Ученые записки Таврического национального университета имени В.И. Вернадского
Серия «Экономика и управление». Том 26 (65). 2013 г. № 1. С. 12-19.

УДК 681.518

СИСТЕМНИЙ ПІДХІД ДО ПРОЕКТУВАННЯ СТІЙКОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ

Бойченко О.В.

У статті розглядаються особливості системного підходу до проектування стійкого програмного забезпечення інформаційних систем управління на основі графових моделей, що дозволяє організувати пошарове проектування програмного забезпечення
Ключові слова: системний підхід, стійке програмне забезпечення, графові моделі, пошарове проектування.

ВСТУП

Вирішення проблемних питань, пов'язаних із забезпеченням стійкості функціонування інформаційних систем управління (ІСУ) є першочерговим завданням вдосконалення інформаційних технологій та комп'ютерних мереж.

Зазначене визначається високими вимогами до живучості ІС, необхідністю забезпечення якісно нового рівня показників достовірності, доступності, конфіденційності, масштабності та повноти даних ІСУ, а також високими вимогами до внутрішніх та зовнішніх показників якості програмного забезпечення (ПЗ) ІСУ.

Однак невідповідність поточного рівня показників живучості, який здатна реалізовувати існуюча ІСУ, недостатність поточного рівня показників безпечності функціонування ПЗ ІСУ, невідповідність характеристик надійності (завершеності, відмово стійкості, відновлюваності) ПЗ ІСУ, що підтверджується результатами досліджень таких вчених, як В.М.Глушков, А.І. Сбітнєв, О.А. Павлов, О.В. Палагін, І.Б. Сироджа, В.А. Широков, О.В. Барабаш, Р.А. Калюжний, Н.В. Шаронова, М.Я. Швець та інших, визначає актуальність проведення наукового дослідження щодо розроблення методів проектування стійкого ПЗ ІСУ.

Метою даної роботи є розроблення методологічних основ застосування системного підходу до проектування стійкого ПЗ ІСУ.

Результати дослідження. Застосування графу макростанів ПЗ ІСУ є основою системного підходу у побудові ситуаційної моделі проектування ПЗ ІСУ. У відповідності до [1] для здійснення декомпозиції необхідно мати ситуаційну модель, що відображає множину ситуацій зовнішнього світу у множину станів ПЗ. У нашому випадку ситуації зовнішнього світу – це сукупність знань про впливи середовища, стан об'єкта управління, керуючої ланки і поточні цілі управління у даний момент часу.

Процес побудови ситуаційної моделі ілюструється рис. 1.

