УДК 004

О. Овсяк

Українська академія друкарства,
Львівська філія Київського національного університету культури і мистецтв

РЕКУРЕНТНО-ДЕКОМПОЗИЦІЙНА МЕТОДОЛОГІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

Описана створена рекурентно-декомпозиційна методологія побудови моделей інформаційних технологій і систем. На її підставі побудовані структурна і функціональна рекурентно-декомпозиційні моделі інформаційної технології генерування програмного коду з формул алгоритмів.

The created recurrence-decomposition methodology of models of information technology and systems. On its basis created structural and functional recurrent decompositions models of information technology to generate code from formulas of algorithms.

Вступ і формулювання задачі

Сучасні інформаційні технології і системи мають графічні інтерфейси і реалізують достатньо багато різноманітних функцій. Проектування їх є складною науковою-технічною задачею. Проектування нових інформаційних технологій і систем, з метою зменшення його складності, виконується декомпозиційним методом. Для розбиття моделі використовуються різноманітні критерії, наприклад, функціонального призначення, модульності та інші.

Методом є прийом або система прийомів, що застосовується у якій-небудь галузі діяльності (науці, виробництві тощо) [1]. Методологією є сукупність методів дослідження, що застосовуються у будь-якій науці відповідно до специфіки об'єкта її пізнання [1]. Рекурентною формулою (*рекурентним співвідношенням*) є формула, яка описує обчислення всіх членів послідовності, якщо задано декілька її перших членів [2]. Декомпозицією є розбиття складної системи на підсистеми [3]. Розрізняють різні типи декомпозиції, наприклад, структурну, функціональну [3].

У загальному випадку структурна чи функціональна декомпозиція є багаторівневою і впорядкованою. Використовувана методологія декомпозиції базується на інтуїтивному описі впорядкованості. На підставі розширеної алгебри алгоритмів впорядкованість описується формально. Моделі інформаційних технологій і систем можуть бути побудовані на підставі рекурентно-декомпозиційної методології, формулювання якої і застосування до проектування описано нижче.

Рекурентно-декомпозиційна методологія

Суть рекурентно–декомпозиційної методології в аналітичному описі впорядкованої багаторівневої декомпозиції інформаційних технологій і систем на складові - унітерми з кінцевим розбиттям на тривіальні та функційні унітерми.

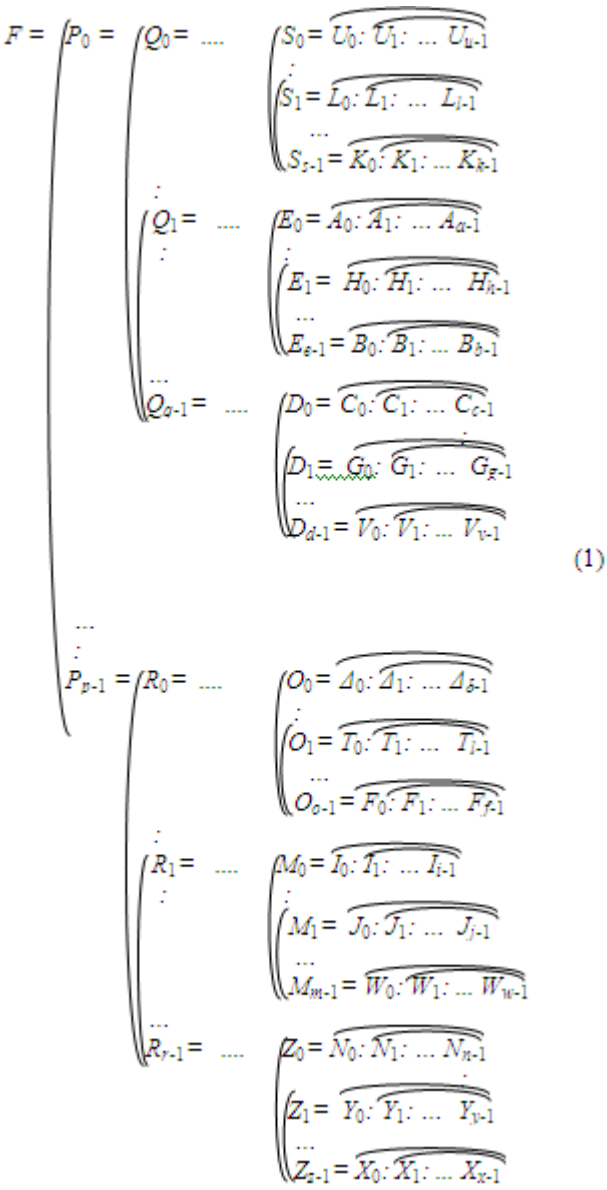
1.Проектування рекурентно-декомпозиційної структури

