В. Овсяк. Загальна модель редактора графічних унітермів / В. Овсяк, М. Козелко // Вісник ТНТУ. — 2013. — Том 69. — № 1. — С.183-192. — (приладобудування та інформаційно-вимірювальні технології).

УДК 004.4:232

В. Овсяк, докт. техн. наук; М. Козелко

Українська академія друкарства

ЗАГАЛЬНА МОДЕЛЬ РЕДАКТОРА ГРАФІЧНИХ УНІТЕРМІВ

***Резюме**. Застосовуючи методи декомпозиції і модифікованої алгебри алгоритмів створено й описано загальну модель редактора проектування багатотипних графічних інтерфейсів користувача інструментальних засобів комп'ютерних систем. Виконано декомпозицію за критерієм функціонального призначення підсистем. Математичну модель інструментальних засобів проектування графічних інтерфейсів користувача описано засобами модифікованої алгебри алгоритмів. Побудована математична модель програмно реалізована й апробована.*

***Ключові слова:** декомпозиція, модифікована алгебра алгоритмів, унітерм, абстракція, графічний інтерфейс, модель, система, проект.*

V. Ovsyak, M. Kozelko

GENERAL MODEL OF EDITOR GRAPHIC UNITERMS

***Summary.** A general editor model of designing graphic user interfaces tools computer systems was created and described by applying the method of decomposition and modified algebra algorithms. Decomposition has been done by the criterion of functional subsystems destination. Mathematical model of instrumental ways of designing graphic user interfaces is described by means of the modified algebra algorithms. The mathematical model is implemented and tested using programming.*

Modern software development systems must meet certain conditions regarding quick and efficient development of software packages and systems. Editor GUIs is one of the main parts of these systems, which should ensure graphic user's interface of computer systems tools, and which also should be most easy to use, intuitional clear and functional. Nowadays, when the functional peculiarities of websites or web projects almost are not worse than the desktop systems and despite of the fact that modern systems of developing and designing GUIs make possible to create projects and to use graphic elements (graphic uniterms of only certain types: buttons, text fields, elements of alternative and non-alternative choice, etc.), there is a problem of creating the editor, which would realise the possibility of developing only one GUI for many types of projects (editor user interfaces of computer systems).

Graphic uniterms editor is a computer system of projecting graphic interfaces of computer system and providing them with functional properties. Graphic interface is characterized by an aggregate of facilities for processing and retrieving of information, maximally adapted for the user's convenience. The user's interface is implemented by a single window or multiwindow mode, change of the colour, size, visibility (transparency, translucency, invisibility) of windows and their elements, their location, sorting of window elements, flexible configuration of windows and some of their elements (files, folders, labels, fonts, etc.) in graphical systems. To build the most easy to use and optimally informative graphic interface modern graphical interface, editors are needed, with the help of which the development of graphic interface would be as quick as possible with the ability of visual location and positioning of interface elements, display and editing of all graphic interface properties as a whole and each element separately, quick display of the result of development, implementation of the graphic interface's response to user actions.

***Key words:** Decomposition, modified algebra of algorithms uniterm, abstract, graphical interface, model, system, design.*

Постановка проблеми. Аналіз останніх досліджень і публікацій. Сучасні системи розроблення програмного забезпечення [1, 2, 3] мають відповідати жорстким

умовам щодо швидкого та ефективного розроблення програмних комплексів та систем. Однією із основних частин цих систем є редактор графічних інтерфейсів, котрий має забезпечувати розроблення максимально зручних у використанні, інтуїтивно зрозумілих і функціональних графічних інтерфейсів користувача інструментальних засобів комп'ютерних систем. На сьогодні, коли функціональні особливості Інтернет-сайтів чи веб-проектів мало чим поступаються настільним системам [4, 5, 6], а також зважаючи на те, що сучасні ситеми розроблення й проектування графічних інтерфейсів надають можливість створювати проекти й використовувати графічні елементи (графічні унітерми) тільки строгих окреслених типів (кнопки, текстові поля, елементи альтернативного і безальтернативного вибору та інші), існує проблема створення редактора, котрий би реалізував можливість розроблення одного графічного інтерфейсу для багатьох типів проектів (редактор багатотипних інтерфейсів користувача комп'ютерних систем).