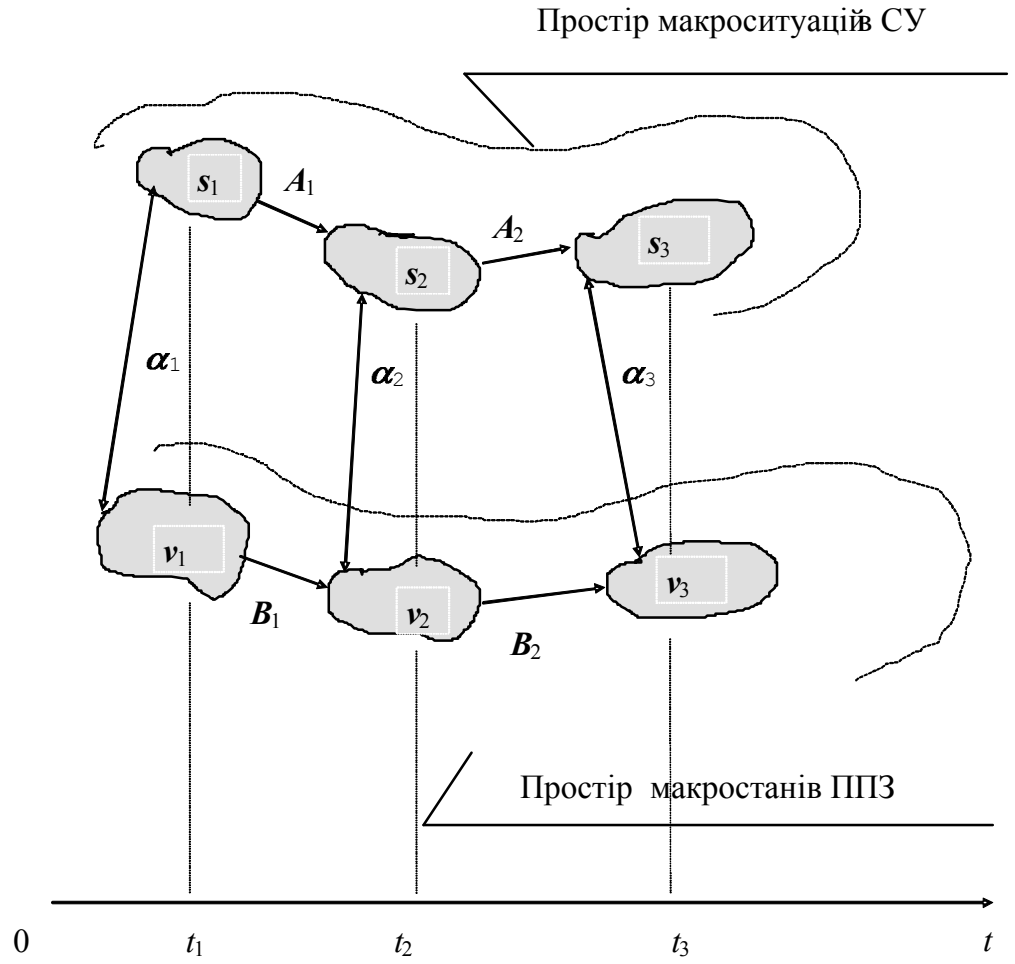

Рис. 1. До побудови ситуаційної моделі

Стани ПЗ – це такі сукупності задач і моделей ІСУ, які реалізуються у відповідній ситуації в ході вирішення задачі управління.

При описі ситуацій для ІСУ використовуються якісні характеристики, виражені вербально, тому ситуаційна модель, що формується у ІСУ, є лише близьким описом процесу управління.

Ситуаційна модель породжує модель станів ПЗ у тому сенсі, що кожній ситуації відповідає сукупність задач і моделей в ІСУ.

На рис. 1 $s_1, s_2, s_3, \ldots$ – ситуації, що склалися для ІСУ у відповідні моменти часу; $v_1, v_2, v_3, \ldots$ – стани ПЗ. Між ними є відповідність $E_i = \langle e_i, s_i, v_i \rangle$.

Нехай $A_1, A_2, \ldots$ – деякі оператори переходу із ситуації s_1 до s_2, із s_2 до s_3, ці оператори ініціюють виникнення деяких операторів $B_1, B_2, \ldots$, за допомогою яких здійснюється перехід ПЗ зі стану v_1 до v_2, v_2 до v_3 ... .

Згадані ситуації для ІСУ є узагальненими і називаються макроситуаціями. Відповідні їм стани ПЗ називаються макростанами.

Кожна макроситуація породжує макростан ПЗ ІСУ. Зв'язки і можливі переходи у просторі макроситуацій породжують відповідні переходи у просторі макростанів ПЗ.

Макростани і переходи між ними утворюють граф макростанів ПЗ ІСУ $L_m = (V, F)$, в якому $V = \{v_i\}$, $i \in I = \{1,2,\ldots,n\}$ – множина вершин; $F = \{\langle vi, vj\rangle\}, \ldots \langle vi, vj\rangle \in V_2, j \in J = \{1,2,\ldots,m\}$ – множина орієнтованих дуг графа, причому v_i — початок, а v_j – кінець дуги. У графі макростанів множина вершин інтерпретується як сукупність задач і моделей, що вирішуються у даному стані ПЗ, а зважені дуги показують напрямок і можливі ступені здійснення переходів ПЗ зі стану у стан.

Графом інформаційних зв'язків (інформаційним графом) [2,3] у нашому випадку будемо називати орієнтований граф $L_I = (Z, U_I)$, що відображає потік даних у системі задач таким чином, що якщо задача $z_j \in Z$ зв'язана за даними із задачею $z_i \in Z$, то $z_j \in U_I z_i$. Цей граф відповідає схемі інформаційних потоків у системі задач і будується за інформацією про вхідні і вихідні набори даних для кожної задачі.

Кожна сукупність задач у ПЗ реалізується під управлінням операційної системи. При цьому обмеження на порядок виконання задач задаються орієнтованим графом $L_C = (Z, U_C)$, де Z — множина задач, U_C — відображення Z у Z таке, що виконання z_j після z_i тягне $z_i \in U_C z_j$, тобто на графі L_C у цьому випадку є орієнтована дуга. Граф L_C називають графом управління, бо він відображає потік управління у системі. Він є базисним графом інформаційного графа і може бути отриманий вилученням транзитивних дуг останнього [93, 94].