Нехай потрібно створити рекурентно-декомпозиційну структуру унітерма F. Для виконання декомпозиції необхідно вибрати ознаки за якими вона буде проведена. Наприклад, такими ознаками можуть бути виконувані функції, внутрішня специфіка створюваної технології чи системи, вимоги і обмеження та інші. Нехай ознака або декілька ознак є вибраними. За цими ознаками спочатку (на першому рівні) декомпозуємо унітерм F на "підунітерми" P_0 , P_1 , … , $P_{p\text{-}1}$. Очевидно, що в F входять усі p унітермів від P_0 до P_1. Обов'язкову наявність усіх унітермів описуємо операцією секвентування з розділювачем між унітермами двох крапок. У тому випадку коли має значення черговість (взаємне розташування) унітермів P_i та P_j, то у секвентуванні замість двох крапок застосовуємо крапку з комою. Якщо ж черговість немає значення, то розділювачем унітермів операції секвентування вибираємо кому. Між і секвенцією унітермів ставимо знак рівності.

По декомпозиції першого рівня виконуємо декомпозицію другого рівня, яка полягає у розбитті унітермів P_0 , P_1 , … , $P_{p\text{-}1}$ на підунітерми. Наприклад, унітерм P_0 декомпозовано на унітерми Q_0, Q_1, … , $Q_{q\text{-}1}$, а унітерм $P_{p\text{-}1}$ на унітерми R_0, R_1, … , $R_{r\text{-}1}$. Аналогічно розбиваємо всі решти унітермів. Кожному із унітермів P_0 , P_1 , … , $P_{p\text{-}1}$ приписуємо секвенцію його "підунітермів" з відповідними розділювачами. Серед унітермів P_0 , P_1 , … , $P_{p\text{-}1}$ можливі такі унітерми, які не підлягають або декомпозицію яких недоцільно виконувати.

Далі виконується декомпозиція на другому, третьому і подальших рівнях. Завершуєься виконання декомпозиції предметними і функцій- ними унітермами.

Загальний вигляд рекурентної моделі структурної декомпозиції описано формулою (1),



(1)

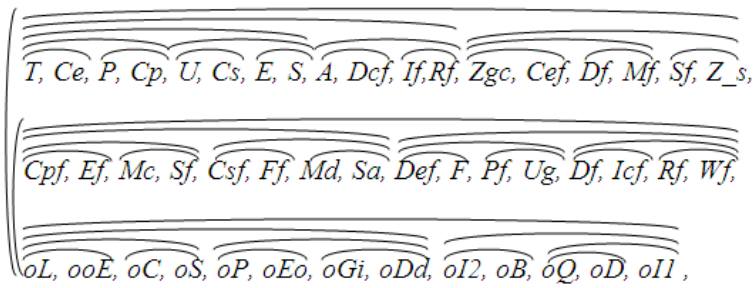
де: F – інформаційна технологія або система (унітерм F); P_0 , … , $P_{p\text{-}1}$ - підтехнології або підсистеми (унітерми P_0 , … , $P_{p\text{-}1}$) першого рівня декомпозиції унітерму F, p - кількість унітермів першого рівня декомпозиції; Q_0, Q_1, … , $Q_{q\text{-}1}$ - унітерми другого рівня декомпозиції унітерму F і першого рівня декомпозиції унітерму P_0, а q - кількість унітермів; S_0, S_1, … , $S_{s\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів P_0, Q_0, … , а s - кількість унітермів; U_0, U_1, … , $U_{u\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів F, P_0, Q_0, … , S_0, а u - кількість унітермів; S_1 – унітерм передостаннього рівня декомпозиції унітермів F, P_0, Q_0, … ; L_0, L_1, … , $L_{l\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів F, P_0, Q_0, … , S_1, а l - кількість унітермів; $S_{s\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів F, P_0, Q_0, … ; K_0, K_1, … , $K_{k\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів F, P_0, Q_0, … , $S_{s\text{-}1}$, а k - кількість унітермів; E_0, E_1, … , $E_{s\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів P_0, Q_1, … , а e

