JCSI 2 (2016) 76-80
Journal of Computer Sciences Institute

Wysłane: 2016-10-12
Przyjęte: 2016-10-18

Analiza porównawcza narzędzi RAD do wizualnego programowania w języku C++

Tetiana Pasikova*, Elżbieta Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono porównanie wybranych narzędzi RAD do wizualnego programowania w języku C++. Do porównania zostały wybrane środowiska programistyczne: C++ Builder, Visual Studio, Qt Creator. Wielokryterialna analiza porównawcza i wybór najlepszego środowiska do nauki wizualnego programowania w C++ zostały przeprowadzone na podstawie danych producenta i przeprowadzonego eksperymentu ze studentami.

Słowa kluczowe: wizualne programowanie; C++; C++ Builder; Visual Studio; Qt Creator.

*Autor do korespondencji.

Adres e-mail: pasikova.t@gmail.com

Comparative analysis of visual programming RAD tools for C++ language.

Tetiana Pasikova*, Elżbieta Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparison of visual programming RAD tools for C++ language. For comparison were chosen following development environments: C ++ Builder, Visual Studio, Qt Creator. Multi-criteria comparative analysis and selection of the best learning environment visual programming in C ++ were carried out on the basis of the manufacturer and the experiment with students.

Keywords: Visual programing; C++; C++ Builder; Visual Studio; Qt Creator.

*Corresponding author.

E-mail address: pasikova.t@gmail.com

1. Wstęp

Celem badań przedstawionych w artykule było porównanie wybranych narzędzi RAD (*Rapid Application Development*) do wizualnego programowania w języku C++ i wyłonienie najlepszego z nich na podstawie wybranych kryteriów. Do porównania zostały wybrane następujące środowiska programistyczne: C++ Builder, Visual Studio, Qt Creator. Analiza obejmowała teoretyczne aspekty funkcjonowania ww. narzędzi oraz wykorzystanie ich w praktyce.

Analiza porównawcza środowisk do wizualnego programowania w języku C++ obejmowała następujące etapy:

- badania literaturowe z zakresu programowania wizualnego,
- opracowanie wybranych środowisk RAD – szybkiego wytwarzania aplikacji,
- sformułowanie hipotezy badawczej,
- dobór kryteriów dla porównania narzędzi,
- zaplanowanie i przeprowadzenie eksperymentu ze studentami,
- opracowanie wyników badań,
- wybór najlepszego środowiska i weryfikacja hipotezy badawczej.

Kluczowymi problemami badań są:

1) Jakie środowisko programowania wizulanego w języku C++ jest najlepsze?
2) Czy wcześniejsza znajomość środowiska przez studentów skraca czas stworzenia programu?
3) W jakim zakresie helpy i autopodpowiedzi środowiska ułatwiają napisanie programu?
4) Czy łatwo jest dostosować się do innego środowiska, jeśli ma się doświadczenie pracy w podobnym?

Szukanie odpowiedzi na postawione pytania pozwoli zweryfikować następującą hipotezę badawczą:

Najlepszym środowiskiem dla wizualnego programowania jest QT Creator.

2. Badania literaturowe

Języki używane w programowaniu wizualnym można klasyfikować w zależności od rodzaju i stopnia ekspresji wizualnej na następujące rodzaje [1]:

- języki wykorzystujące obiekty, gdy wizualne środowisko programistyczne zapewnia elementy graficzne lub symboliczne, które mogą być manipulowane interaktywnie zgodnie z pewnymi zasadami;
- języki schematów opartych na idei "kształty i linie", gdzie kształty (prostokąty, owale etc.) są traktowane jako przedmioty i są połączone liniami (strzałki, łuki), np. UML (*Unified Model Language*).

Programowanie wizualne wykorzystuje narzędzia typu RAD, które [2]:

- są ukierunkowane na zminimalizowanie czasu tworzenia aplikacji,
- umożliwiają szybkie opracowanie prototypu w celu udoskonalenia wymagań klienta,
- pozwalają na cykliczny rozwój oprogramowania: każda nowa wersja produktu opiera się na ocenach wyników prac poprzedniej wersji klienta, wykorzystanie gotowych modułów skraca czas tworzenia nowych wersji. Zespół korzystający z narzędzi RAD musi ściśle współpracować, każdy uczestnik musi być gotowy do wykonywania wielu zadań.

Programowanie wizualne umożliwia programowanie (a właściwie projektowanie) interfejsu aplikacji metodą umieszczania na pustych formularzach wymaganych elementów (komponentów). Komponenty wizualne (z interfejsem GUI) umieszczone są na formularzach metodą drop ciągnij i upuść, po skompilowaniu aplikacji wyglądają identycznie (WYSIWYG). Powiązania komponentów też mogą być realizowane wizualnie – można wykorzystać diagramy klas (dla aplikacji bazodanowych), często bazujące na wcześniej zaprojektowanym modelu UML. Programowanie wizualne wykorzystuje narzędzia typu CASE (*Computer Aided Software Engineering*) – zmiany dokonywane wizualnie automatycznie są zamieniane na odpowiedni kod często w kilku plikach jednocześnie, np. w specyfikacji klasy dodawana jest deklaracja metody, natomiast w ciele szablon definicji. Cechy obiektów (własności, metody i zdarzenia) dostępne są za pomocą wygodnych tabelarycznych lub drzewiastych struktur oraz podpowiadane podczas tworzenia kodu (*code completion*). Konfiguracja wartości złożonych i powiązań wspierana jest za pomocą licznych kreatorów (wizards). Składowe projektu zarządzane są za pomocą łatwo konfigurowalnych struktur hierarchicznych [3,4].