Наявність даного графа дозволяє вирішувати задачі розподілу програм в обчислювальній системі, моделювати боротьбу за ресурс на початкових стадіях проектування.

Нехай задана множина елементів програмного забезпечення (структурні примітиви, модулі, інтерфейси і т. ін.) $A = \{a_j\}$. На цій множині задається

відношення включення R_6 таке, що $a_i R_6 a_j \leftrightarrow a_j \subset a_i$, тобто a_i і a_j зв'язані відношенням R_6 тоді і тільки тоді, коли елемент a_j є структурною компонентою елемента a_i [4].

Відношення R_6 породжує дуги на графі $L_s = (A, W)$

$$a_i R_6 a_j \leftrightarrow \langle a_j a_i \rangle; a_i, a_j \in A; \langle a_i, a_j \rangle \in W .$$

Графом структури програмного забезпечення (структурним графом) називається базисний граф графа L_s.

Даний граф може бути використаний при дослідженні питань стійкості програмного забезпечення.

У самому загальному випадку схема будь-якої програми містить оператори, що виконують дві функції: перетворення і розпізнавання. Відповідні оператори, що реалізують ці функції, назвемо перетворювачами (P_r) і розпізнавачами (P) (предикатами) [5].

Нехай існує робоча програма, що реалізує одну з задач ПЗ. Сукупність дій, що полягає у введенні даних, одноразовому виконанні програми і що закінчується отриманням результатів, назвемо прогоном програми.

На множині $\{\mathrm{Pr}\}$ перетворювачів введемо відношення R_n («виконується при одному прогоні»): $\mathrm{Pr}_i R_n \mathrm{Pr}_j$. На цій же множині введемо відношення R_{Bl} («завжди виконується при одному прогоні»), тобто $\mathrm{Pr}_l R_{Bl} \mathrm{Pr}_k$ значить, що якщо при прогоні виконується оператор Pr_l, то виконується Pr_k, і навпаки.

Відношення R_{Bl} має властивості симетричності, рефлексивності і транзитивності, тобто є відношенням еквівалентності, яке визначає на множині $\{\mathrm{Pr}\}$ деяке розбиття на класи еквівалентності $\{\mathrm{Pr}\} = \bigcup_j Bl_j, Bl_s \cap Bl_k = \otimes$.

Клас еквівалентності Bl_j назвемо програмним блоком.

Програмний блок – структурна програмна одиниця, що являє собою безперервну послідовність операторів і має таку властивість: при виконанні хоча б одного з операторів виконуються всі інші оператори послідовності [6].

Графовою моделлю програми (графом програми) називається орієнтований граф з позначеними дугами, в якому кожній вершині відповідає програмний блок.

$$L_p = (Bl, K), Bl = \{Bl_i\}, K = \{\langle Bl_i, Bl_j \rangle\}.$$

Дуга $\langle Bl_i, Bl_j \rangle$, позначена предикатом P, існує тоді і лише тоді, коли після виконання програмного блока, асоційованого з вершиною Bl_i, управління передається на вхід програмного блока, асоційованого з вершиною Bl_j, якщо предикат P істинний $(P) = 1$.

У графі програми шляхи з початкової вершини до кінцевої зображуються послідовністю дуг

$$l = \langle \langle Bl_1, Bl_2 \rangle, \langle Bl_2, Bl_3 \rangle, \ldots \langle Bl_{n-1}, Bl_n \rangle \rangle.$$

Тоді предикат шляху є кон'юнкцією предикатів дуг, що складають цей шлях

$$P^l = P_{1,2} \,\&\, P_{2,3} \,\&\, \ldots \,\&\, P_{n-1,n}.$$

Нехай Λ - множина вхідних даних програми, на графі якої задана множина шляхів $\{l\}$ з визначеними на них предикатами шляхів $\{P^l\}$.

Розглянемо два випадки [7]:

предметна область всіх предикатів P^l збігається з множиною Λ;

предметна область всіх предикатів P^l не збігається з Λ.

