аналіз і діагностика: моногр. / Ротштейн О. П., Ларюшкін Є. П., Мітюшкін Ю. І. — Вінниця: УНІВЕРСУМ-Вінниця, 2008. — 144 с. 11. Рывчин В. И. Техническое редактирование / Рывчин В. И., Леонардова Е. И., Овчинников А. И. — М.: Книга, 1977. — 248 с. 12. Саати Т. Принятие решений (Метод анализа иерархий) / Саати Т. — М.: Радио и связь, 1993. — 278 с. 13. Сеньківський В. М. Автоматизоване проектування книжкових видань: моногр. / В. М. Сеньківський, Р. О. Козак. — Львів: Укр. акад. друкарства, 2008. — 224 с. 14. Сикорский Н. М. Теория и практика редактирования / Сикорский Н. М. 2-е изд., испр. и доп. — М.: Высш. шк., 1980. — 328 с. 15. Сявавко М. С. Інформаційна система «Нечіткий експерт» / Сявавко М. С. — Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. — 320 с. 16. Тимошик М. С. Видавнича справа та редагування: навч. посіб. / М. С. Тимошик. — К.: Наша культура і наука — Концерн «Видавничий дім Ін Юре», 2004. — 224 с. 17. Чихольд Я. Облик книги: Избранные статьи о книжном оформлении / Ян Чихольд; (пер. с нем. В. В. Лазурского, В. П. Милютина, П. Ф. Чекрыжова). — М.: Книга, 1980. — 240 с. 18. Энциклопедия книжного дела / И. Г. Андреева и др. — М.: Юристъ, 1998. — 535 с.

СИНТЕЗ МОДЕЛИ ФАКТОРОВ ПРОЕКТИРОВАНИЯ КНИЖНЫХ ИЗДАНИЙ

Выбраны факторы влияния на процесс проектирования книжного издания. На основе экспертных суждений построена исходная модель связей между факторами, образованы матрицы зависимости и достижимости, с использованием которых синтезирована модель приоритетного влияния факторов на качество проектирования изданий.

SYNTHESIS OF MODEL OF FACTORS OF PLANNING OF BOOK EDITIONS

Factors which influence on the process of planning of book edition are chosen. On the basis of expert judgements the initial model of connections is built between factors, the matrices of dependence and reach, with the use of which the model of priority influence of factors is synthesized on quality of planning of editions, are formed.

Стаття надійшла 07.09.10

УДК 004

О. В. Овсяк

Українська академія друкарства

Львівська філія Київського національного університету культури і мистецтв

ДЕКОМПОЗИЦІЯ І МОДЕЛЬ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ РОБОТИ З УНІТЕРМАМИ

Засобами алгебри алгоритмів описана декомпозиція і побудована математична модель підсистеми введення, збереження та опрацювання унітермів. Модель програмно реалізована й апробована.

Декомпозиція, підсистема, модель, алгоритм, унітерм

ВСТУП І ФОРМУЛЮВАННЯ ЗАДАЧІ

Генератор програмного коду з формул алгоритмів [1, 3–5] є складною інформаційною системою. Окремі частини системи, якими виконуються тільки одна або декілька функцій, менш складні та є її підсистемами. Модель декомпозиції системи генерування програмного коду наведена у дослідженні [1]. Однією з підсистем інформаційної системи генерування коду є підсистема *Uniterm,* призначена для опрацювання унітермів. Однак у роботі [1] не розкрито зміст підсистеми *Uniterm*. Ця задача розв'язується в даній праці.

ДЕКОМПОЗИЦІЯ ПІДСИСТЕМИ *Uniterm*

Підсистема утворена заголовком, змінними і функційними унітермами. Частина підсистеми, яка має інформацію про метод доступу до неї, власну назву і назви наслідуваних нею підсистем, є заголовком підсистеми. Усе решта, що містить підсистема, називається моделлю, алгоритмом або функційним змістом підсистеми. Між заголовком і алгоритмом підсистеми записується знак =. Алгоритм підсистеми має змінні та підалгоритми (функційні унітерми).

Опис заголовка підсистеми

Оскільки підсистема має бути доступна з інших підсистем, наприклад підсистем секвентування (описує автоматизоване створення операції секвентування), елімінування, паралелення та інших [1], то задаємо до неї загальний метод доступу (*pu*). Введемо умовне позначення підсистеми *U* і задамо наслідування (*:*) нею абстрактної підсистеми *Term*, яку позначимо *T*. У такому разі заголовок підсистеми *U* матиме такий вигляд: *pu @TE.U:T*, де *@* — ідентифікатор підсистеми; TE — назва системи.

