УДК 004.4’232

О.В. Овсяк
Київський національний університет культури і мистецтв, Львівська філія;
Українська академія друкарства

МОДЕЛЬ АБСТРАКТНОЇ ПІДСИСТЕМИ КОМП’ЮТЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГЕНЕРУВАННЯ КОДУ

Описано модель створеного розширення функційних унітермів. Наведено синтезовану модель декомпозиції абстрактної підсистеми комп’ютерної системи генерування програмного коду. Описано побудовані моделі функційних унітермів. Наведено приклад фрагментів програмної реалізації синтезованої моделі абстрактної підсистеми.

Ключові слова: **модель, функційний унітерм, розширення, декомпозиція, підсистема.**

The created expansion of functional description uniterms. A synthesized decompositiv model of abstract subsystem for computer code generatiny system is prented. We describe a functional model based uniterm. An example program fragment of the synthesized model subsystem is gitem.

Keywords: **model, featured uniterm, expansion, decomposition, subsystem.**

Вступ і формулювання задачі

Алгебра алгоритмів [1, 2, 3] надає формальні засоби для опису, перетворення та оптимізації алгоритмів, а також, на підставі формул алгоритмів, дає змогу створити комп’ютерну систему генерування програмного коду. Сучасне об’єктне програмування основане на декомпозиції моделі створюваної інформаційної системи на класи. Самі ж класи утворені методами (процедурами), інтерфейсами, делегатами, перерахунками, властивостями, полями, змінними, операторами та інструкціями [4, 5], які в алгебрі алгоритмів названо унітермами. Самі ж методи (процедури) мов об’єктного програмування відповідають функційним унітермам алгебри алгоритмів [6, 7]. У дослідженнях [6, 7] описано створену модель опису фікційних унітермів, яка має такі складові, як назву функційних унітермів, вхідні параметри і змінні та значення вихідних параметрів і змінних. Але серед складових функційних унітермів немає ключів доступу і модифікаторів. Система класів комп’ютерної системи генерування програмного коду наведена у дослідженні [8]. Однак у цій роботі не розкрито змісту абстрактного класу *Term*, призначеного для задання методів опрацювання унітермів. Ці задачі розв’язуються у цій роботі.

Модель розширеного опису функційних унітермів

Порівняно з дослідженнями [6, 7], у яких трактування функційних унітермів, до деякої міри, обмежувалося аналогіями методів (процедур) мов об’єктного програмування, наприклад, таких загальновідомих алгоритмічних мов, як C#, C++, Java, Delphi та інших, у цій праці розглядатимо як функційні унітерми і класи мов об’єктного програмування. Функційні унітерми рівня класів мов об’єктного програмування, які описані засобами алгебри алгоритмів, називатимо підсистемами моделей інформаційних систем.

Функційні унітерми утворено двома частинами. Першу частину утворено загальними даними, якими є методи доступу, властивості, назва та вхідні й вихідні змінні й параметри. Друга частина утворена тим, що, власне, виконує функційний унітерм. Називатимо першу частину функційного унітерма заголовком, а другу – його змістом.

Модель опису заголовків підсистем

Поля моделі підсистеми містять такі дані: ідентифікатор підсистеми, яким є знак @; назву підсистеми, яка утворена літерами латиського алфавіту (великими або малими чи великими і малими), може мати цифри десяткової системи числення, але починатися літерою латинського алфавіту і не може містити знаків пропуску; ознаку наслідування інших підсистем, якою є знак: (двокрапки); розділених комами назви наслідуваних підсистем, якщо їх є більше від однієї (у назвах наслідуваних систем пропуски не допускаються, а у разі відсутності наслідуваних підсистем ознака наслідування не записується); назву системи, якої вона є підсистемою (назва системи записується такими самими знаками, як і підсистема), а сама назва системи може бути відсутньою; між назвами системи і підсистеми записується знак розділювача, яким є крапка (.), а коли назва системи є відсутньою, то розділювач не записується; параметри підсистеми, які утворені методами доступу (наприклад, загальний, обмежений, спеціальний та інші) і властивостями (наприклад, абстрактна, статична тощо).

Загальний формат моделі опису заголовка підсистеми має такий вигляд:

Md_V_Ns.@NpX:XNp1X,XNp2X,X … ,XNpnX=X,

де ***Md*** – метод доступу; ***_*** – знак наявності хоча б одного знаку пропуску (пробілу); ***V*** – властивість або властивості підсистеми; ***Ns*** – назва системи; ***Np*** – назва підсистеми; ***X*** – наявність або відсутність знаків пропуску; ***Np1***, ***Np2***, … ; ***Npn*** – назви наслідуваних підсистем.