- кількість унітермів; $A_0, A_1, \ldots, A_{a\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_1, \ldots, E_0$, а a - кількість унітермів; E_1 – унітерм передостаннього рівня декомпозиції унітермів $F, P_0, Q_1, \ldots ; H_0, H_1, \ldots, H_{h\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_1, \ldots, E_1$, а h - кількість унітермів; $E_{e\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів $F, P_0, Q_1, \ldots ; B_0, B_1, \ldots, B_{b\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_1, \ldots, E_{e\text{-}1}$, а b - кількість унітермів; $D_0, D_1, \ldots, D_{d\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів $P_0, Q_{q\text{-}1}, \ldots$, а d - кількість унітермів; $C_0, C_1, \ldots, C_{c\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_{q\text{-}1}, \ldots, D_0$, а c - кількість унітермів; D_1 – унітерм передостаннього рівня декомпозиції унітермів $F, P_0, Q_{q\text{-}1}, \ldots ; G_0 : G_1 : \ldots G_{g\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_{q\text{-}1}, \ldots, D_1$, а g - кількість унітермів; $D_{d\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів $F, P_0, Q_{q\text{-}1}, \ldots ; V_0 : V_1 : \ldots V_{v\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_0, Q_{q\text{-}1}, \ldots, D_{d\text{-}1}$, а b - кількість унітермів; $R_0, R_1, \ldots, R_{r\text{-}1}$ - унітерми другого рівня декомпозиції унітерму F і першого рівня декомпозиції унітерму $P_{p\text{-}1}$, а r - кількість унітермів; $O_0, O_1, \ldots, O_{o\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів $P_{p\text{-}1}, R_0, \ldots$, а o - кількість унітермів; $\Delta_0 : \Delta_1 : \ldots \Delta_{\delta\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_0, \ldots, O_0$, а δ - кількість унітермів; O_1 – унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_0, \ldots ; T_0 : T_1 : \ldots T_{t\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_0, \ldots, O_1$, а t - кількість унітермів; $O_{o\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_0, \ldots ; F_0 : F_1 : \ldots F_{f\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_0, \ldots, O_{o\text{-}1}$, а f - кількість унітермів; $M_0, M_1, \ldots, M_{m\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів $P_{p\text{-}1}, R_1, \ldots$, а m - кількість унітермів; $I_0 : I_1 : \ldots I_{i\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_1, \ldots, M_0$, а i - кількість унітермів; M_1 – унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_1, \ldots ; J_0 : J_1 : \ldots J_{j\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_1, \ldots, M_1$, а j - кількість унітермів; $M_{m\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_1, \ldots ; W_0 : W_1 : \ldots W_{w\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_1, \ldots, M_{m\text{-}1}$, а w - кількість унітермів; $Z_0, Z_1, \ldots, Z_{z\text{-}1}$ - унітерми передостаннього рівня декомпозиції унітерму F і унітермів $P_{p\text{-}1}, R_{r\text{-}1}, \ldots$, а z - кількість унітермів; $N_0 : N_1 : \ldots N_{n\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_{r\text{-}1}, \ldots, Z_0$, а n - кількість унітермів; Z_1 – унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_{r\text{-}1}, \ldots ; Y_0 : Y_1 : \ldots Y_{y\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_{r\text{-}1}, \ldots, Z_1$, а y - кількість унітермів; $Z_{z\text{-}1}$ - унітерм передостаннього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_{r\text{-}1}, \ldots ; X_0 : X_1 : \ldots X_{x\text{-}1}$ - унітерми останнього рівня декомпозиції унітермів $F, P_{p\text{-}1}, R_{r\text{-}1}, \ldots, Z_{z\text{-}1}$, а z - кількість унітермів; $:$ - розділювач унітермів, який є розділювачем-комою, якщо впорядкованість унітермів немає значення (у цьому випадку операція секвентування є комутативною) або – крапкою з комою, якщо впорядкованість унітермів має значення (операція секвентування є некомутативною).