Змінні

Унітерми набувають значень. Для оперативного зберігання значень унітермів вводимо змінну *x*. Нехай її значення належать до стандартного текстового класу string [2, 6] (*@str*). Опис змінної: *x*∈*@str*. Для створення графічного абстрактного унітерма вводимо змінну *y*, яка належить до стандартного системного класу пензлів Brush [2, 6]: *y*∈*@Bru*. У створенні предметних унітермів використовуємо змінну *z*∈*@Bru*. У процесі обчислення кількості графічних фігур використовуємо змінну *k*∈*@int*, де *int* є стандартним класом цілих чисел *int* [2, 6]. Оперативне збереження номера вибраної формули алгоритму є значенням змінної *v*∈*@int*, У процесі рисування графічних елементів застосовуємо змінну стандартного класу рисування DrawingVisual [2, 6]: *d*∈*@DraVis*.

Функційні унітерми підсистеми Uniterm

Для задання змінним початкового значення вводимо підалгоритм загального методу доступу: *pu U()*.

Обчислення геометричних розмірів унітермів опишемо підалгоритмом загального методу доступу *pu*, який замінює аналогічний функційний унітерм

(властивість *over*, що є стандартною властивістю override [2, 6]) абстрактної підсистеми *T*, має назву *Cs* і два вхідних параметри *c*∈*@DraVis* та *s*∈*@Siz*, де *Siz* є назвою стандартного класу Size [2, 6]: *pu over Cs(c*∈*@DraVis, s*∈*@Siz)*.

Для деселекції вибраного унітерма вводимо функційний унітерм, який має загальний метод доступу (*pu*), власну назву *Des()* і вхідний параметр *mf*∈*@Mf*: *pu over Des(mf*∈*@Mf)*замінює (*over*) аналогічний метод абстрактного класу *T*.

Селекція унітермів описується функційним унітермом р таким заголовком: *pu over (k*∈*T) Des(mf*∈*@Mf, f*∈*Siz, x*|*y*|*mx*|*my*|*moX*|*moY_*∈*Dou)*, де вираз *x*|*y*|*mx*|*my*|*moX*|*moY*∈*Dou* описує приналежність значень змінних *x, y, mx, my, moX, moY* до стандартного класу Double [2, 6] . Змінні *mx, my* описують відступи від краю форми до робочого поля інтерфейсу користувача інформаційної системи, *x, y* — змінні координати унітерма, *moX, moY* — координати курсора маніпулятора «миш» у місці натискання лівою клавішею «мишки».

Рисування унітермів описується функційним унітермом такого заголовка: *pu over Dra(mf*∈*@Mf, bb*∈*@Bru, gb*∈*@Bru, sp*|*dp*∈*Pen, x*|*y*|*mX* |*mY*∈*Dou)*, де *bb* і *gb* — назви елементів стандартного класу *Pen* [2, 6] інструментів рисування.

Заголовок підалгоритму побудови *xml* — опису унітермів — має такий вигляд: *pu over Cxml(xD*∈*XmlDoc, n*∈*XmlEle),* де *XmlDoc* і *XmlEle* є стандартними класами XmlDocument і XmlElement [2, 6].

Функційні унітерми форматування висоти і ширини унітермів мають заголовки *pr (x*∈*Dou) Fh(t*∈*Str)* і *pr (x*∈*Dou) Fw(t*∈*Str)*, де *pr* — приватний (private [2, 6]) метод доступу до підалгоритму, *x* — відформатований текст, *t* — вхідний текст для форматування.

Підалгоритм урахування регіональних особливостей тексту має заголовок *pr (x*∈*@ForTex) Ft(t*∈*@Str)*, де *ForTex* скорочена назва відомої підсистеми форматування тексту FormattedText [2, 6].

Для обчислення кількості графічних елементів, які є на робочому столі системи, вводимо заголовок функційного унітерма *pu (k*∈*@int) Kf(mf*∈*@Mf)*, де *k* — вихідний параметр кількості фігур, *Kf* — власна назва функційного унітерма, *mf* — назва вхідного параметра типу підсистеми *Mf*.

Функційний унітерм усунення графічних елементів вводимо з таким заголовком: *pu Sf(n*∈*@int, mf*∈*@Mf)*.