Модель опису заголовків функційних унітермів

До відомої моделі опису функційних унітермів додатково вводимо методи доступу (наприклад, загальний, частковий, рекомендований тощо) і властивості (наприклад, абстрактний, змінений, новий тощо). Узагальнена структура моделі опису заголовка функційних унітермів має такий формат:

MdFu_VFu_(Xw1X,Xw2X, … XwnX)Nfu(Xv1X,Xv2X, … XvmX)X=X,

де ***MdFu*** – умовне позначення методів доступу до функційного унітерма; ***_*** – знак наявності хоча б одного знаку пропуску (пробілу); ***VFu*** – властивості (властивості) функційного унітерма; ***w1, w2, … wn*** – назви вихідних параметрів; ***Nfu*** – назва функційного унітерма; ***X*** – наявність або відсутність знаків пропуску; ***v1, v2, … vm*** – вхідні змінні та параметри функційного унітерма.

Декомпозиція підсистеми *Term*

Функційні унітерми підсистеми Term

В опрацюванні унітермів будуть використовуватися змінні, які потрібні для зберігання значень порівняння (позначимо *par)*, довжини (*wid*), ширини (*hei*) і вибору (*sel*) функційних унітермів. Задамо секвентні області значень цих змінних, які позначимо Q_0, $\underline{Q_1}$, $\underline{Q_2}$. Знак підкреслення цих секвентних областей означає, що вони є стандартними відомими значеннями. Запис належності значень змінних цим областям значень запишемо з використанням символа $\hat{I}$ (належності). Нехай $par\,\hat{I}\,\underline{Q_0}$ (наприклад, змінна є типу підсистеми), $wid\,\hat{I}\,\underline{Q_1}$ (наприклад, значення змінної є дійсними числами із заданого діапазону значень заданої точності), $hei\,\hat{I}\,\underline{Q_1}$ і $sel\,\hat{I}\,\underline{Q_2}$ (наприклад, ця секвентна область має логічні значення *true* і *false*). Для надання змінним початкових значень вводимо функційний унітерм, який позначимо *T()*. Задамо для нього метод загального доступу, який позначимо *pu*. Враховуючи заданий метод доступу, опис заголовка (назву функційного унітерма з назвами методів доступу, властивостями, вхідними і вихідними параметрами називатимемо заголовком) цього методу матиме такий вигляд *pu T()*.

Розміри знаків операцій секвентування, елімінування і паралелення залежать від розмірів унітермів, над якими вони вирисовуються. Тому необхідно мати функційний унітерм, призначений для обчислювання геометричних розмірів унітермів, який позначимо *Cs()*. Конкретні розміри унітермів обчислюються окремо у кожній конкретній операції, чим визначається абстрактна (*abs*) властивість функційного унітерма. Очевидно, що розміри унітермів залежать передусім від розміру кегля, який позначимо *f*, а його належність до підсистеми $@\underline{Siz}$ запишемо як $f\,\hat{I}\,@\underline{Siz}$. Знак

підкреслення є ідентифікатором стандартної відомої області значень. У підсистемі з унітермами оперуватимемо як з графічними обєктами. Тому вхідним параметром задаємо змінну *dv* яка належить до підсистеми графічного відображення *@DraV* (тут і надалі знак підкреслення означатиме, що ця підсистема є відомою і стандартною), що запишемо як *dv Î @DraV*. Задавши загальнодоступний метод доступу *pu*, отримаємо такий опис функційного унітерма

pu abs Cs(dv Î @DraV, f Î @Siz).

Вибір унітермів описується у підсистемі *@Mf*, а зняття вибору унітермів відбувається у підсистемах, якими описуються моделі формування операцій алгебри алгоритмів. Тому в підсистемі *Term* доцільно ввести абстрактний (*abs*) функційний унітерм деселекції вибору унітермів, який назвемо *Des()*. А для забезпечення доступу до параметрів підсистеми задання параметрів вибору *@Mf* необхідно створити вхідну змінну типу цього класу, що запишемо як *mf Î @Mf*. Тоді опис із заданням загального методу доступу (*pu*) є таким:

pu abs Des(mf Î @Mf).