Редактор графічних унітермів – це комп'ютерна система проектування графічних інтерфейсів комп'ютерних систем та задавання їх функціональних властивостей. Характеризується графічний інтерфейс сукупністю засобів для опрацювання та відображення інформації, максимально пристосованих для зручності користувача. У графічних системах інтерфейс користувача реалізовується одновіконним чи багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон та їх елементів, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштовуваннями як самих вікон, так і окремих їхніх елементів (файли, папки, ярлики, шрифти тощо). Для побудови максимально зручного в користуванні й оптимально інформаційного графічного інтерфейсу потрібні сучасні редактори графічних інтерфейсів, з допомогою котрих розроблення графічного інтерфейсу проходило б максимально швидко, з можливістю наочного розташування й позиціювання елементів інтерфейсу, відображення та редагування всіх властивостей графічного інтерфейсу в цілому й окремо кожного елемента, миттєве відображення результату розроблення, реалізації реакції графічного інтерфейсу на дії користувача.

Метою роботи є створення загальної математичної моделі редактора багатотипних графічних інтерфейсів користувача, яка б забезпечувала можливість розроблення одного графічного інтерфейсу, придатного для багатьох типів проектів.

Завдання дослідження. Використовуючи методи декомпозиції й розширеної алгебри алгоритмів, створити декомпозиційну модель редактора багатотипних графічних інтерфейсів користувача, яку запрограмувати мовою C# платформи Microsoft Visual Studio. NET [5, 6].

Редактор графічних унітермів (умовно позначим *S*) можна декомпозувати на кілька підсистем, незалежні одна від одної, хоча при розробленні графічного інтерфейсу використовуються здебільшого всі підсистеми. Модель редактора графічних унітермів декомпозуємо на підсистеми за ознакою функціонального призначення, а саме: створення унітерма-форми інтерфейсу (*A*), генерування графічних унітермів інтерфейсу (*B*), опрацювання дій над графічними унітермами та надання їм функціональних особливостей (*C*), конвертації абстрактних унітермів до типових унітермів графічного інтерфейсу (*D*), генерування створеного проекту до Windows- чи

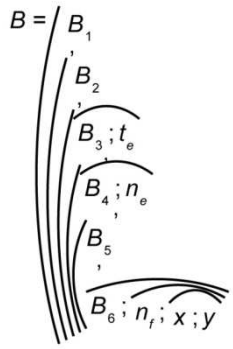
Web-проекту (*E*), компіляції створеного проекту (*F*). Підсистема *A* – створення унітерма-форми призначена для генерування одного або кількох (у випадку розроблення багатовіконного проекту) унітермів форми, котрі є основою графічного інтерфейсу і на котрих розміщуються всі потрібні графічні унітерми. Підсистема *B* застосовується для генерування графічних унітермів обраного типу та розташування їх на унітермі-формі. Для редагування властивостей графічного унітерму чи властивостей унітерма-форми, а також наданя функціональних властивостей унітермам у відповідь на дії користувача використовується підситема *C*. Конвертація абстрактних унітермів до типових унітермів відбувається у підсистемі *D*. Перед компіляцією готового проекту його потрібно зберегти для подальшого чи повторного використання, а також вказати тип згенерованого проекту. Підситема *E* призначена для забезпечення цих можливостей. Компіляція готового проекту обраного користувачем типу реалізується підсистемою *F*. Модель редактора графічних унітермів можна описати формулою (1) модифікованої алгебри алгоритмів [7 – 11] у вигляді

