УДК 004
© В. Овсяк[1], М. Козелко[1] , 2012

МОДЕЛЬ АБСТРАКТНИХ ГРАФІЧНИХ ІНТЕРФЕЙСІВ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

Описана модель створення абстрактних графічних інтерфейсів інформаційних технологій і систем, наведено приклад її використання і фрагмент програмної реалізації.

The model of abstract graphical interface of information technologies and systems have been described , an example of its usage and selection of program implementation showed.

1. ВСТУП І ФОРМУЛЮВАННЯ ЗАДАЧІ

Сучасні системи розробки і проектування графічних інтерфейсів інформаційних технологій і систем, типовим представником яких є платформа Microsoft Visual Studio.NET [1 – 5], використовують графічні елементи тільки строго окреслених типів. Перш за все окреслюється тип проекту для якого має бути створено інтерфейс користувача. Наприклад, ним може бути Windows чи Web проект. Для кожного із них платформою генерується специфічне вікно. На згенероване вікно методом вибору і перетягування вставляються конкретні графічні елементи (кнопки, мітки, елементи альтернативного і безальтернативного вибору та інші). Задаються властивості встановлених на формі графічних елементів.

Для текстового опису графічних елементів і програмування комп'ютерної графіки розроблена мова розмітки XAML [1, 6]. Код форми і розташованих на ній графічних елементів генерується платформою автоматично. Є можливість зміни коду і його доповнення, наприклад, заданням методів опрацювання подій.

На функціональному рівні сучасними інформаційними технологіями і системами використовуються абстрактні і віртуальні методи [1 – 6]. Їх застосування забезпечує певну структурованість та універсалізацію програмного коду. Однак сучасними технологіями формування графічних інтерфейсів не використовуються абстрактні графічні елементи. У даній роботі власне вводиться модель абстрактних графічних елементів, призначених для проектування інструментальних засобів інформаційних технологій і систем.

[1]Українська академія друкарства

2. МОДЕЛІ АБСТРАКТНИХ ГРАФІЧНИХ ЕЛЕМЕНТІВ – УНІТЕРМІВ

Абстрактний графічний елемент характеризується такими ознаками ознаками, як властивості та події. До властивостей відносяться всі властивості притаманні графічним елементам такі, як висота, ширина, назва, прозорість, активність та інші. Події представляють методи опрацювання дій користувача, наприклад цок кнопкою миші на елементі, натиснення кнопки на клавіатурі.

Після етапу проектування інтерфейсу абстракний графічний елемент (унітерм) можливо конвертувати у будь-який типовий графічний елемент застосувавши до нього всі властивості і події абстрактного унітерма.

Блок-схема процесу творення інтерфейсу користувача з абстрактними унітермами зображено на рис.1.

Рис. 1. Блок-схема процесу формування інтерфейсу користувача з використанням абстрактних графічних унітермів (елементів)

Перш за все користувачем створюється новий проект, основою для побудови інтерфейсу якого є абстрактний унітерм з назвою Form1. На рис. 2 зображено графічний вид абстрактного унітерма на рівні проекту.

Рис. 2 Графічний абстрактний унітерм проекту

Далі користувач проектує свій інтерфейс з використанням графічних абстрактних чи предметних графічних елементів – унітермів. На рис. 3 наведено графічний інтерфейс, який утворено абстрактним графічним елементом з навою Abstract0, двома кнопками (з назвами Button1 і Button2), елементом альтернативного вибору CheckBox4 та абстрактним унітермом у вигляді білого прямокутника.