Унітерми чи операції вибирають після встановлення на них курсора і натискання клавіші мишки та перевірки збігу їхніх геометричних координат з координатами місця розташування курсора, на якому відбулося натискання клавіші мишки. Для перевірки збігу уведемо функційний унітерм (позначимо його *Cfc()*). Такий збіг координат виконується для окремих унітермів та операцій. Тому доцільно ввести абстрактну властивість (*abs*) та загальний доступ (*pu*), доступ до параметрів головної підсистеми (*mf Î @Mf*), розмір кегля (*f Î @Siz*), координати унітерма чи формули, на якій відбулося натискання клавіші мишки (*x Î Q*$_1$, *y Î Q*$_1$), відступи від краю форми до початку робочого поля (*mx Î Q*$_1$, *my Î Q*$_1$) і координати курсора, у місці розташування якого натиснута клавіша мишки (*mox Î Q*$_1$, *moy Î Q*$_1$). Крім цього, задаємо повернення вихідних параметрів, якими будуть змінні підсистеми *Term* (*w Î @T*), значення яких використовуватимуться для виконання подальших дій. Враховуючи це все, функційний унітерм опишеться так:

pu abs (w Î @T)Cfc(mf Î @Mf, f Î @Siz, x Î Q$_1$*, y Î Q*$_1$*, mx Î Q*$_1$*, my Î Q*$_1$*, mox Î Q*$_1$*, moy Î Q*$_1$*).*

Рисування унітермів і знаків операцій описується окремим функційним унітермом, який для кожної операції чи унітерма враховує їхні особливості. Врахувати особливості можна, створивши у підсистемі загальнодоступний (*pu*) абстрактний (*abs*) функційний унітерм рисування (*Dra()*) з вхідними параметрами і змінними, якими є доступ до параметрів головної підсистеми (*mf Î @Mf*), розмір кегля (*f Î @Siz*), доступи до стандартних підсистем інструментів рисування (*bb Î @Br, gb Î @Br, sp Î @Pe, dp Î @Pe*), координати унітерма чи формули (*x Î Q*$_1$, *y Î Q*$_1$), відступи від краю форми до початку робочого поля (*mx Î Q*$_1$, *my Î Q*$_1$):

pu abs Dra(mf Î @Mf, f Î @Siz, bb Î @Br, gb Î @Br, sp Î @Pe, dp Î @Pe, x Î Q$_1$*, y Î Q*$_1$*, mx Î Q*$_1$*, my Î Q*$_1$*).*

Синтезовані формули алгоритмів записуються у пам'ять комп'ютера у вигляді файлів. Кожна операція алгебри алгоритмів має особливий формат *xml*-опису. Для врахування особливостей – подання кожної операції у підсистемі – створюємо для загального доступу (*pu*) абстрактний (*abs*) функційний унітерм *Cxml()* з вхідними параметрами документа (*d Î @DoXm*) й елемента (*e Î @ElXm*) *xml*-подання, опис якого є таким:

pu abs Cxml(d Î @DoXm, e Î @ElXm).

Для перетворення формули, яка подана *xml*-файлом, у графічно-текстову формулу створюємо функційний унітерм *Txml()* загального доступу (*pu*) зі статичною властивістю (*st*) і вхідними параметрами, якими є змінна типу підсистеми *Term*, що запишемо як *term Î @T* та змінна *n* типу стандартної підсистеми *@Nod*, використовувана для вибору унітермів з *xml*-файла. Вихідним є параметр *te Î @T*. Тому заголовок функційного унітерма має такий опис:

pu st (te Î @T)Txml(term Î @T, n Î @Nod).

З метою виконання замін операцій алгебри алгоритмів унітермами вводимо функціональний унітерм *Gct()* з методом загального доступу (*pu*) і статичною властивістю (*st*) та вхідними

параметрами типу підсистеми *(tT Î @T)* й типу цілого числа *z Î @Q_3*, заданого секвентною областю Q_3. Унітерм має вихідний параметр типу підсистеми *(w Î @T)*. Заголовок функційного унітерма є таким:

pu st (w Î @T)Gct (tT Î @T, z Î @Q_3).

Для виконання безпосередньої заміни операцій алгебри алгоритмів унітермами вводимо функційний унітерм *Rt(r)* з методом загального доступу (*pu*), статичною властивістю (*st*), трьома вхідними параметрами типу підсистеми (*rT Î @T, oT Î @T, nT Î @T,*) й одним вихідним параметром типу підсистеми (*x Î @T*). Опис заголовка функційного унітерма

pu st (x Î @T)Rt(rT Î @T, oT Î @T, nT Î @T).