$$S = \overset{\frown}{A, B, C, D, E, F} \tag{1}$$

Підсистема *A* створює новий унітерм-форму. Підсистема містить графічний унітерм A_1, з вибором котрого відбувається ініціалізація створення нового унітерма-форми. Зважаючи на те, що користувач може створити кілька утітермів-форм, реалізуючи тим самим багатовіконний проект, кожному створеному унітерму-формі задається назва унітерму (n_f). Назва унітерму-форми n_f потрібна для ідентифікації унітерма-форми при розміщенні на ньому графічних унітермів, а також формування ієрархічного дерева відображення унітермів-форм, тобто послідовності виклику та відображення унітермів-форм у багатовіконному проекті. Для задавання назви n_f призначений унітерм-форма A_2, відображення котрого відбувається після вибору користувачем графічного унітерму A_1. Наступним елементом підсистеми створення унітерму-форми є функціональний унітерм A_3, в котрому відбувається безпосереднє створення унітерма-форми. Функціональний унітерм A_3 характеризується властивостями *h* і *w*, потрібними для задавання розмірів унітерма-форми, висоти і довжини відповідно. Після створення унітерма-форми вона розташовується і відображається в редакторі графічних унітермів. Відображення унітерма-форми реалізується функціональним унітермом A_4. Математичну модель підсистеми створення унітерма-форми описано формулою (2) модифікованої алгебри алгоритмів.

$$A = \overset{\frown}{A_1, A_2; n_f, A_3; h, w, A_4} \tag{2}$$

Підсистема генерування графічного унітерма складається з кількох функціональних унітермів, описаних формулою алгебри алгоритмів (3).

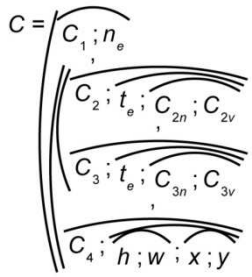


(3)

До складу підсистеми *B* належить перелік графічних унітермів B_1. Перелік складається з усіх доступних графічних унітермів визначеного типу, а також з унітерма абстрактного типу. Перелік графічних унітермів B_1 призначений для вибору графічного унітерма обраного типу та подальшого його генерування й розміщення на унітермі форми. Функціональний унітерм B_2 застосовується для генерування абстрактного графічного унітерму. Унітермом B_2 створюється екземпляр абстрактного унітерма, задавання початкових значень властивостям унітерма: висоти, ширини, кольору та ін. Генерування графічного унітерма відомого типу реалізується функціональним унітермом B_3. Принцип побудови графічного унітерма такий, як і при побудові абстракного унітерма. Однак існує одна відмінність – функціональним унітермом B_2 генерується тільки абстрактний унітерм, а функціональному унітерму B_3 задається тип графічного унітерма t_e і відповідно до значення t_e створюється екземпляр графічного унітерма заданого типу. Значення типу унітерма t_e визначається при виборі користувачем з переліку графічних унітермів B_2 унітерма потрібного типу. Для реалізації можливості зміни властивостей конкретного графічного унітерма і задавання їм методів опрацювання подій користувача, кожен графічний унітерм повинен мати унікальну назву n_e для його ідентифікації. Назва графічного чи абстракного графічного унітерма задається в унітермі форми B_4, відображення котрого відбувається при виборі унітерма з переліку B_1. Функціональний унітерм B_5 задає створеному графічному унітерму функцію вибору його користувачем для задавання чи редагування властивостей унітерма. Розміщення створеного графічного унітерма реалізується функціональним унітермом B_6. Унітерм B_6 характеризується трьома ознаками: назвою унітерма-форми n_f та координатами розміщення *x* і *y* графічного унітерма на унітермі-формі. Назва n_f – це назва активного унітерма-форми, на якому повинні розташовуватися створені графічні унітерми. Оскільки користувач може генерувати кілька унітермів-форм, така реалізація підсистеми і зокрема функціонального унітерма B_6 забезпечує вірне розміщення створеного графічного унітерма саме на потрібному унітермі-формі.