Формула декомпозиції підсистеми Uniterm

Підсистема *Uniterm* декомпозована на змінні і підалгоритми (функційні унітерми), утворена всіма описаними вище змінними і функційними унітермами. Взаємне розташування змінних і підалгоритмів не має значення, оскільки черговість їх виконання визначається черговістю вибору. Для виконання вибору використовуються назви функційних унітермів, тому для опису підсистеми з наведених змінних і функційних унітермів застосовуємо операцію секвентування з розділювачем, який являє собою кому та має такий вигляд:

```
pu @U:T =
(
 (x ∈@str ,
  y ∈@Bru ,
  z ∈@Bru ,
  k ∈@int ,
  v ∈@int ,
  d ∈@DraVis ,
  =x;$,
  pu over Cs(c ∈@DraVis, s ∈@Siz) ,
 (pu over Des(mf ∈@Mf) ,
  pu over (k ∈T) Sel(mf ∈@Mf, f ∈Siz, x|y|mx|my|moX|moY ∈Dou),
  pu over Dra(mf ∈@Mf,bb|gb ∈@Bru,sp|dp ∈Pen,x|y|mX|mY ∈Dou),
  pu over Cxml(xD ∈XmlDoc, n ∈XmlEle),
 (pr (x ∈Dou) Fh(t ∈Str),
  pr (x ∈Dou) Fw(t ∈Str),
 (pr (x ∈@ForTex) Ft(t ∈@Str),
  pu (k ∈@int) Kf(mf ∈@Mf),
  pu Sf(n ∈@int, mf ∈@Mf)
```

МОДЕЛІ ФУНКЦІЙНИХ УНІТЕРМІВ

Нехай функційним унітермом *U()* змінній *x* задається початкове значення *$*, що запишемо секвенційним унітермом $=\overbrace{x;\$}$, де *$* — початкове значення.

Задаємо початкове функціонування системи з виводу на робочий стіл інтерфейсу користувача абстрактного унітерма. Нехай його розміри визначаються довжиною і шириною великої літери X і вибраним користувачем кеглем. Якщо ж користувачем системи вже задано унітерм, то визначаються його розміри. Такий алгоритм описує функційний унітерм обчислення розмірів унітермів (1). *r*∈@*Siz* описує створення змінної типу *Siz*, *r=@Siz(Fw(x), Fh(x))* — обчислення довжини (*Fw(x)*) і ширини (*Fh(x)*) унітерма *x*, *r=@Siz(Fw(«X»), Fh(«X»)* — обчислення довжини (*Fw(x)*) і ширини (*Fh(x)*) унітерма *X*, *w=r.Wid* і *h=r.Hei* — виділення стандартними властивостями *Wid* (Width [2, 6]) і *Hei* (Height [2, 6]) довжини (*w*) і ширини (*h*) унітерма.

```
pu over Cs(c ∈@DraVis, s ∈@Siz) =

(r ∈@Siz
 ;
 (r=@Siz(Fw(x), Fh(x))
    ;
  r=@Siz(Fw("X"), Fh("X")                    (1)
    ;
  x≠$
 ;
 (w=r.Wid
   ;
  h=r.Hei,
                                              ,
```

Модель функційного унітерма деселекції описується формулою

$$pu\ over\ Des(mf \in @Mf) = \overbrace{=sel;\ fa}: = \overbrace{mf.cD;fa}^{},$$

де $\overbrace{=sel;\ fa}$ — приписування змінній селективного вибору (*sel*) значення *false*, $\overbrace{mf.cD;fa}$ — секвенційний унітерм приписування змінній *cD* підсистеми *Mianform* значення *false*.

Алгоритм селективного вибору унітермів оснований на виявленні належності координат натисканням клавішею «мишки» прямокутній області унітерма й описується такою формулою:

$$\begin{array}{l} pu\ over\ (k \in T)\ Sel(mf \in @Mf,\ f \in Siz,\ x|y|mx|my|moX|moY \in Dou) = \\ \left(\begin{array}{l} re = @Re(x,\ y,\ w,\ h); \\ \hline \left(\begin{array}{l} \overbrace{=mf.cD;tru};\ k = S;\ (sel = re.Cont(moX,\ moY))\text{-}? \\ ; \\ \left(\begin{array}{l} mf.selU = tru \\ ; \\ k = th\ . \end{array} \right. \end{array} \right. \end{array} \right. \end{array}$$

де *re=@Re(x, y, w, h)* — опис створення прямокутника для заданих координат *x* і *y*, довжини *w* і ширини *h* з використанням стандартного методу Rect() [2, 6]; *tru* — значення істинності true; *sel=re.Cont(moX, moY)* — використання системного методу Contains()[2, 6] для визначення попадання координат натисканням «мишки» в прямокутник унітерма і встановлення ознаки вибору унітерма *sel* у відповідне істинне значення; *selU* — змінна підсистеми *U* вибору утерма; *th* — значення змінних класу *T*.