Для ідентифікації вибору унітермів вводимо алгоритм

in (w Î @T)Cfc(p Î @B, f Î @Siz, x-y-mx-my-moX-moY Î @Dou),

де *in* – назва методу доступу з обмеженнями; *w* – вихідна змінна типу класу *T*; *p* – вхідна логічна змінна типу стандартного класу *B*; *x-y-mx-my-moX-moY* – вхідні змінні *x, y, mx, my, moX, moY* стандартного типу *Dou*, які призначені для задання координат місця знаходження унітерма (*x* і *y*), відступу робочого поля від лівого краю форми (*mx* і *my*) та координат курсора (*moX* і *moY*), які зафіксовано натисканням клавіші мишки.

Модель декомпозиції підсистеми Term

Заголовок підсистеми утворимо загальним методом доступу (*pu*), абстрактною властивістю (*abs*), назвою системи (*TE*) підсистемою якої вона є, ідентифікатором підсистеми (@) і назвою (*T*), що запишемо у вигляді

pu abs TE.@T.

Описані у підпункті 3.1 функційні унітерми уворюють підсистему ***Term***. Черговість розташування функційних унітермів у підсистемі ***Term*** не має значення. Тому для опису моделі застосовуємо операцію секвентування з розділювачем – комою, що дає таку формулу

pu abs TE.@T=

pu T(),
pu abs Cs(dv Î @DraV, f Î @Siz),
pu abs Des(mf Î @Mf),
pu abs (w Î @T)Cfc(mf Î @Mf, f Î @Siz, x Î Q_1, y Î Q_1,
mx Î Q_1, my Î Q_1, mox Î Q_1, moy Î Q_1),
pu abs Dra(mf Î @Mf, f Î @Siz, bb Î @Br, gb Î @Br,
sp Î @Pe, dp Î @Pe, x Î Q_1, y Î Q_1, mx Î Q_1, my Î Q_1),
pu abs Cxml(d Î @DoXm, e Î @ElXm),
pu st (te Î @T)Txml(t Î @T, n Î @Nod),
pu st (w Î @T)Gct(tT Î @T, z Î @Q_3),
pu st (x Î @T)Rt(rT Î @T, oT Î @T, nT Î @T),
in (w Î @T)Cfc(p Î @B, f Î @Siz, x-y-mx-my-moX-moY Î @Dou),

Моделі функційних унітермів

Модель підсистеми має предметні та абстрактні функційні унітерми. Предметними є функційні унітерми, властивості яких не є типу *abs*. До них належать унітерми з назвами *T(), Txml(), Gct()* і *Rt()*. Моделі абстрактних унітермів будуть описані у моделях інших підсистем.

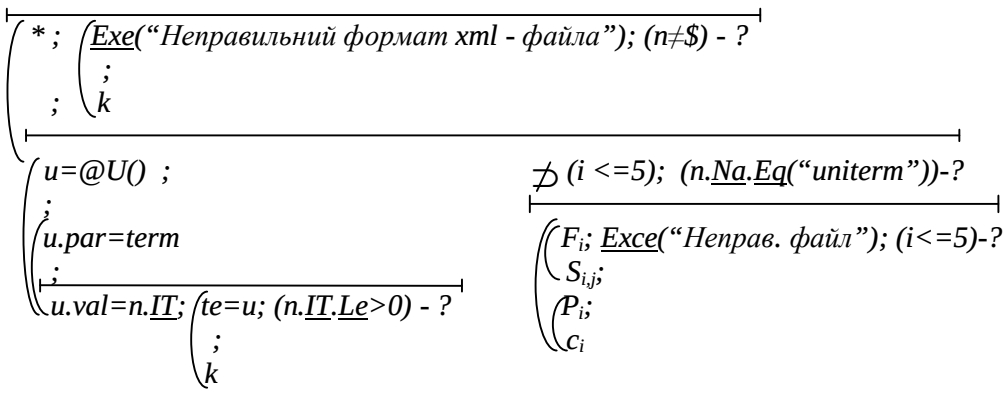
Модель функційного унітерма T()

Унітермам-змінним задаються початкові значення, що запишемо так:

$$pu\ T()=\overbrace{par=c_0,\ wid=c_1,\ hei=c_1,\ sel=c_2}\ ,$$

де c_0, c_1 і c_2 – початкові значення змінних. Черговість задання початкових значень змінним не має значення, тому в операціях секвентування розділювачем між унітермами поставлена кома.