Створені графічні унітерми генеруються з параметрами на замовчування, тобто з початковими значеннями властивостей, а також не мають ніякого функціонального наповнення – не реагують на дії користувача над ними. Графічний унітерм генерується з метою реалізації певного завдання визначеним розробником проекту. Це може бути реакція графічного унітерма на дії користувача, що призведе до виклику методу опрацювання чи відображення певної інформації, або ж реакція унітерма внаслідок опрацювання інформації іншими графічними унітермами. Підсистема *C* призначена

саме для налаштування створеного графічного унітерма до потреб розробника графічного інтерфейсу, редагування потрібних значень властивостей унітерма, задавання опрацьовувачів подій користувача чи зміни стану самого графічного унітерма. Математично підсистема *C* описується формулою (4) модифікованої алгебри алгоритмів.



(4)

Після вибору графічного унітерма на унтітермі-формі користувачем графічний унітерм опрацьовується пісистемою *C*. Функціональним унітермом C_1 визначається власна назва графічного унітерма n_e з метою застосування змін властивостей гафічного унітерма саме до обраного графічного унітерма. Функціональним унітермом C_2 визначається тип обраного графічного унітерма t_e Визначення типу t_e необхідне для визначення всіх властивостей, притаманних обраному графічному унітерму. На основі типу обраного графічного елемента формується перелік усіх доступних властивостей унітерма C_{2n}, а також перелік C_{2v} зі значень відповідних властивостей із переліку C_{2n}. Значення переліку C_{2v} доступні для редагування. При зміні значень властивостей в переліку C_{2v} автоматично відбувається застосування нових значень властивостям обраного графічного унітерма. Задавання функціональних особливостей графічному унітерму відбувається схоже до редагування властивостей графічного унітерма. Функціональний унітерм C_3 визначає тип обраного графічного унітерма t_e, формується перелік C_{3n} усіх доступних дій над графічним унітермом обраного типу t_e, а також перелік C_{3v} полів вводу назви методів опрацювання відповідних подій з переліку C_{3n}. Після вводу назви методу в поле обраної події цей метод автоматично закріплюється за обраною подією графічного унітерма. Таким чином графічному унітерму задаються функціональні особливості – при настанні певної події відбувається звернення до відповідного методу, у якому опрацьовується певна інформація. Окремі властивості графічного унітерма (такі, як висота, ширина, координати розміщення на унітерміформі) підсистема *C* надає змогу редагувати, проводячи дії зміни розміру й положення графічного унітерма з допомогою миші, застосовуючи ці дії безпосередньо до обраного графічного унітерма. Такий метод швидше реалізує зміну розмірів і положення графічного унітерма, оскільки не потрібно звертатися до переліку властивостей унітерма із десятка всіх доступних властивостей шукати потрібну. Даний метод реалізується функціональним унітермом C_4. Характеризується унітерм C_4 властивостями висоти графічного унітерма *h*, ширини *w*, відстані, на якій розміщується графічний унітерм на унітермі-формі чи іншому графічному унітермі від лівого краю *x*, і верхнім відступом *y*.

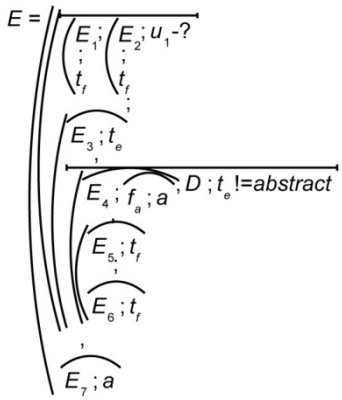
Абстрактні унітерми користувач може конвертувати до типових графічних унітермів. Підсистема *D* призначена для конвертації абстрактних графічних унітермів.