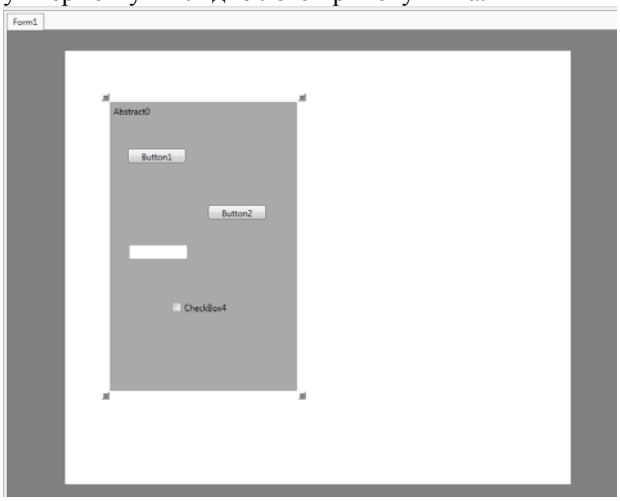

Рис. 3. Зображення інтерфейсу з використанням абстрактних і предметних графічних унітермів

Користувач може редагувати властивості абстрактних і предметних унітермів відповідності з вимогами до інструментальних засобів (рис.4), а також задати всі необхідні події (рис.5.).

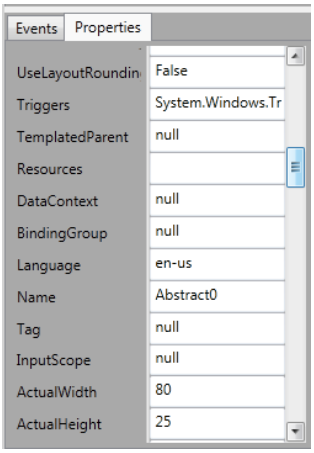

Рис. 4. Вигляд панелі для редагування властивостей абстрактних і предметних графічних унітермів

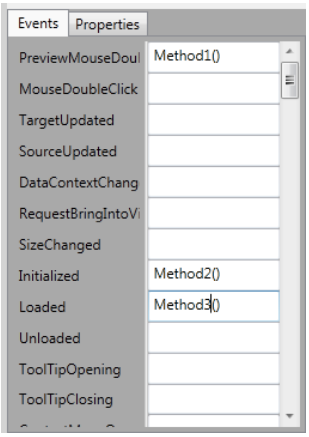

Рис. 5. Вигляд фрагмента панелі для задання подій абстрактних і предметних графічних унітермів інтерфейса користувача

З метою збереження абстрактного проекту для повторного використання або внесення змін у проект, виконується його збереження.

Користувач має змогу конвертувати абстрактні унітерми у будь-який типовий елемент на вибір з присвоєнням йому всіх властивостей і подій абстрактного елемента. На рис.6 зображено інтерфейс після конвертації абстрактного унітерма з навою Abstract0 у графічний елемент StackPanel та абстрактного елемента у вигляді білого прямокутника – у Label.

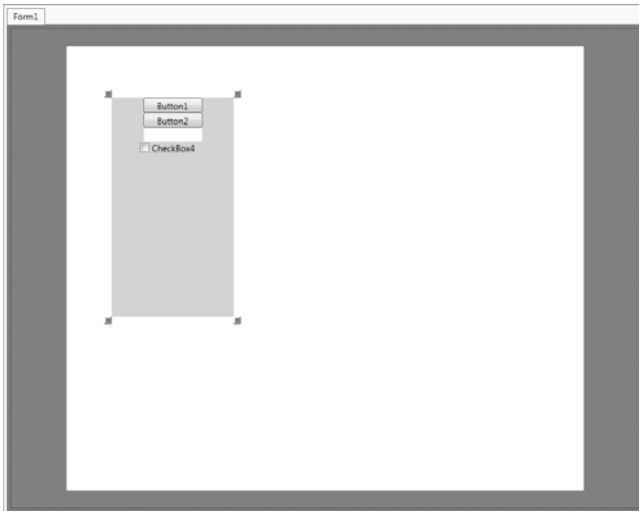

Рис. 6. Інтерфейс проекту після конвертації абстрактних елементів

Після побудови інтерфейсу, користувач може зберегти його як проект одного із таких типів: Windows програма, інтернет сторінка, елемент користувача. Вибір типу проекту виконується у вікні, вигляд якого наведено на рис.7.