Рекурентна структурна декомпозиція системи генерування коду

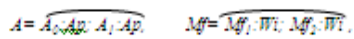
Формула (2) описує за критерієм функціонального призначення структурну декомпозицію першого рівня інформаційної технології генерування програмного коду з формул алгоритмів. У цій формулі T - унітерм (підсистема) призначений для задання переліку функцій них унітермів, які використовуються підсистемами

опрацювання операцій алгебри алгоритмів; *Ce* – унітерм опрацювання операції циклічного елімінування, *P* - унітерм опрацювання операції паралелення, *Cp* – циклічного паралелення, *U* - унітермів формул алгоритмів, *Cs* – циклічного секвентування, *E* – елімінування, *S* – секвентування, *A* - зв'язку з операційною системою, *Dcf* – задання режимів видалення циклічних операцій, *If* – задання режимів заміни, *Rf* – задання режимів підставляння, *Zgc* – генерування коду, *Cef* – задання форми циклічного елімінування, *Df* - задання форми доступу до ключів бази алгоритмів, *Mf* - головної форми, *Sf* - задання форми секвентування, *Z_s* - запису-зчитування, *Cpf* - задання форми циклічного паралелення, *Ef* - задання форми елімінування, *Mc* - оперування з графічними фігурами, *Sf* - задання форми секвентування, *Csf* - задання форми циклічного секвентування, *Ff* - задання шрифтів, *Md* – задання області ждля рисування, *Sa* – задання спеціальних символів унітермів, *Def* – задання форми видалення циклічних операцій, *F* – задання параметрів форми, *Pf* – задання форми операції паралелення, *Ug* – задання форми унітерму, *Df* – задання форми доступу до бази алгоритмів, *Icf* – задання форми заміни складових циклічних операцій, *Rf* – задання тмпів замін операцій, *Wf* – видалення графічних фігур, *oL* – зв'язку підсистеми оптимізації з редактором формул алгоритмів, *ooE* – оптимізація на основі властивостей елімінування, *oC* – оптимізація циклічних операцій, *oS* – оптимізація секвентування, *oP* – оптимізація паралелення, *oEo* -, *oGi* – генерування індексів XML-опису формул алгоритмів, *oDd* – задання унітермів-делегатів оптимізації, *oI1*, *oI2* – задання списків функційних унітермів оптимізації, *oBo* – базовий унітерм оптимізації, *oQ* – комунікаційний унітерм оптимізації, *oD* – оптимізації методом введення додаткової умови.



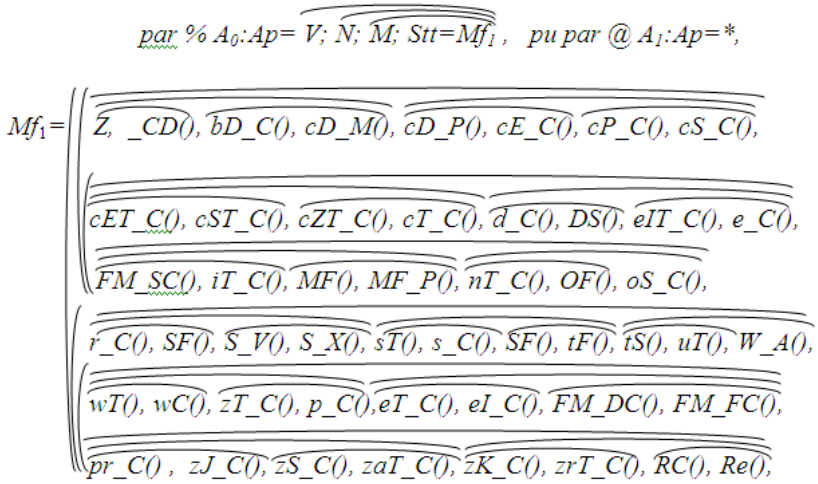
(2)

Другого рівня рекурентна декомпозиція унітермів *A* та *Mf* має такий вигляд



де A_0 та A_1 – підунітерми унітерма *A*, які наслідують (*:* - ознака наслідування) типовий унітерм *Ap*, що реалізується відомим класом Application [5, 6], Mf_0 та Mf_1 - підунітерми унітерма *Mf*, які наслідують відомий клас Window [5, 6].