У першому випадку область істинності P^l збігається з деяким $\Lambda_i \subset \Lambda$. Вибір деякого елемента даних $\lambda_s \in \Lambda_i$ при прогоні програми забезпечує проходження крізь дуги (і відповідні вершини) шляху l_k. Області істинності предикатів шляхів у цьому випадку утворюють розбиття вхідних даних

$$\Lambda = \bigcup_i \Lambda_i; \Lambda_i \cap \Lambda_i = \otimes, i \neq j \text{ або } \Lambda = \bigcup_j Pl_{(u)}; P_{i(u)} \cap P_{j(u)} = \otimes.$$

При цьому

$$\Lambda_i = Pl_{(и)} = (\mathrm{P_{l1l2}} \,\&\, \mathrm{P_{l2l3}} \,\&\, \mathrm{P_{l(n\text{-}1)ln(и)}}) = \mathrm{P^{(u)}_{l1l2}} \cap \mathrm{P^{(u)}_{l2l3}} \cap \ldots \cap \mathrm{P^{(u)}_{l(n\text{-}1)ln}}$$

.

Це означає, що область істинності предиката шляху l_i є перетин областей істинності предикатів дуг, що його складають.

У другому випадку можна виділити варіант, коли предметна область деякої змінної предиката P^l збігається з деякою проекцією множини Λ на деяку вісь $P^l{}_{(u)} \subseteq \mathrm{пр}_{l1\ldots ls}\Lambda$, тобто області істинності предикатів шляхів утворюють розбиття деяких проекцій множини вхідних даних, а, отже, всієї множини.

Якщо такого варіанта виділити не вдається, проводиться перетворення предметних змінних предиката шляху в змінні з предметної області, яка збігається з

однією з проекцій Λ, тобто необхідно провести заміну змінних на деякі функції від змінних з вхідного набору.

Таким чином, твердження про те, що області істинності предикатів шляхів графа створюють розбиття множини вхідних даних програми, яка моделюється цим графом, виявляється справедливим і в другому випадку.

Таке подання програми у вигляді графової моделі є плідним, оскільки при завданні імовірнісного розподілу вхідних наборів дозволяє обчислити деякі важливі імовірнісні характеристики програм, а також провести додаткові дослідження ступеня їх структурної складності.

Граф програми L_p відображає потік управління у системі на більш низькому рівні (шарі) по відношенню до графа управління L_s.

Наведені графові моделі мають концептуальну єдність. Це дозволяє подати їх у вигляді графа мультиструктури, що описує ПЗ у цілому.

Для побудови мультиструктури вводиться операція детальної розшифровки [8].

Нехай кортеж $\gamma = \langle A, C, D, E \rangle$ описує один з рівнів формального подання, де A - множина вхідних вершин; C - множина вершин; D - множина міток; E - відображення множини $(A \cup C)$ у множину $C : E : (A \cup C) \times D \to C$.

Припускається, що будь-яка вершина $c \in C$ досяжна з деякої вершини $a \in A$.

Операція детальної розшифровки γ_1 за допомогою γ_2 у вершині *c* позначається $\gamma_1(c) \Leftarrow \gamma_2(a)$ і визначає новий рівень формального опису

$$\gamma = \langle A_1, C_1 \cup C_2, D_1 \cup D_2, E \rangle,$$

де $E(b,d) = E_1(b,d)$, якщо $b \in A_1 \cup C_1, d \in D_1$; $E(b,d) = E_2(b,d)$, якщо $b \in C_2, d \in D_2$; $E(b,d) = E_2(a,d)$, якщо $b = c, d \in D_2$.

На практиці операція детальної розшифровки має вигляд $\gamma_1(c) \Leftarrow \gamma_2$,

оскільки добре структурована програма повинна мати одну вхідну точку $(A2 = \{a\})$.

Наявність графових моделей дозволяє організувати пошарове проектування ПЗ, ставити і вирішувати оптимізаційні задачі структурного аналізу і синтезу, що розподілені по етапах, відповідних прийнятим рівням абстракції.

Важливою характеристикою ПЗ є його усталеність: спроможність зберігати певний рівень працездатності, незважаючи на несприятливі впливи зовнішнього (і внутрішнього) середовища [9].

Усталеність ПЗ до впливів, що збурюють, може бути охарактеризована одним із таких рівнів:

спроможність продовжувати виконання функцій у повному обсязі (переходити на резерв у тій або іншій формі);

спроможність виконувати задачі зі зменшенням обсягу функцій і, можливо, із зниженням якості (реалізувати шлях відступу);

спроможність переходити в стан «м'якої» відмови при помилці, що не піддається усуненню.