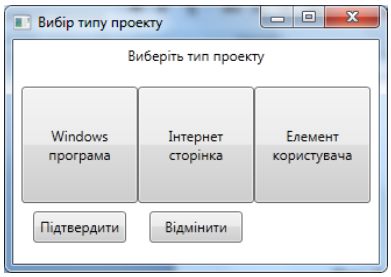

Рис. 7. Вікно вибору типу проекту

На рис.8 показано графічне вікно готового, збереженого і скомпільованого проекту типу Windows програми.

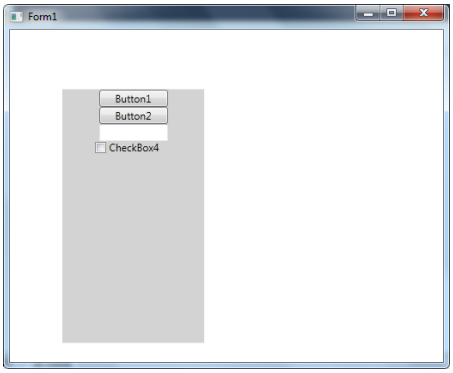

Рис. 8. Вікно скомпільованого проекту типу Windows програми

Вікно збереженого проекту як інтернет сторінки наведено на рис.9.

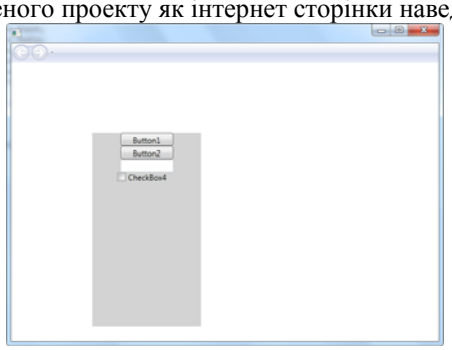

Рис. 9. Вікно проекту збереженого як інтернет сторінка

3. ФРАГМЕНТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Створена модель проектування абстрактних графічних інтерфейсів інформаційних технологій і систем програмно реалізовано на платформі Microsoft Visual Studio .NET, її програмну реалізацію мовами C# і XAML названо “Візуалізатор”.

```
void childCheckbox_Checked(object sender, RoutedEventArgs e)
{
   //elementList.Items.Clear();
   CheckBox lb = (CheckBox)sender;
   Eventlist(lb);
   propertyList(lb);
   removRes();
   re = new ResizeElement(lb, lb.Parent);
   Readevent();
   contr = lb;
```

```
            e.Handled = true;
        }
        void childCheckbox_MouseDown(object sender, MouseButtonEventArgs e)
        {
            //elementList.Items.Clear();
            CheckBox lb = (CheckBox)sender;
            Eventlist(lb);
            propertyList(lb);
            removRes();
            re = new ResizeElement(lb, lb.Parent);
            Readevent();
            contr = lb;
            e.Handled = true;
        }
        private void AddListBox(double x, double y)
        {
            childListBox = new ListBox();
            childListBox.Height = 150;
            childListBox.Width = 80;
            childListBox.BorderThickness = new Thickness(1);
            childListBox.BorderBrush = Brushes.Black;
            childListBox.Name = "ListBox" + bn;
            bn++;
            contr = childListBox;
            Canvas.SetLeft(childListBox, x);
            Canvas.SetTop(childListBox, y);
            oblist.Add(childListBox);
            childListBox.MouseDown += new MouseButtonEventHand-
ler(childListBox_MouseDown);
            CreateItem(childListBox);
            SortedList sl = new SortedList();
            EventL.Add(childListBox.Name, sl);
        }
        void childListBox_MouseDown(object sender, MouseButtonEventArgs e)
        {
            ListBox lb = (ListBox)sender;
            Eventlist(lb);
            propertyList(lb);
            removRes();
            re = new ResizeElement(lb, lb.Parent);
            Readevent();
            contr = lb;
            e.Handled = true;
        }
        private void CreateItem(FrameworkElement cn)
        {
            TreeViewItem iName = new TreeViewItem();
```

```
            iName.Header = cn.Name;
            itemTr.Items.Add(iName);
            TreeViewItem iPropery = new TreeViewItem();
            iPropery.Header = "Properties:";
            iName.Items.Add(iPropery);
            TreeViewItem iEvent = new TreeViewItem();
            iEvent.Header = "Event:";
            iName.Items.Add(iEvent);
        }
        private void Readevent()
        {
            for (int i = 0; i < EventL.Count; i++)
            {
                if (EventL.GetKey(i).ToString() == contr.Name.ToString())