Третього рівня рекурентна декомпозиція унітермів A_0, A_1, Mf_1 :



де *par* – ознака частини підсистеми, *%* - ідентифікатор опису графіки, *pu* - ознака загальнодоступності підсистеми, *@* - ідентифікатор функційної підсистеми, *Z* - унітерми-змінні, *_CD* - унітерм вибору унітермів-делегатів, *bD_C* - перевірки ключів доступу до бази алгоритмів, *cD_M* - вибору операції алгебри алгоритмів, *cD_P* - селекції унітермів, *cE_C* - операції циклічного елімінування, *cP_C cP_C* - циклічного паралелення, *cS_C* - циклічного секвентування, *cET_C* - вибору функційного унітерма циклічного елімінування, *cST_C* - вибору функційного унітерма циклічного секвентування, *cZT_C* - вибору функційного унітерма циклічного паралелення, *cT_C* – вибору шрифта і кегля, *d_C* – видалення операцій, *DS* – рисування абстрактного унітерма, *eIT_C* – вибору функційного унітерма експорту та імпорту до бази алгоритмів, *e_C* – опрацювання елімінування, *eT_C* – вибору функційного унітерма опрацювання елімінування, *eI_C* - експорту і імпорту до бази алгоритмів, *FM_DC* – подвійного цоку на головній формі, *FM_FC* – запису змін, *FM_SC* – задання початкових значень розмірів форми і сітки, *iT_C* – інформації про редактор формул алгоритмів, *MF* – задання початкових значень унітермів-змінних унітерму Mf_1, *MF_P* – вибору функційного унітерма рисування, *nT_C* – перерисування, *OF* – відкриття файла XML-опису формули алгоритму, *oS_C* – відкриття файла і задання кегля шрифта, *p_C* – опрацювання операції паралелення, *pr_C* – властивості операцій алгебри алгоритмів, *RC* – перевірки параметрів доступу до бази алгоритмів, *Re* - рисування початкового абстрактного унітерма, *r_C* - заміни унітермів, *SF* – запис формули у вигляді XML-опису, *S_V* – створення об'єкта відомого класу, *S_X* – задання смуг прокрутки, *sT* – вибору функційного унітерма опрацювання секвентування, *s_C* – функційний унітерм опрацювання секвентування, *SF* – функцій ний унітерм видалення графічних об'єктів, *tF* – обчислення розмірів абстрактного унітерму, *tS* – зчитування розміру кегля заданого шрифта , *uT* – вибору функційного унітерму видалення унітермів, *W_A* – вибору функційного унітерма рисування знаків операцій, *wT* – вибір функційного унітерма властивостей операцій алгебри алгоритмів, *wC* – функцій ний унітерм формування XML-опису доступу до бази алгоритмів, *zT_C* – закриття форми, *zJ_C* - вибір функційного унітерма запису файла із заданням його назви, *zS_C* - вибір функційного унітерма запису файла, *zaT_C* - вибір функційного унітерма заміни, *zK_C* - вибір функційного унітерма генерування коду, *zrT_C* – вибір функційного унітерма опрацювання операції паралелення.

Фрагмент рекурентної структурної декомпозиції Mf_0 описується такою формулою

$$Mf_0 = \left(F_0, F_1, F_2, \ldots, F_{24} \right.$$

де $F_0, F_1, F_2, \ldots, F_{24}$ – параметри унітермів, частково, F_{22}= *Load = "FM_SC"* у якій є відомою типовою властивістю Loaded [5, 6], а *FM_SC* - назва функційного унітерма опрацювання події *Load*.

Рекурентні структурні декомпозиції унітерму *U, Mc* та *Wf* описуються такими формулами:

$$pu @ \; U:T = \left(\overline{Zm, U(), CS(), CFC(), CXML(), Des(), Dra(), GFT(), GTH(),} \; \overline{GTL(), KF(), SF()} \right.,$$

$$pu @ \; Mc:Can = \left(\overline{Zc, AV(), DV(), GV(), GVC(), GVs(), GV\_C(), HTC(),} \; \overline{HTC\_C(), VCC, N()} \right.,$$