Функційний унітерм рисування утворимо з такими складовими, як усунення графічних елементів (*Uge*), рисування рамки вибору (*Rrv*), рисування унітерма (*Ru*) і запам'ятовування рамки вибору (*Zrv*) унітерма. Частини алгоритму впорядковані операцією секвентування, і тому загальний вигляд формули алгоритму рисування унітерма є таким:

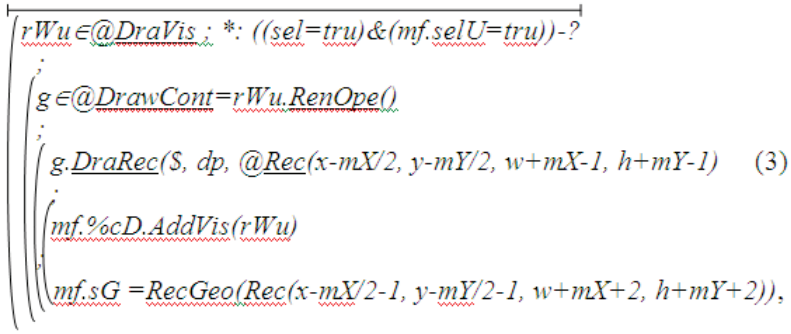
$$\begin{array}{l} pu\ over\ Dra(mf \in @Mf,\ bb|\ gb \in @Bru,\ sp|dp \in Pen,\ x|y|mX|mY \in Dou) = \\ \overbrace{Uge;\ Rrv;\ Ru;\ Zrv}^{}\ . \end{array}$$

Підалгоритм усунення графічних елементів починається з розпізнавання ознаки видалення. Після того вона скидається й обчислюється розмір прямокутника, графічні елементи якого будуть видалені. Далі розпізнається, чи існує зчитування з бази алгоритмів. Його наявність потребує порівняння розмірів обчисленого прямокутника з розмірами зчитуваної з бази алгоритмів формули. Якщо її розміри більші, то запам'ятовується вона замість обчисленого прямокутника і визначається кількість графічних елементів цієї області, усуваються згадпні елементи. Дана послідовність дій описується формулою (2). У ній *mf.zF=fal* — обнулення (*fal*) ознаки (*zF*) видалення графічних елементів, яка є змінною підсистеми *Mianform*; *mf.zF=tru* — перевірка встановленої ознаки видалення фігур; — *mf.sG = @RecGeo(Rec(x-mX/2-1, y-mY/2-1, w+mX+2, h+mY+2))* — обчислення (*@RecGeo(Rec(x-mX/2-1, y-mY/2-1, w+mX+2, h+mY+2))* з вико-

ристанням стандартного класу (RectangleGeometry, методу (Rect()[2, 6]) та запис у (*mf.sG*) прямокутної області; *≠sGp;$* — чи попередньо виділена область видалення унітермів не є порожньою; *<sG.Rec.Hei; mf.sGp.Rec.Hei* — чи ширина обчисленої області є меншою за ширину попередньо виділеної області; *<sG.Rec.Wid; mf.sGp.Rec.Wid* — чи довжина обчисленої області є меншою за довжину попередньо виділеної області; *mf.sG=mf.sGp* — змінній приписується значення розміру попередньо обчисленої області; *mf.sGf=mf.%cD.Gv(mf.sG)* — стандартним методом (GetVisuals()[2, 6]) виділення фігур заданої області (*sG*) робочого стола (*%cD,* де є ознакою графічного елемента) і приписування їх змінній (*mf.sG*) ; *mf.kF=mf.sGf.Cou* — обчислення стандартною властивістю (Count [2, 6]) кількості графічних елементів (*kF*), *(mf.kF>0)* — порівняння кількості формул з нулем, *i= mf.kF* — запис у кількості формул; *Sf(i, mf)* — функційний унітерм видалення фігур.