Після вибору графічного унітерма функціональним унітермом D_1 відбувається визначення типу обраного графічного унітерма. Далі підсистемою D відбувається перевірка умови належності типу обраного графічного унітерма до абстрактного типу t_e=*abstract*, конвертація обраного елемента можлива тільки при виконанні умови t_e=*abstract*. Всі графічні унітерми умовно можна розділити на дві групи – графічні унітерми, на котрих можуть розміщуватись інші графічні унітерми (контейнерні), та графічні унітерми, на котрих неможливо розташовувати інші (дочірні унітерми). Зважаючи на таку особливість графічних унітермів і на те, що користувач має змогу розташувати на абстрактному графічному унітермі інші графічні унітерми, функціональним унітермом D_2 відбувається визначення кількості дочірніх графічних унітермів k. Залежно від значення кількості дочірніх графічних унітермів, що розташовані на обраному абстрактному графічному унітермі, відбувається формування доступних типів графічних унітермів, до яких абстрактний графічний унітерм може бути конвертований. Якщо $k=0$, тоді формується перелік усіх можливих типів графічних унітермів D_3 для конвертації абстрактного графічного унітерма. Коли ж на абстрактному графічному унітермі розміщуються інші дочірні унітерми відносно абстрактного унітерма, тобто $k>0$, формується перелік D_3 тільки з унітермів контейнерного типу. При виконанні всіх описаних вище умов функціональним унітермом D_5 відбувається конвертація абстрактного унітерма до типового графічного унітерма. Математичний опис підсистеми D засобами модифікованої алгебри алгоритмів описаний формулою (5)

$$D = \left(D_1 ; t_e \right)' \not\subset t_e = abstract \left(D_2 ; k \right)' \left(\not\subset k=0 \; D_3 \right)' \left(\not\subset k>0 \; D_4 \right)' D_5 \quad (5)$$

Усі властивості та опрацьовувачі подій, задані абстрактному унітерму автоматично підсистемою D, присвоюються конвертованому графічному унітерму.

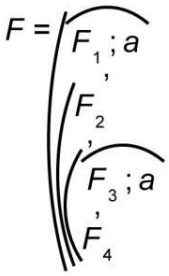
Створений з допомогою редактора графічних унітермів проект дає можливість зберегти у файловій системі персонального комп’ютера для подальшого компілювання у готову комп’ютерну систему. Підсистема E надає можливість зберегти розроблений проект як настільну комп’ютерну систему (комп’ютерна система, яка розміщується у файловій системі ПК та працює у середовищі операційної системи цього ПК), або як веб-систему (система, яка розміщується на віддаленому сервері, а звертання до неї відбувається через систему «переглядач» використовуючи для зв’язку мережу Інтернет). Математична модель такої підсистеми описана формулою (6) модифікованої алгебри алгоритмів.



(6)

Ініціалізація генерування і збереження проекту відбувається підсистемою *E* після вибору користувачем відповідного графічного унітерма збереження проекту. Оскільки користувачу доступно для збереження два типи проекту – настільна і веб-система, то відбувається перевірка умови вибору користувачем потрібного типу системи – умовою u_1. Якщо користувач вибрав збереження проекту як настільної системи, тоді функціональний унітерм E_1 ініціалізує генерування потрібних файлів для збереження системи настільного типу. Також змінному унітерму t_f задається значеня типу проекту, обраного користувачем для зберігання, тобто настільна система. В іншому випадку функціональний унітерм E_2 ініціалізує генерування файлів, потрібних для збереження веб-системи, а унітерму t_f задається значення типу веб-системи. Функціональний унітерм E_3 призначений для визначення типу t_e усіх графічних унітермів, розміщених на створених унітермах-формах. Зважаючи на те, що перед генеруванням розробленого проекту в одну із систем усі абстрактні унітерми повинні бути конвертовані до типових графічних унітермів, проводиться перевірка належності типу t_e абстрактному типу (умова t_e*!=abstract*). Якщо хоча б один із графічних унітермів належить до абстрактного типу t_e*=abstract*, тоді відбувається виклик підсистеми *D* для конвертації абстрактного унітерма до типового графічного унітерма. При виконанні умови t_e*!=abstract* функціональним унітермом E_4 задається шлях збереження проекту *a* у файловій системі персонального комп'ютера. Шлях збереження проекту *a* задається користувачем при відображенні унітерма-форми збереження проекту f_a. Функціональний унітерм E_4 призначений для генерування текстового опису унітермів-форм проекту та їх дочірних графічних унітермів для подальшого запису цього опису у файл. Генерування текстового опису унітермів форми відбувається залежно від значення унітерма t_f . У текстовому описі вказується назва та тип кожного з графічних унітермів, їх властивості й значення, а також опис усіх подій графічних унітермів з посиланням на метод опрацювання цих подій. Опис функціональної частини створеного проекту відбувається функціональним унітермом E_6. Функціональний опис також формується залежно від типу t_f. Після успішного опису всіх графічних унітермів та їх функціональних особливостей відбувається запис функціональним унітермом E_7 опису у файл. Запис відбувається у файловій системі ПК за вказаним шляхом збереження *a*. Особливістю створеного проекту з допомогою редактора графічних унітермів є можливість генерування створеного проекту у системи кількох типів (настільні системи та веб-системи).