Таким чином, виявлення, діагностика і ідентифікація помилок ПЗ та його відновлення – системою заходів усталеності ПЗ. Практика роботи програм сучасних ІСУ вказує на низькі показники реалізації стійкого ПЗ з приводу відсутності системного підходу та іногрування можливості прояву помилок при застосування відомих методів і засобів контролю роботи програм.

ВИСНОВКИ

Системний підхід до проектування стійкого ПЗ ІСУ забезпечується: розробленням специфікацій, що базуються на визнанні факту можливості виникнення перекручувань у роботі обчислювальних засобів (ОЗ) і ПЗ; розробленням програмних засобів контролю і виправлення помилок у роботі ОЗ; вибором ОС, що має розвинуті засоби контролю виконання програм і роботи ОЗ, максимальним використанням і розвитком діагностичних можливостей ОС; використанням концепції головної задачі − інтерфейсу між ОС і ПЗ для організації контролю і відновлення; розробленням структури ПЗ, що використовує зворотний зв'язок між підпорядкованим і верхнім рівнем; розміщенням засобів контролю виконання ПЗ відповідно до рівнів ієрархії в системі; резервуванням найважливіших функцій ІСУ за допомогою локальних пристроїв; правильною експлуатацією ІС; відповідною організацією роботи колективу розроблювачів.

Реалізація усталеності ПЗ забезпечується виявленням, діагностикою і корекцією помилки з наступним відновленням. В даний час є велика кількість методів і засобів контролю роботи програм, проте, широкій практичній реалізації стійкого ПЗ заважає відсутність системності підходу, недооцінка можливості прояву помилок.

Список літератури

1. Сбитнев А. И. Декомпозиция программного обеспечения в многомашинных системах управления: Системный анализ / А. И. Сбитнев, Б.Д. Шепетюк. – К.: ИК АН УССР, 1982. С. 26–52.
2. Сбитнев А. И. Использование псевдофизического языка для описания ситуаций при проектировании СМПО КСАУ военного назначения / А. И. Сбитнев, А. Ю. Пермяков // Автоматика-98: 5-а українськ. конф.: тези допов. – Ч. 4. – К.: НТУУ КПІ, 1998. – С. 304–309.
3. Сбитнев А. И. Анализ графов математического обеспечения для нахождения параметров операционных систем: Системный анализ / А. И. Сбитнев, Б. А. Ромов. – К.: ИК АН УССР, 1974. – С 83-85.

4. Хетагуров Я. А. Проектирование информационно-вычислительных комплексов: монография / Я. А. Хетагуров, Ю. Г. Древс. – М.: Высшая школа, 1987. –280 с.
5. Bohm C. Flow diagrams, Turing machines and Languages with only two formation rules / C. Bohm, G. Jacopini // Comm. of ACM. –1986. –Vol. 9. – №5. – P. 366-371.
6. Сбитнев А. И. Структурная организация и проектирование математического обеспечения АСУ ТП: дис. … д-ра техн. наук: 05.13.11 / Сбитнев Анатолий Иванович. – К., 1989. – 447 с.
7. Сбітнєв А.І. Аналіз моделей життєвого циклу програмного забезпечення / А.І. Сбітнєв, С.В. Лєнков, О.М. Грищак, О.В. Горшков // Вісник Київського національного університету імені Тараса Шевченка. – К., 2005. – №10. – С. 36-39.
8. Вирт Н. Алгоритмы + структуры данных = программы: монография / Н. Вирт; [пер. с англ.] – М.: Мир, 1985. – 406 с.
9. Wirth N. On the Composition of Well-Structured Programs / N. Wirth // Computing Surveys. – 6. – №4, 1974. – PP. 257-259.

Бойченко О.В. Системный поход к проектированию устойчивого программного обеспечения информационных систем управления / Бойченко О.В. // Ученые записки Таврического национального университета имени В. И. Вернадского Серия: «Экономика и управление». – 2013. – Т. 26 (65). № 1. - С. 12-19.

В статье рассматриваются особенности системного подхода к проектированию устойчивого программного обеспечения информационных систем управления на основе графовых моделей, позволяющих организовать послойное проектирование программного обеспечения.

Ключевые слова: системный подход, устойчивое программное обеспечение, графовые модели, послойное проектирование.

Статья поступила в редакцию 02. 09. 2013 г.