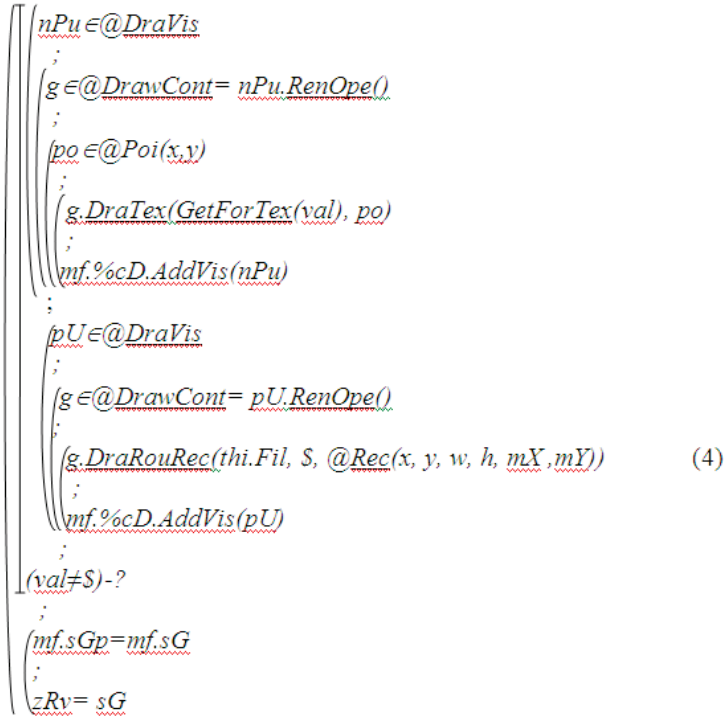
```
( mf.zF=fal; *; (mf.zF=tru)-?
  ;
  mf.sG = @RecGeo(Rec(x-mX/2-1, y-mY/2-1, w+mX+2, h+mY+2))
  ;
  ( mf.sG=mf.sGp; *; & ≠sGp;$;&( <sG.Rec.Hei; mf.sGp.Rec.Hei ;   (2)
      ;                          <sG.Rec.Wid; mf.sGp.Rec.Wid
    mf.sGf=mf.%cD.Gv(mf.sG)
      ;
    mf.kF=mf.sGf.Cou
    ;
    ( i= mf.kF; *: (mf.kF>0)-?
      ;
      ⊄ i≥0
      Sf(i, mf); *; ((i-1) ≥0)-?,
```

Алгоритм рисування рамки вибору (*Rrv*) включає обчислення рамки вибраного унітерма та обчислення і запам'ятовування рамки попередньо вибраного унітерма (3), де *(sel=tru)&(mf.selU=tru)* — перевірка наявності режиму вибору (*sel=tru*) і вибору унітерма (*mf.selU=tru*), *rWu∈@DraVis* — створення змінної стандартного класу DrawingVisual [2, 6], *g∈@DrawCont=rWu.RenOpe()* — створення змінної (*g*) стандартного класу (DrawingContext) і відкриття (RenderOpen()) її для запису графічних об'єктів, *g.DraRec($, dp, @Rec(x-mX/2, y-mY/2, w+mX-1, h+mY-1)* — стандартною процедурою DrawRectangle() приписування змінній *g* створеного стандартною процедурою Rect() прямокутника, *mf.%cD.AddVis(rWu)* — до змінної *mf* класу *MainForm* дописування стандартною процедурою AddVisual() створеного на робочому столі *%cD* графічного елемента, який знаходиться в *rWu*, *mf.sG=RecGeo(Rec(x-mX/2-1, y-mY/2-1, w+mX+2, h+mY+2)* — приписування змінній створеної стандартними процедурами RectangleGeometry() і Rect()рамки вибору унітерма.



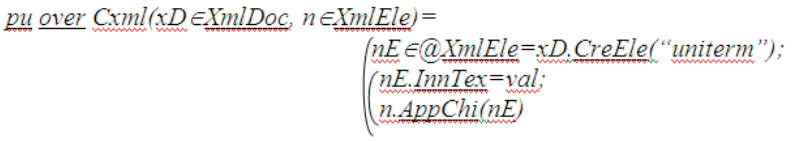
(3)

Алгоритм рисування унітерма (*Ru*) описує розпізнавання порожнього () унітерма. Якщо унітерм порожній, то вирисовується абстрактний графічний унітерм, яким є зафарбований прямокутник. Окремо вирисовується не порожній унітерм. Формула алгоритму описана виразом (4), де *po*∈*@Poi(x,y)* — створення змінної стандартного класу Point [2, 6], *g.DraTex(GetForTex(val), po)* — форматування і рисування стандартними процедурами GetFormattedText() і DrawText() значення унітерма, яке є значенням змінної *val* та збереження унітерма в змінній *g*, *g.DraRouRec(*thi.Fil*, $, @Rec(x, y, w, h, mX ,mY)* — створення стандартною процедурою DrawRoundedRectangle()повністю заповненого кольором прямокутника (this.Fill), отриманого стандартною процедурою Rect() і збереження його в *g*. Решта унітермів алгоритму описані в попередніх функційних унітермах.