                {
                    SortedList st = (SortedList)EventL.GetByIndex(i);
                    for (int j = 0; j < st.Count; j++)
                    {
                        for (int k = 0; k < evInf.Length; k++)
                        {
                            if (evInf[k].Name.ToString() == st.GetKey(j).ToString())
                            {
                                if (TempGr.Children[k * 2 + 1].GetType() == type-
of(TextBox))
                                {
                                    TextBox tB = (TextBox)TempGr.Children[k * 2 + 1];
                                    tB.Text = EventL.GetKey(i).ToString() + "_" +
st.GetKey(j).ToString();
                                }
                            }
                        }
                    }
                }
            }
        }
        private void removRes()
        {
            if (re != null)
            {
                if (re.moveThumb.Parent != null)
                {
                    if (re.moveThumb.Parent.GetType().Name == "Canvas")
                    {
                        Canvas cn = (Canvas)re.moveThumb.Parent;
                        cn.Children.Remove(re.ldouneThumb);
                        cn.Children.Remove(re.rdouneThumb);
                        cn.Children.Remove(re.rupThumb);
```

```
                cn.Children.Remove(re.moveThumb);
            }
            if (re.moveThumb.GetType().Name == "Abstract")
            {
                Canvas cn = (Canvas)re.moveThumb.Parent;
                cn.Children.Remove(re.ldouneThumb);
                cn.Children.Remove(re.rdouneThumb);
                cn.Children.Remove(re.rupThumb);
                cn.Children.Remove(re.moveThumb);
            }
        }
    }
}
```

4. ВИСНОВКИ

1. Використанням абстрактних графічних елементів – унітермів забезпечує можливості абстрагування від строго типізованих графічних елементів на етапі розробки інтерфейсів інформаційних технологій і систем.

2. Створена модель є універсальнішою, гнучкішою, забезпечує повторне використання спроектованих інтерфейсів для різних типів проектів.

3. Використання абстрактних унітермів дає більші можливості у розробці інтерфейсів та зменшує затрати часу на їх виконання.

4. Створена модель проектування абстрактних графічних інтерфейсів інформаційних технологій і систем програмно реалізовано на платформі Microsoft Visual Studio .NET, її програмну реалізацію названо “Візуалізатор”.

1. Троэлсен Э. Язык программирования C# 2010 и платформа .NET 4.0. – Москва: ООО И.Д. “Вильямс”. 2010. – 1392 с. 2. Powers L., Snell M. Microsoft Visual Studio 2005. Księga eksperta. – Gliwice: HELION. 2007. – 840 s. 3.Шилд Г. C# 4.0: полное руководство. – Москва: ООО И.Д. “Вильямс”. 2011. – 1056 с. 4.Rychlicki-Kicior K. C#. Tworzenie aplikacji graficznych w .NET 3.0. – Gliwice: HELION. 2007. – 301 s. 5.Троэлсен Э., Шпушта М. Microsoft ASP.NET 3.5 с примерами на C# с примерами для профессионалов. – Москва: ООО И.Д. “Вильямс”. 2008. – 1424 с. 6.WPF: Windows Presentation foundation в .NET 3.5 с примерами на C# 2008 для профессионалов. 2-е издание: Пер. с англ. – М. : ООО "И.Д. Вильямс". 2008. – 928 с.