$$pu @ \; Wf = \overline{SF(), UF(), Wf()},$$

де *Zm* – унітерми-змінні, *U()* – функційний унітерм задання початкових значень, *CS()* – обчислення розмірів унітерма, *CFC()* – вибору унітерма (селекції), *CXML()* – створення –опису формули алгоритму, *Des()* – де селекції унітерма, *Dra()* – рисування унітерма, *GFT()* – форматування тексту, *GTH()* – форматування висоти тексту, *GTL()* – форматування довжини тексту, *KF()* – обчислення кількості фігур, *SF()* – видалення фігур, *Zc* – унітерми-змінні, *AV()* – доручення об'єкта, *DV()* – видалення графічних обєктів, *GV()* – пошуку попадань об'єктів у задану область, *GVC()* – видача обєкта за його індексом, *GVs()* – підготування параметрів і пошук попадань, *GV_C()* – перевірка часткового попадання графічних обєктів у виділену область, *HTC()* – перевірка ступеня попадання об'єкта у виділену область, *HTC_C()* - перевірка ступеня попадання багатьох об'єктів у виділену область, *VCC* – унітерм - властивістьвидачі кількості фігур, *N()* – видачі об'єкта за його номером, *SF()* – видалення графічних обєктів, *UF()* – підготування до видалення графічних об'єктів, *Wf()* – задання початкових значень, *Can* – унітермБ який реалізується відомим графічний елементом Canvas [5, 6] на якому відображаються інші графічні об'єкти.

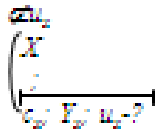
Функціональна декомпозиція

Функціонування інформаційних технологій і систем починається з вихідної точки. Нею, наприклад, можуть бути ввімкнення системи чи запуск на виконання. У будь-якому випадку має бути виконана умова (u) початку функціонування. Якщо умова не виконана, то відбувається очікування її виконання, що запишемо поверненням у цикл (c_u) за умовою u. Коли ж умова виконана, то функціонування починається з унітерму *P*. У вигляді формули процес очікування виконання умови опишемо формулою

Нехай унітермом *P* вибирається унітерм *A*.

Далі розглянемо побудову моделі у якій усі унітерми (*A, B, C, D, …*) мають вибір самих себе (є рекурсивними) і мають вибір кожного із унітермів та кожен з яких може бути вибраним всіма іншими. Опис вибору всіх всіма унітермами складається з двох частин. Перша частина утворена елімінуваннями усіх циклічних секвенткванв від унітерма *A* до всіх решти унітермів (*B, C, D, …*). Ці циклічні секвентування запишемо як u_{ab}, u_{ac}, u_{ad}, …, а їх елімінування запишемо формулою

Рекурентність унітермів опишемо формулою

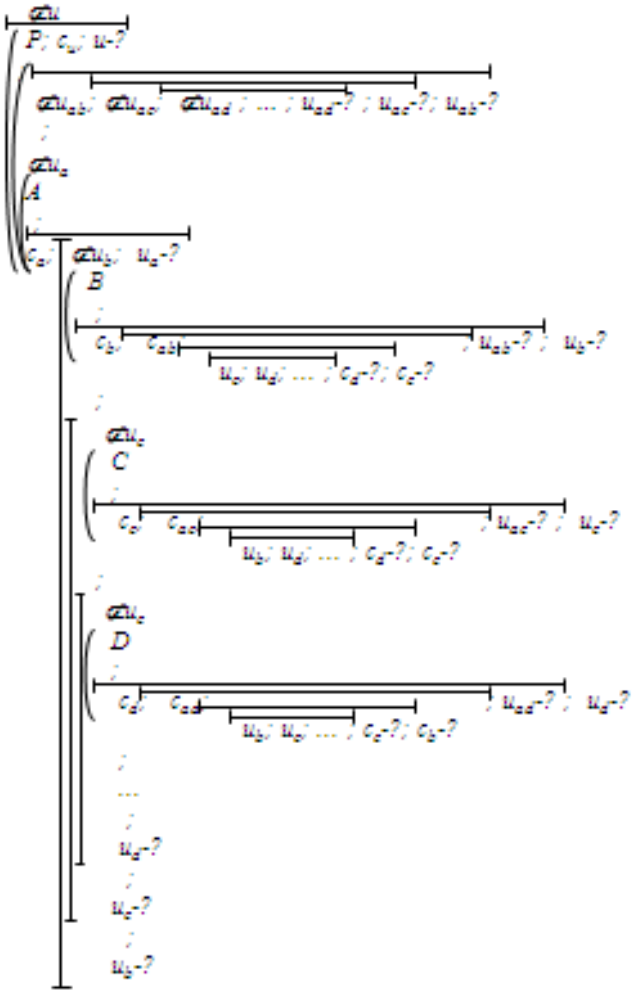


де *X* є унітермами *A, B, C, D*, u_x - умова циклу, c_x - ознака повернення у цикл за умовою u_x, Y_x - продовження алгоритму. Загальна формула функціональної декомпозиції описана формулою (3).