(4)

Алгоритм формування опису унітерма є такою формулою:



де *nE*∈@XmlEle=xD.CreEle(«uniterm») — створення змінної *nE* типу стандартної підсистеми XmlElement і приписування їй стандартною процедурою CreateElement() значення *«uniterm»*, *nE.InnTex=val* — стандартною властивістю InnerText дописування до значення змінної *nE* значення унітерма, *n.AppChi(nE)* — стандартною процедурою AppendChild() дописування нового елемента до *xml* — формату формули алгоритму.

Моделі алгоритмів форматування тексту унітерма описуються формулами

де =x;GetForTex(t).Hei і =x;GetForTex(t).Wid — стандартним методом GetFormattedText() форматування висоти (Height) і довжини (Width) значення (*t*) унітерма і приписування його змінній *x*; *st*∈@FonSty=FonSts.Nor — створення змінної *st* типу стандартного класу FontStyle та її нормалізація стандартною властивістю Normal класу FontStyles; *tf@Typf=@Typf(@MaiFor.fF, st, @FonWei.Lig, @FonStr.Med)* — створення змінної *tf* типу стандартного класу Typeface і задання для неї стандартних властивостей Light і Medium; *x*∈@ForTex=@ForTex(t, @CulInf.CurCul, FloDir.LefToRig, tf, @MaiFor.fS, Bru.Bla) — форматування значення унітерму *t* стандартною процедурою FormattedText з урахуванням регіональних особливостей (CurrentCulture, LeftToRight) і приписування відформатованого значення змінній типу стандартного класу FormattedText; *x.TexAli=@TexAli.Lef* — задання стандартної прив'язки Left тексту унітерма.

Обчислення кількості графічних елементів робочого стола системи описується формулою (5), де *mcD∈@Poi=@Poi(mf.MinWid, mf.MinHei)* — стандартними властивостями MinWidth і MinHeight встановлення координат мінімальної довжини і ширини робочого стола і приписування їх значення змінній *mcD* типу стандартного класу Point; *acD∈@Poi=@Poi(mf.ActWid, mf.ActHei)* — стандартними властивостями ActualWidth і ctualHeight встановлення координат актуальної довжини і ширини робочого стола і приписування їх значення змінній *acD* типу стандартного класу Point; *g∈@ RecGeo= RecGeo(mcD, acD* — створення змінної *g* типу стандартного класу RectangleGeometry і приписування їй обчислених Rect() розмірів робочого стола; *fcD∈@Lis<DraVis>=mf.%cD.GetVis(g)* — створення змінної *fcD* типу стандартного списку List<DrawingVisual> і приписування їй стандартним методом GetVisuals() фігур робочого стола; *k=fcD.Cou* — обчислення стандартною властивістю Count кількості фігур на робочому столі.

$$
\begin{array}{l}
pu\ (k \in @int)\ Kf(mf \in @Mf) = \\
\left(\begin{array}{l}
mcD \in @Poi = @Poi(mf.MinWid,\ mf.MinHei); \\
acD \in @Poi = @Poi(mf.ActWid,\ mf.ActHei); \\
g \in @RecGeo = RecGeo(mcD,\ acD); \\
fcD \in @Lis<DraVis> = mf.\%cD.GetVis(g); \\
k = fcD.Cou\ ,
\end{array}\right.
\end{array}
\qquad (5)
$$

Функційний унітерм усунення графічних елементів опишемо формулою

$$
\begin{array}{l}
pu\ Sf(n \in @int,\ mf \in @Mf) = \\
\left[\begin{array}{l}
(sV \in @DraVis = (@DraVis) mf.\%cD.Nom(n) \\
mf.\%cD.DelVis(sV) \\
\ ; \\
ex \in @Exc = @MesBox.Sho(ex.Mes) \\
; \\
u\text{-}?,
\end{array}\right.
\end{array}
$$

де *sV∈@DraVis =(@DraVis)mf.%cD.Nom(n)* — створення змінної *sV* типу стандартного класу DrawingVisual і приписування їй номера графічного елемента процедурою Nomer() з перетворенням *(@DraVis)* до типу DrawingVisual; *mf.%cD.DelVis(sV)* — стандартним методом DeleteVisual усунення фігури *sV* з робочого стола *%cD; ex∈@Exc=@MesBox.Sho(ex.Mes)* — створення змінної ex стандартного класу Exception і приписування їй стандартною процедурою MessageBox.Show() висвітлюваного повідомлення про помилку, а *u* — умова виникнення помилки.

ФРАГМЕНТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МОДЕЛІ

Фрагмент програмної реалізації функційного унітерма наведено мовою C# [2, 6]

```
public override void Draw(MainForm mf, Size f, Brush bb, Brush
gb, Pen sp, Pen dp, /*int*/double x, /*int*/double y, /*int*/dou-
ble marginX, /*int*/double marginY)
      {
         if (mf.znuszczytyFigury == true)
         {
            mf.znuszczytyFigury = false;

               mf.isSelecting_Geometry = new RectangleGeom-
etry(
                 new Rect(x - marginX /2 - 1,
                        y - marginY /2 - 1,
                        width + marginX  + 2,
                        height + marginY + 2));

               if ( (mf.isSelecting_Geometry_Poperednia != null)
                    && (mf.isSelecting_Geometry.Rect.Height <
mf.isSelecting_Geometry_Poperednia.Rect.Height)
                     && (mf.isSelecting_Geometry.Rect.Width <
mf.isSelecting_Geometry_Poperednia.Rect.Width))
               {
                  mf.isSelecting_Geometry = mf.isSelecting_Ge-
ometry_Poperednia;
               }

            mf.isSelecting_Geometry_Figury = mf.canwasDraw.
GetVisuals(mf.isSelecting_Geometry);
            mf.isSelecting_KilkistFigur = mf.isSelecting_Geom-
etry_Figury.Count;
            if (mf.isSelecting_KilkistFigur > 0)
            {
               for (int i = mf.isSelecting_KilkistFigur - 1; i >= 0;
i--)
               {
                  SpalytyFiguru(i, mf);
               }
            }

            mf.isSelecting_Geometry_Figury = mf.canwasDraw.
GetVisuals(mf.isSelecting_Geometry);
            mf.isSelecting_KilkistFigur = mf.isSelecting_Geom-
etry_Figury.Count;
            if (mf.isSelecting_KilkistFigur > 0)
```

```
            {
                for (int i = mf.isSelecting_KilkistFigur - 1; i >= 0;
i--)
                {
                    SpalytyFiguru(i, mf);
                }
            }
        }
        if (selected && mf.selected_Uniter)
        {
            DrawingVisual ramkaWyboruUniterma = new Drawing-
Visual();
            using (DrawingContext g = ramkaWyboruUniterma.
RenderOpen())
            {
                g.DrawRectangle(/*bb*/null, dp, new Rect(x - mar-
ginX / 2,
                                                y - marginY / 2,
                                                width + (marginX /**
2*/) - 1,
                                                height + (marginY /**
2*/) - 1));
                mf.canwasDraw.AddVisual(ramkaWyboruUniterma);
            }
            mf.isSelecting_Geometry = new RectangleGeometry(
              new Rect(x - marginX / 2 - 1,
                    y - marginY / 2 - 1,
                    width + (marginX /** 2*/ + 2),
                    height + (marginY /** 2*/) + 2));

        if (value != null)
        {
            DrawingVisual nePoroznijUniterm = new DrawingVi-
sual();
            using (DrawingContext g = nePoroznijUniterm.Ren-
derOpen())
            {
                po = new Point(x, y);
                g.DrawText(GetFormattedText(value), po);
                mf.canwasDraw.AddVisual(nePoroznijUniterm);
            }
        }
        else
```

```
        {
            DrawingVisual poroznijUniterm = new DrawingVisu-
al();
            using (DrawingContext g = poroznijUniterm.Ren-
derOpen())
            {
                g.DrawRoundedRectangle(this.Fill, null, new  Rect(x,
y, width, height),
                                                    marginX, mar-
ginY);
                mf.canwasDraw.AddVisual(poroznijUniterm);
            }
        }

        mf.isSelecting_Geometry_Poperednia= mf.isSelecting_Ge-
ometry;
        mf.zbererzennia_RamkyWyboru = mf.isSelecting_Geom-
etry;
    }
```

Виконаною декомпозицією підсистеми введення й опрацювання унітермів зменшена складність проектування та застосовані стандартні процедури. Створено моделі функційних унітермів, які забезпечують обчислення розмірів абстрактних і предметних унітермів, ідентифікацію вибору, відміну вибору, рисування і заміну унітермів, формування xml — формату опису унітермів. Достовірність моделей підтверджено програмною реалізацією та апробацією.