Модель функційного унітерма *pu st (te Î @T)Txml(term Î @T, n Î @Nod)*
Загальний вигляд цього предметного унітерма є таким
pu st (te Î @T)Txml(term Î @T, n Î @Nod) =



де ***Exce()*** – стандартний алгоритм виведення на екран повідомлення, *$* – початкове значення змінної типу *@Nod*, *k* – кінець алгоритму (ознакою кінця може бути знак .), *u* – змінна типу класу *@U()*, *u.par* – присвоєння змінній *par* значення вхідної змінної *term*, вибір стандартної властивості ***IT*** із рядка *n* атрибуту і обчислення стандартним алгоритмом ***Le*** кількості символів у вибраному атрибуті, *te=u* – приписування вихідній змінній *te* значення *u*, *u.val=n.IT* – приписування вихідній змінній *val* значення атрибуту рядка *xml* - файла,

$i \hat{I} \overbrace{0; 1; \dots 5}$, $F_i = \overbrace{a_i = @b_i\ ;\ a_i.par = term}$, де при $i = 0$ $a_i = s$ і $b_i = S()$, при $i = 1$ $a_i = e$ і $b_i = E()$, при $i = 2$ $a_i = p$ і $b_i = P()$, при $i = 3$ $a_i = cs$ і $b_i = CS()$, при $i = 4$ $a_i = ce$ і $b_i = CE()$, при $i = 3$ $a_i = cp$ і $b_i = CP()$, $j \hat{I} \overbrace{0; 1}$,

$$S_{i,j} = \begin{cases} a_i.x_{i,j} = @\ b_i.\ y_{i,j}.z_{i,j}\ ; \\ a_i.x_{i,j} = @\ b_i.\ y_{i,j}.q_{i,j}\ ; \\ \underline{Exce}()\ ; \\ n.\underline{Attri}[\text{“}x_{i,j}\text{”}].\underline{Val}.\underline{Equ}(\text{“}z_{i,j}\text{”})\text{-?} \\ n.\underline{Attri}[\text{“}x_{i,j}\text{”}].\underline{Val}.\underline{Equ}(\text{“}q_{i,j}\text{”})\text{-?} \end{cases}$$

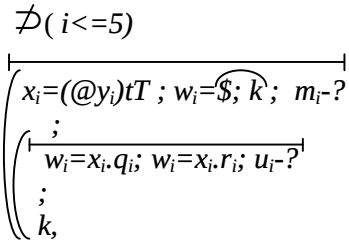
де при $i=0$ та $j=0$, $a_i=s$ і $b_i=S$, $x_{i,j}=sep$, $y_{i,j}=Sep$, $z_{i,j}=Semi$, $q_{i,j}=Com$; при $i=0$ та $j=1$ $a_i=s$ і $b_i=S$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$; при $i=1$ та $j=0$ $S_{i,0}=*$; при $i=1$ та $j=1$, $a_i=e$ і $b_i=E$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$; при $i=2$ та $j=0$ $a_i=p$ і $b_i=P$, $x_{i,j}=sep$, $y_{i,j}=Sep$, $z_{i,j}=Semi$, $q_{i,j}=Com$; при $i=2$ та $j=1$ $a_i=p$ і $b_i=P$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$; при $i=3$ та $j=0$ $S_{3,0}=*$; при $i=3$ та $j=1$ $a_i=cs$ і $b_i=CS$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$; при $i=4$ та $j=0$ $S_{4,0}=*$; при $i=4$ та $j=1$ $a_i=ce$ і $b_i=CE$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$; при $i=5$ та $j=0$ $S_{5,0}=*$; при $i=5$ та $j=1$ $a_i=cp$ і $b_i=CP$, $x_{i,j}=ori$, $y_{i,j}=Ori$, $z_{i,j}=Hor$, $q_{i,j}=Ver$;

$$P_i = \begin{cases} a_i.t_i = TF(a_i,\ n.\underline{CN}[0]); \\ a_i.r_i = TF(a_i,\ n.\underline{CN}[1]); \\ R_i; \\ te = a_i, \end{cases}$$

де при $i = 0$ $a_i = s$ і $R_i = *$, при $i = 1$ $a_i = e$ і $R_i = (a_i.w_i = TF(a_i,\ n.\underline{CN}[2]))$, при $i = 2$ $a_i = p$ і $R_i = *$, при $i = 3$ $a_i = cs$ і $R_i = *$, при $i = 4$ $a_i = ce$ і $R_i = *$, при $i = 3$ $a_i = cp$ і $R_i = *$, *TF()* – алгоритм зчитування з файла унітерма.