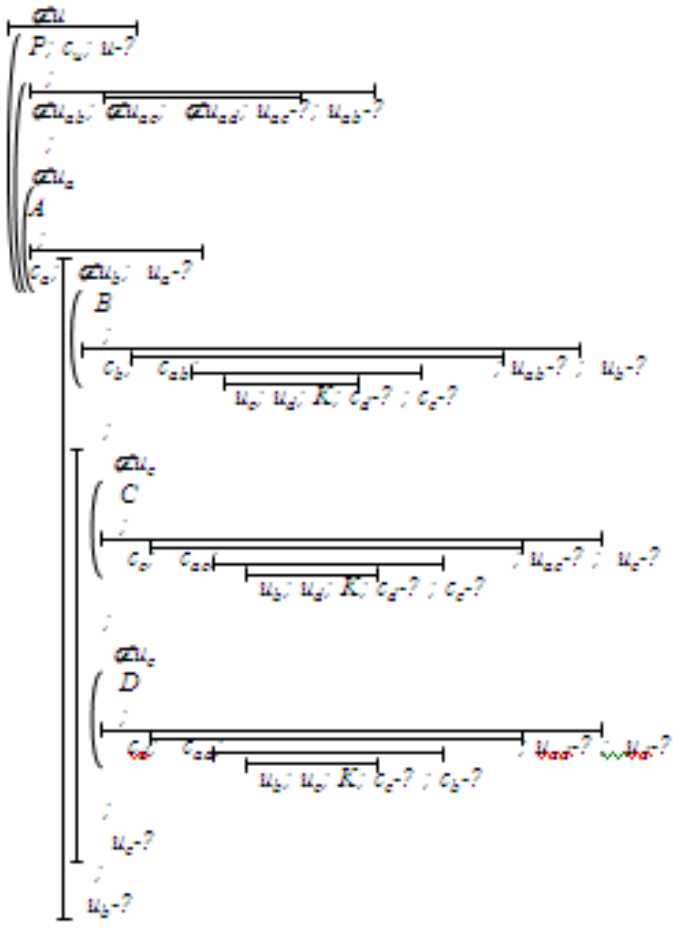
Із формули (3) для унітермів *A, B, C, D* отримаємо модель функційної декомпозиції описану формулою (4).

Рекурентна функціональна декомпозиція системи генерування коду

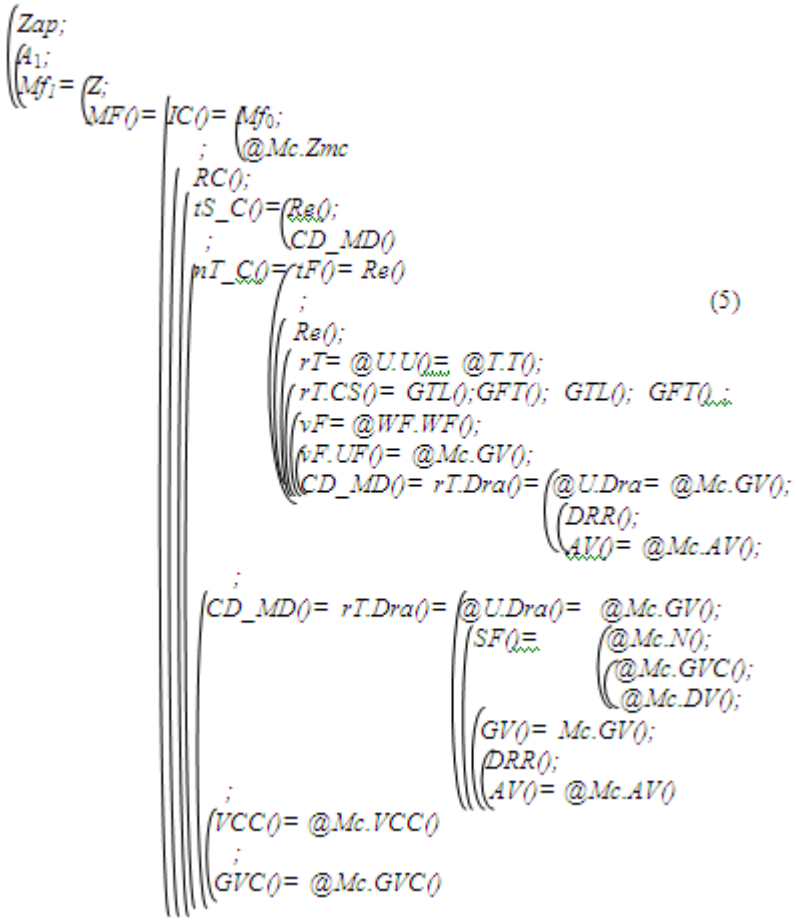
Початок функціонування системи генерування программного коду описується формулою (5). У ній *Zap* - унітерм запуску функціонування системи, *IC()* – функційний унітерм, який реалізується відомою процедурою InitializeComponent() [5, 6], призначеною для ініціалізації елементів графічного інтерфейсу системи, *@Mc.Zmc* – унітерми-змінні (*@Mc.Zmc*) підсистеми *Mc*.