1.Овсяк О. Класи інформаційної системи генерування коду /О. Овсяк //Вісн. Терноп. нац. техн. ун-ту імені Івана Пулюя. — 2010. — № 1, — С. 171 — 176. 2.MacDonald M. Windows presentation foundation в .NET 3.5 с примерами на C# 2008 /М. Мак-Дональд. — М., СП., К.: Apress, 2008. — 922 с. 3. Owsiak W. Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych /W. Owsiak, A. Owsiak, J. Owsiak. — Opole: Politechnika opolska, 2005. — 275 s. 4.Ovsyak V. K. Computation Models and Algebra of Algorithms [електронний ресурс] / V. K. Ovsyak. — Режим доступу: http://www.nbuv.gov.ua/Portal/natural/ VNULP/ISM/ 2008_621/01.pdf 5.Owsiak W. Rozszerzenie algebry algorytmów / W. Owsiak, A. Owsiak //Pomiary, automatyka, kontrola. — 2010. — № 2, — S. 184 — 188. 6.Petzold C. Programowanie Windows w języku C# /C. Petzold. —Warszawa: RM, 2002. — 1161 s.

ДЕКОМПОЗИЦИЯ И МОДЕЛЬ ИНФОРМАЦИОННОЙ ПОДСИСТЕМЫ РАБОТЫ С УНИТЕРМАМИ

Средствами алгебры алгоритмов описана декомпозиция и построена математическая модель подсистемы ввода, сохранения и обработки унитермов. Модель программно реализировано и апробировано.

DECOMPOSITION AND INFORMATION SUBSYSTEM MODEL WORK UNITERMS

By means of algebra decomposition algorithms are described and the mathematical model of the subsystem input, storage and processing uniterms. Model software is implemented and tested.

Стаття надійшла 15.10.10

УДК 004.94

І. В. Гілета

Українська академія друкарства

ІМІТАЦІЙНА МОДЕЛЬ РОЗРАХУНКУ ВАГОВИХ КОЕФІЦІЄНТІВ ПАРАМЕТРІВ МАКЕТА ШПАЛЬТИ

Окреслюється розроблена імітаційна модель розрахунку експертної оцінки узгодженості вагових коефіцієнтів параметрів макета шпальти .

Оцінка параметрів, метод парних порівнянь, імітаційна модель

Для прогнозування якості газетного видання потрібно провести моделювання його структури. Таке завдання передбачає декомпозицію процесу виготовлення макета газетної шпальти. На початковому етапі відбуваються виокремлення множини узагальнених параметрів та формалізація зв'язків між ними у вигляді графічної моделі ієрархічного графа [1]. Результатом такого моделювання є визначення вагомості впливу кожного параметра на структуру макета шпальти (рис. 1). Встановлення рівня важливості довільного параметра розроблюваного макета порівняно з іншими спричиняє необхідність оцінки пріоритетів параметрів та їх узгодженості з експертними судженнями.

При проектуванні шпальт газетних видань впливи між параметрами визначаються суб'єктивно на підставі експертних оцінок. Вони не завжди однозначні і часто виражаються в якісних оцінках. Для якісного аналізу потрібні кількісні оцінки. Систематичним методом вирішення цієї задачі є метод бінарних (або попарних) порівнянь, запропонований американським дослідником Т. Сааті [4]. Для цього використовується рангова дев'ятибальна шкала, яка дозволяє якісні оцінки параметрів відтворювати у шкалі відношень.За цією шкалою на основі думок експертів виставляється оцінка переваги одного параметра над іншим. Шкала створена за науковими даними. Порівняно з іншими відомими шкалами має ряд переваг. Базується на наступних положеннях:

1. Якісна різниця переваги одного об'єкта над іншим має достатню точність і на практиці може бути виражена числом.

2. Психофізіологічні властивості людини дозволяють досить добре розрізняти якісні відмінності об'єктів за шкалою з п'яти рівнів: немає переваги;