Модель функційного унітерма *pu st (w Î @T)Gct (tT Î @T, z Î @Q_3)*

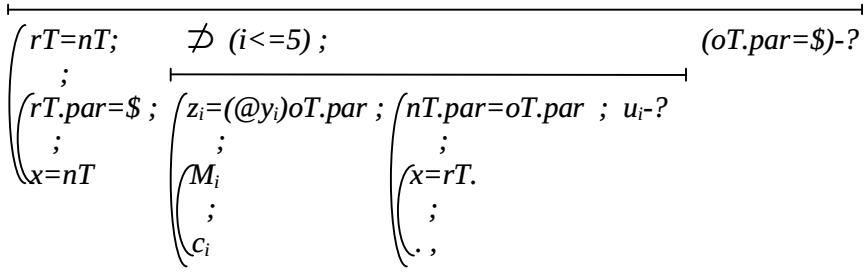
Формула моделі



де *i Î 0; 1; … 5,* а при *i=0* x_i*=s* y_i*=S,* q_i*=tA,* r_i*=tB,* u_i*=(z=1)-?,* m_i*=tT.GT().ToS().Equ(„TE.S”),* при *i=1* x_i*=e* y_i*=E,* q_i*=tA,* r_i*=tB,* u_i*=(z=1)-?,* m_i*=tT.GT().ToS().Equ(„TE.E”),* при *i=2* x_i*=p* y_i*=P,* q_i*=tA,* r_i*=tB,* u_i*=(z=1)-?,* m_i*=tT.GT().ToS().Equ(„TE.P”),* при *i=3* x_i*=cs* y_i*=CS,* q_i*=cond,* r_i*=term,* u_i*=(z=0)-?,* m_i*=tT.GT().ToS().Equ(„TE.CS”),* при *i=4* x_i*=ce* y_i*=CE,* q_i*=cond,* r_i*=term,* u_i*=(z=0)-?,* m_i*=tT.GT().ToS().Equ(„TE.CE”),* при *i=5* x_i*=cp* y_i*=CP,* q_i*=cond,* r_i*=term,* u_i*=(z=0)-?,* m_i*=tT.GT().ToS().Equ(„TE.CP”),* описує заміну операцій секвентування, елімінування, паралелення, циклічного секвентування, циклічного елімінування і циклічного паралелення. Вибір операції відбувається за умовою m_i, а умовою вибирають унітерм, яким виконується замінювання.

Модель функційного унітерма *pu st (x Î @T)Rt(rT Î @T, oT Î @T, nT Î @T)*

Заміни унітермів операцій описуються формулою



де *i Î 0; 1; … 5,* а при *i=0* M_i*=* z_i*.*q_i*=nT;* z_i*.*r_i*=nT; (*z_i*.*q_i*=oT)-?* z_i*=s* y_i*=S,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.S”),* при *i=1* M_i*=* z_i*.*q_i*=nT; (*z_i*.*r_i*=nT); (*z_i*.*q_i*=oT)-?* z_i*=e* y_i*=E,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.E”),* при *i=2* M_i*=* z_i*.*q_i*=nT; (*z_i*.*r_i*=nT); (*z_i*.*q_i*=oT)-?* z_i*=p* y_i*=P,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.P”),* при *i=3* M_i*=(*z_i*.term=nT),* z_i*=cs* y_i*=CS,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.CS”),* при *i=4* M_i*=(*z_i*.term=nT),* z_i*=ce* y_i*=CE,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.CE”),* при *i=5* M_i*=(*z_i*.term=nT),* z_i*=cp* y_i*=CP,* q_i*=tA,* r_i*=tB,* u_i*=oT.par.GT().ToS().Equ(„TE.CP”).* Виконання умови *(oT.par=$)* означає, що замінюваний унітерм є порожнім і тому він замінюється новим унітермом. У циклі за умовою ідентифікується операція алгебри алгоритмів (секвентування (TE.S), елімінування (TE.E), паралелення (TE.P), циклічного секвентування (TE.CS), циклічного елімінування (TE.CE) і циклічного паралелення (TE.CP)).

Модель унітерма *in (w Î @T)Cfc(p Î @B, f Î @Siz, x-y-mx-my-moX-moY Î @Dou)* містить реалізацію у наслідуваних класах ідентифікації вибору унітермів.

Фрагменти програмної реалізації моделей

Однією з найсучасніших мов об’єктного програмування є C# [4, 5]. Фрагменти програм функційних унітермів класу наведемо написані власне цією мовою.

Програма заголовка підсистеми *Term*

Код моделі заголовка підсистеми має ідентифікатор простору назв (`namespace`), назву простору назв (`TE`) і початок (`{`) та кінець (`}`) опису його вмісту, метод доступу до класу

(public), тип класу (abstract), ідентифікатор опису класу (class), знаки початку ({) і кінця опису вмісту класу (}) та є таким

```
namespace TE
  {
   public abstract class Term
        {

     }
}
```

Програмна реалізація моделі функційного унітерма *pu T()*

Програма опису унітерма є такою

```
public T()
     {
      par = null;
      wid = 0;
      hei = 0;
      sel = false;
     }
```

У цьому коді константи моделі функційного унітерма *T()*, якими є c_0, c_1 і c_2, набули значень null, 0 і false, відповідно, а специфікатор доступу *pu* у коді описаний як public.

Код абстрактних функційних унітермів *pu abs Cs(dv Î @DraV, f Î @Siz), pu abs Des(mf Î @Mf), pu abs (w Î @T)Cfc(mf Î @Mf, f Î @Siz, x Î Q1, y Î Q1, mx Î Q1, my Î Q1, mox Î Q1, moy Î Q1), pu abs Dra(mf Î @Mf, f Î @Siz, bb Î @Br, gb Î @Br, sp Î @Pe, dp Î @Pe, x Î Q1, y Î Q1, mx Î Q1, my Î Q1), pu abs Cxml(d Î @DoXm, e Î @ElXm)* має такий вигляд

```
public abstract void Cs(DrawingVisual dv, Size f);
public abstract void Des(MainForm mf);
public abstract Term Cfc(MainForm mf, Size f, double x, double y,
        double marginX, double marginY, double mouseX, double mouseY);
public abstract void Dra(MainForm mf, Size f, Brush bb, Brush gb, Pen
      sp, Pen dp, double x, double y, double marginX, double marginY);
public abstract void Cxml(XmlDocument xmlDoc, XmlElement node);
```

Фрагмент програми алгоритму *pu st (te Î @T)Txml(term Î @T, n Î @Nod)* є таким

```
public static Term TermFromXML(Term term, XmlNode n)
  {
   if (n == null)throw new Exception("Неправильний формат файла XML.");
   if (n.Name.Equals("uniterm"))
     {
      Uniterm u = new Uniterm();
      u.par = term;
      if (n.InnerText.Length > 0) u.val = n.InnerText;
      return u;
      }
    else if (node.Name.Equals("seq"))
          {
           Sequence s = new Sequence();
           s.par = term;
           if (n.Attributes["sep"].Value.Equals("semi"))
             s.sep = Sequence.Separator.Semicolon;
           else if (n.Attributes["sep"].Value.Equals("com"))
                  s.sep = Sequence.Separator.Comma;
```

```
                else throw new Exception("Неправильний формат файла
XML.");
            if(n.Attributes["ori"].Value.Equals("hor"))
                  s.ori = Sequence.Orientation.Horizontal;
            else if (n.Attributes["ori"].Value.Equals("ver"))
                  s.ori = Sequence.Orientation.Vertical;
                 else throw new Exception("Неправильний формат файла
XML.");
                  s.tA = TermFromXML(s, n.ChildNodes[0]);
                  s.tB = TermFromXML(s, n.ChildNodes[1]);
                  return s;
           }
                .
                .
                .

            else if (n.Name.Equals("cp"))
            {
            CyclicParallelisation cp = new CyclicParallelisation();
            cp.par = term;
            if (n.Attributes["ori"].Value.Equals("hor"))
            cp.orientation =
CyclicParallelisation.Orientation.Horizontal;
            else if (n.Attributes["ori"].Value.Equals("ver"))
            cp.orientation =
CyclicParallelisation.Orientation.Vertical;
            else throw new Exception("Неправильний формат файла XML.");
            cp.condition = TermFromXML(cp, n.ChildNodes[0]);
            cp.term = TermFromXML(cp, n.ChildNodes[1]);
            return cp;
            }
            else
            {
             throw new Exception("Неправильний формат файла XML.");
            }
       }
```

Фрагмент коду функціонального унітерма *pu st (w Î @T)Gct(tT Î @T, z Î @Q_3)*

```
 public static Term Gct(Term tT, int z)
    {
     if (tT.GetType().ToString().Equals("TE.S"))
       {
        Sequence s = (Sequence)tT;
        if (z == 1)return s.tA;
        else return s.tB;
       }
     else if tT.GetType().ToString().Equals("TE.E"))
            {
             Elimination e = (Elimination)tT;
             if (z == 1)return e.tA;
             else return e.tB;
            }
            .
            .
            .
     else if tT.GetType().ToString().Equals("TE.CP"))
```

```
            {
             CyclicParallelisation cp = (CyclicParallelisation)tT;
             if (z == 0) return cp.con;
             else return cp.term;
            }
     return null;
    }
```