(3)



(4)



(5)

Результатом роботи програмної реалізації рекурсивної функціональної моделі інформаційної технології генерування програмного коду з формул алгоритмів є графічне вікно інтерфесу користувача, яке показаноно на рисунку.

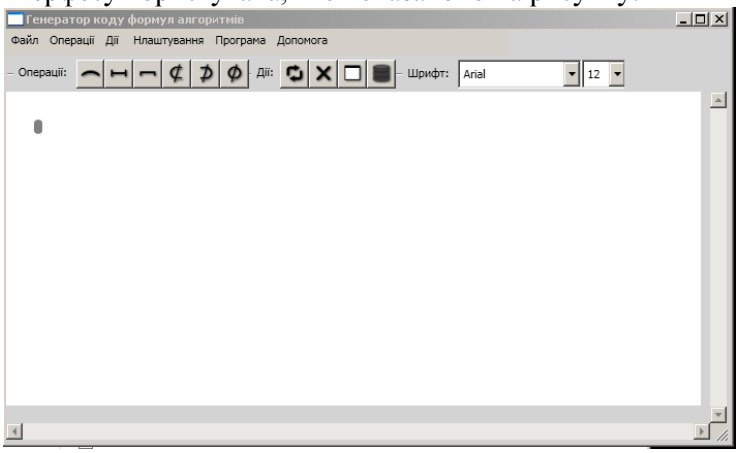


Рис.1. Графічне вікно інформаційної технології генерування програмного коду з формул алгоритмів.

Висновки

1. Створеною рекурентно-декомпозиційною методологією декомпозиційні моделі інформаційних технологій і систем описуються формулами над якими, використовуючи властивості розширеної алгебри алгоритмів, можуть бути виконані тотожні перетворення з метою оптимізації за кількістю унітермів і операцій.

2. Рекурентно-декомпозиційна методологія може бути застосована для побудови рекурентних декомпозиційних моделей різного виду (структурних, функціональних та інших).

3. Реалізація рекурентних декомпозиційних моделей інформаційної технології генерування програмного коду з формул алгоритмів забезпечує створення графічного інтерфейсу користувача.

1. Великий тлумачний словник сучасної української мови / Уклад. і голов. ред. В.Т.Бусел. – К.; Ірпінь: ВТФ «Перун», 2002. – 1440 с.

2. Математическая энциклопедия / Гл. ред. И.М.Виноградов. –М: Советская Энциклопедия. Т. 4 Ок – Сло. 1984. – 1216 стб., ил.

3. http://en.wikipedia.org/wiki/Decomposition_(computer_science).

4. W.Owsiak, A.Owsiak. Rozszrzerzenie algebry algorytmów //Pomiary, automatyka, kontrola 2, 2010. –S.184-188.

5. C. Petzold. Programowanie Windows w języku C#. –Warszawa: „RM", 2002. – 1161 s.

6. Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C# 2008. – Москва, Санкт-Петербург,Киев: "Apress", 2008. – 922 с.