Згенеровану й збережену комп'ютерну систему користувач має змогу скомпілювати і запустити на виконання. Компіляція проекту проводиться підсистемою *F*. Підсистема *F* математично описана формулою (7) модифікованої алгебри алгоритмів.



(7)

Функціональним унітермом F_1 відбувається звернення до файлів збереженого проекту, котрі знаходяться у файловій системі за шляхом *a*. Опрацювання обраних файлів проводиться функціональним унітермом F_2. Відбувається визначення типу збереженого проекту, генерування всіх потрібних графічних форм та надання їм функціональних особливостей. Компіляція проекту проводиться функціональним унітермом F_3. Скомпільований проект зберігається у файловій системі персонального компютера за шляхом *a*. Після успішної компіляції проекту скомпільована система запускається на виконання функціональним унітермом F_4. Такий підхід надає можливість користувачу відразу отримати результат своєї роботи.

Наступний фрагмент програмного коду реалізує генерування графічного елемента (графічного унітерма) типу кнопки, а також метод опрацювання події натискання кнопки користувачем. Після вибору користувачем генерування графічного унітерма типу кнопки викликається метод Add But (double x, double y) з параметрами розміщення елемента x і y, котрий створює новий екземпляр елемента типу кнопки childbutton = new Button(). Щойно згенерованому елементу задаються початкові параметри висоти та ширини childbutton. Height = 20, childbutton.Width = 80 і координати розміщення його на формі Canvas.SetLeft (childbutton, x), Canvas.SetTop (childbutton, y). Викликається метод ElementName (childbutton) для задання назви кнопки, а також задання тексту котрий на ній відображається childbutton.Content = "Button"+bn.ToString(). Щоб забезпечити можливість вибору кнопки користувачем їй задається подія цоку по ній childbutton.Click += new RoutedEventHandler(childbutton_Click) настання котрої призводить до виконання методу childbutton_Click(). Виконання методу реалізує відображення елемента зміни розмірів і положення кнопки re = new ResizeElement(but, but.Parent), відображення переліку властивостей та їх значень Eventlist(but), а також переліку подій і методів їх опрацювання propertyList(but).

```
private void Add But (double x, double y)
    {
      childbutton = new Button();
      childbutton. Height = 20;
      childbutton. Width = 80;
```

```
            Element Name (childbutton);
            childbutton. Content = "Button" + bn. To String();
            childbutton. Vertical Alignment = Vertical Alignment. Center;
            childbutton. Horizontal Alignment = Horizontal Alignment. Center;
            Canvas. Set Left (childbutton, x);
            Canvas. Set Top (childbutton, y);
            childbutton.Vertical Alignment = Vertical Alignment. Center;
            childbutton. Horizontal Alignment = Horizontal Alignment. Center;
            bn++;
            contr = childbutton;
            oblist. Add (childbutton);
childbutton. Click += new Routed Event Handler (childbutton_Click);
            Create Item (childbutton);
            Sorted List sl = new Sorted List();
            EventL. Add (childbutton. Name, sl);
        }
        void childbutton_Click(object sender, RoutedEventArgs e)
        {
            Button but = (Button) sender;
            Eventlist (but);
            property List (but);
            remov Res();
            re = new Resize Element (but, but. Parent);
            Readevent();
            Contr = but;
        }
```

Результати дослідження. Засобами модифікованої алгебри алгоритмів побудовано загальну математичну модель декомпозиції редактора з абстрактними графічними елементами – унітермами, призначеним для проектування інструментальних засобів комп’ютерних систем. Створену математичну модель програмно реалізовано й апробовано.