***Фрагмент програми алгоритму** pu st (x Î @T)Rt(rT Î @T, oT Î @T, nT Î @T)*

```
public static Term Rt(Term rT, Term oT, Term nT)
        {
            if (oT.par == null)
            {
                rT = nT;
                rT.par = null;
                return rT;
            }
            if (oT.par.GetType().ToString().Equals("TE.S"))
            {
                S s = (S)oT.par;
                if (s.tA == oT) s.tA = nT;
                else s.tB = nT;
            }
                    .
                    .
                    .
            else if (oT.par.GetType().ToString().Equals("TE.CP"))
            {
                CP cp = (CP)oT.par;
                cp.term = nT;
            }

         n.par = oT.par;
         return rT;
        }
```

***Код функційного унітерма** in (w Î @T)Cfc(p Î @B, f Î @Siz, x-y-mx-my-moX-moY Î @Dou)*

Код утворено методом доступу (`internal`) до процедури (Cfc), типом повертаного процедурою значення (`Term`), вхідними параметрами процедури (`bool p, Size f, double x, double y, double mx, double my, double moX, double moY`), ідентифікаторами початку ({) і кінця (}) опису вмісту процедури, яким є назва вибраної стандартної процедури (`NotImplementedException()`) з параметрами (`throw new`)

```
   internal Term Cfc(bool p, Size f, double x, double y, double mx,
double my, double moX, double moY)
                        {
                         throw new NotImplementedException();
                        }
```

Підсумки

Створена методологія опису моделей комп'ютерних інформаційних систем передбачає виконання з використанням алгебри алгоритмів декомпозиції системи на підсистеми, з їхньою подальшою декомпозицією на функційні унітерми.

Введена модель опису заголовка підсистем зі спеціальним знаком підсистеми забезпечує однозначну швидку ідентифікацію заголовка і параметрів та змінних типу підсистеми.

Удосконалення моделі опису функційних унітермів забезпечує можливості задання методів доступу і властивостей.

Виконана декомпозиція підсистеми забезпечує можливості обчислення розмірів, ідентифікацію вибору, відмічення вибору, рисування і заміни унітермів, формування і аналізу *xml*-формату унітермів та заміни операцій алгебри алгоритмів унітермами.

Функційний унітерм аналізу *xml*-формату описує вибір унітермів з операцій секвентування, елімінування, паралелення, циклічного секвентування, циклічного елімінування і циклічного паралелення. Програмною реалізацією функційного унітерма аналізу *xml*-опису формул алгоритмів здійснюється вибір унітермів.

Функціональний унітерм заміни описує заміну унітермів операцій алгебри алгоритмів поданих *xml*-описом. Кодом моделі виконуються заміни унітермів.

1. Owsiak W., Owsiak A., Owsiak J. Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych. – Opole: Politechnika opolska, 2005. – 275 s. 2. Ovsyak V.K. Computation Models and Algebra of Algorithms. http://www.nbuv.gov.ua/Portal/natural/ VNULP/ISM/ 2008_621/01.pdf 3.Owsiak W., Owsiak A. Rozszerzenie algebry algorytmów // Pomiary, automatyka, kontrola. – № 2, 2010. – S. 184–188. 4. Petzold C. Programowanie Windows w języku C#. –Warszawa: „RM", 2002. – 1161 s. 5.Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C# 2008. – Москва, Санкт-Петербург, Киев: Apress, 2008. – 922 с. 6.Овсяк О. Моделі рекурсії та рекуренції / О.Овсяк // Вісник Нац. ун-ту "Львівська політехніка":" Комп'ютерні науки та інформаційні технології". – № 663, 2010. – С. 116 – 122. 7. Овсяк О.В. Граматика опису функційних унітермів / О.В. Овсяк // Поліграфія і видавнича справа: Збірник наукових праць Української академії друкарства. – № 2(50), 2009. – С. 18 – 22. 8. Овсяк О. Класи інформаційної систеи генерування коду / О. Овсяк // Вісник Тернопільського державного технічного університету: "Тернопільський національний технічний університет імені Івана Пулюя". – № 1, 2010. – С. 171 – 176.