – 2012. – № 5 (56). – С. 73-80.*
7. *Жураковський Б.Ю. Об'єктно-орієнтована технологія проектування систем управління / Жураковський Б.Ю., Варфоломеєва О.Г., Гладких О.В., Хахлюк О.А. // Вісник ДУІКТ. – 2013. – №1. – С. 49-53.*
8. *Верес О.М. Застосування об'єктно-орієнтованого підходу до побудови моделі СППР / О.М. Верес, Ю.О. Верес // Вісник Нац. ун-ту "Львівська політехніка": Інформаційні системи та мережі . – 2013. – № 770. – С. 30-36.*
9. *Басюк Т.М. Особливості побудови системи аналізу помилок в ПЗ / Т.М. Басюк, А.М. Вальковський // Інформаційні технології та комп'ютерна інженерія: тези доп. IV Міжнар. наук.-практ. конф., 28-30 травня 2014 р. – Вінниця: ВНТУ, 2014. – С. 35-37.*
10. *Макгрегор Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие: Пер. с англ. / Джон Макгрегор, Девид Сайкс. – К.: ООО "ТИД ДС", 2002. – 432 с.*
11. *Buede D.M. The Engineering Design of Systems Models and Methods / Dennis M. Buede. – New York: Wiley, 2009. – 536 p.*
12. *Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. / Джеймс Питерсон. – М.: Мир, 1984. – 264 с.*

13. Доренський О.П. Алгоритм синтезу тестової моделі взаємодії екземплярів класу на стадії проектування об'єктно-орієнтованого програмного забезпечення / О.П. Доренський // Проблеми інформатики та комп'ютерної техніки: Праці Міжнар. наук.-практ. конф., 27-30 травня 2014 р. – Чернівці: Видавничий дім "Родовід", 2014. – С. 168-170.

14. Dorensky O.P. Method of the Models' Synthesis for Software Automated System Objects' States in Digital Images Processing / O.P. Dorensky // Збірник наукових праць Кіровоградського національного технічного університету. – 2014. – Випуск 27. – С. 283-292.

Надійшла до редколегії 29.04.2014

Рецензент: д-р техн. наук., проф. Ю.В. Кравченко, Государственный університет телекоммуникаций, Киев.

МЕТОД СИНТЕЗА ТЕСТОВЫХ СТРУКТУР ВЗАИМОДЕЙСТВИЯ ПРОГРАММНЫХ ОБЪЕКТОВ ПРИ ПРОЕКТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ ТЕХНОЛОГИ

А.П. Доренский

В статье на основе положений объектно-ориентированной методологии, имитационного моделирования и теории сетей Петри разработана процедура формализированного представления модели кооперации программных объектов в терминах формального аппарата Е-сетей, алгоритм функционирования перехода Е-сетевой тестовой MUT-модели взаимодействия программных объектов, процедура регистрации результатов функционирования тестовой модели. По результатам исследования предложен метод синтеза тестовых структур взаимодействия программных объектов объектно-ориентированного программного обеспечения, который является основой для автоматизации процесса тестирования взаимодействия программных объектов с целью выявления алгоритмических и системных ошибок, а также получения показателей качества, которые характеризуют такие свойства сложных программных систем, как точность, безотказность, функциональная полнота.

Ключевые слова: *программное обеспечение, объектно-ориентированная технология, программный объект, модель, Е-сеть, тестовая структура*

METHOD OF TEST STRUCTURE SYNTHESIS FOR OBJECTS INTERACTION DURING DESIGN SOFTWARE BASED ON OBJECT-ORIENTED TECHNOLOGY

O.P. Dorensky

This article presents the development of procedures for formal representation of software objects model of cooperation in terms of the formal apparatus of e-networks, algorithm of conversion of E-network test MUT models of software objects interaction and procedure for the registration of test model operation. They are designed on the basis of object-oriented methodology, simulation and theory of Petri nets. According to a research proposed the method of synthesis of test structures the interaction of software objects for object-oriented software. The method is a framework for automating the process of testing the interaction of software objects to identify systemic and algorithmic errors, and obtain quality indicators that characterize the accuracy, reliability, completeness functional complex software systems are designed.

Keywords: *software, object-oriented technology, software object, model, E-network, test structure.*