Висновки. Використання редактора грфічних унітермів забезпечує розроблення сучасного графічного інерфейсу, котрий може використовуватись як і в Інтернет-проектах, так і настільних системах. Забезпечує наочне розташування й позиціювання елементів інтерфейсу, відображення та редагування усіх властивостей графічного інтерфейса в цілому й окремо кожного елемента, миттєве відображення результату розроблення, а також використання унітермів абстрактного типу.

Використання абстрактних унітермів на етапі розроблення інтерфейсу надає можливість відмовитися від строго визначених графічних елементів-унітермів. Такий підхід є універсальнішим, гнучкішим, передбачає повторне використання спроектованих інтерфейсів для різних типів проектів, таких, як Windows програма чи Інтернет-сторінка, а також в інших проектах. Таким чином використання абстрактних унітермів дає більші можливості при розробленні інтерфейсів та зменшує затрати часу на його розроблення.

Conclusions. The use of graphic editor's uniterms provides the development of modern graphic interface, which can be used in online projects and in desktop systems. It provides obvious location and positioning of the interface elements, displaying and editing of all

properties of graphic interface as a whole and each element separately, instant displaying the result of development and using of abstract uniterms.

Using the abstract uniterms during the interface development provides an opportunity to refuse from the well- defined graphic elements – uniterms. This approach is more universal, more flexible, and allours the repeated use of the designed interfaces for different types of projects, such as Windows program or web page as well as in other projects. Thus the use of abstract uniterms provides better possibilities at interface's development and reduces the costs of time for its development.

Список використаної літератури

1. Шилдт, Г. C 4.0: полное руководство [Текст] / Г. Шилдт. – Москва: Вильямс, 2011. – 1056 с.
2. Троэлсен, Э. Язык программирования C#2010 и платформа. NET 4.0 [Текст] / Э. Троэлсен. – Москва: Вильямс, 2010. – 1392 с.
3. Petzold, C. Programowanie Windows w języku C / C. Petzold. – Warszawa: Rm, 2002. – 1161 s.
4. Powers, L. Microsoft Visual Studio 2005 Księga eksperta / L. Powers, M. Snell. – Gliwice: HELION, 2007. –840 p.
5. Троэлсен, Э. Microsoft ASP. NET 3.5 с примерами на C с примерами для профессионалов [Текст] / Э. Троэлсен, М. Шпушта. – Москва: Вильямс, 2008. – 1424 с.
6. Мак-Дональд, М. WPF: Windows Presentation Foundation в NET 3.5 с примерами на C 2008 для профессионалов [Текст] / М. Мак-Дональд. – Москва: Вільямс, 2008. – 928 с.
7. Ovsyak, A. The extended algebra of algorithms width multiconditional elimination / A. Ovsyak, V. Ovsyak // Вісник Національного університету «Львівська політехніка». – Львів: Видавництво Львівської політехніки, 2010. – № 672. – С. 291 – 300.
8. Овсяк, О. Розширена алгебра алгоритмів і модель зв’язку між класичною і розширеною операцією елімінування [Текст] /О. Овсяк // Комп’ютерні технології друкарства. – Львів: Українська академія друкарства, 2010. – № 23. – С. 45 – 53.
9. Owsiak, A. Rozszrzerzenie algebry algorytmów / A. Owsiak, W.Owsiak // Pomiary, automatyka, kontrola, 2010. – № 2. – P. 184 – 188.
10. Овсяк, О.В. Розширення алгебри алгоритмів аксіомами операцій циклів [Текст] /О.В. Овсяк // Вісник Національного університету «Львівська політехніка» – Львів: Видавництво Львівської політехніки, 2010. – № 685. – С. 12 – 20.
11. Ovsyak, A. The extended algebra of algorithms with additional cycle elimination axiomats / A. Ovsyak, V. Ovsyak // Conference “Intelligent Information and Engineering Systems” – Poland: Polańczyk, 2011. – P. 23 – 34.

Отримано 12.11.2012