Programowanie wizualne jest ściśle powiązane z programowaniem zdarzeniowym. Przebieg realizacji programu zależy od obsługi zdarzeń jakie zachodzą na komponentach aplikacji typu naciśnięcie klawisza, wprowadzanie danych do okienek edycyjnych, przesuwanie myszy, zamykanie formularzy itp.

Programowanie zdarzeniowe jest najczęściej zaliczane do metodyk typu imperatywnego. Do zalet programowania zdarzeniowego można zaliczyć [5]:

- ograniczenie kodu do opisu tylko obsługiwanych zdarzeń,
- łatwość opisania interakcji z otoczeniem,
- posługiwanie się inną metodyką podczas samego opisu funkcji obsługi zdarzeń.

Wady programowania zdarzeniowego[6]:

- przy programowej implementacji wykorzystanie ukrytej, lecz *de facto* istniejącej części programu, odpowiadającej za obsługę pętli komunikatów,
- potrzeba znajomości możliwych zdarzeń i sposobu dostarczenia ich parametrów
- skomplikowanie budowy systemu – potrzeba implementacji sposobu obsługi nadchodzących przerwań lub komunikatów, wyszczególnienie, zdefiniowanie i udokumentowanie zdarzeń.

3. Charakterystyka wybranych środowisk RAD

Do analizy wybrano 3 środowiska szybkiego wytwarzania aplikacji w C++: Qt Creator, C++ Builder, Visual Studio.

Qt jest zespołem bibliotek umożliwiających tworzenie wszechstronnych aplikacji w języku C++. Stanowią one doskonałą bazę dla wieloplatformowych aplikacji z zaawansowanym graficznym interfejsem użytkownika. Wieloplatformowość w tym kontekście należałoby rozumieć jako zdolność do kompilacji kodu źródłowego danego rozwiązania na każdej obsługiwanej platformie bez wprowadzania jakichkolwiek zmian. W chwili obecnej wśród obsługiwanych systemów operacyjnych jest Linux, BSD, Windows, MacOS oraz wiele mniejszych mobilnych platform typu SymbianOS, MeeGo czy Windows CE. Qt jest dostępny pod dwoma licencjami: komercyjną, wymagającą wykupienia ale umożliwiającą produkcję zamkniętego oprogramowania oraz w wersji wolnej opartej na licencji GPL. Na ten moment, chyba najbardziej rozbudowanym projektem ukazującym możliwości biblioteki Qt jest K Desktop Environment (*KDE*) [7].

Borland C++ Builder jest zintegrowanym środowiskiem programistycznym, stanowiącym zbiór niezbędnych narzędzi pomocnych w szybkim tworzeniu aplikacji. Zawiera rozbudowane edytory tekstowe i graficzne, kompilator, linker oraz inne narzędzia pomocnicze. Przy wykorzystaniu C++ Buildera możliwe jest tworzenie aplikacji graficznych, bibliotek dll, kontrolek ActiveX. Z racji tego, że kompilator C++ Builder jest zgodny z standardami ANSI/ISO języka C++ możliwe jest też budowanie aplikacji konsolowych [8].

Wersja Personal zawiera bibliotekę z ponad 100 wizualnymi komponentami wielokrotnego użytku, które są przeciągane myszą na formularz. Oprócz dobrze znanych komponentów sterujących Windows (przyciski, paski przewijania, pola edycyjne, proste i połączone listy, itp) biblioteka udostępnia nowe komponenty wspierające dialog, usługi bazy danych oraz wiele innych.

Visual C++ to część bezpłatnego środowiska programistycznego Visual Studio. Program posiada edytor wizualny z wieloma wbudowanymi kontrolkami, rozbudowany edytor kodu i możliwością tworzenia własnych, system inteligentnych podpowiedzi, debugger, a także oferuje pełną integrację z Microsoft SQL Server 2008 i 2008 R2. Aplikacja posiada także opcję instalacji pakietu MSDN Library - zintegrowanej, również kontekstowej wersji offline pomocy dostępnej na stronach internetowych MSDN. Do instalacji programu wymagane jest połączenie z Internetem – instalator pobiera odpowiednie komponenty online [9].

4. Dobór kryteriów dla porównania narzędzi

Metodą weryfikującą postawioną hipotezę będzie wielokryterialna analiza porównawcza.

Wybrane środowiska zostały przetestowane według poniższych kryteriów:

1) *Licencja*. Cel: czy program jest płatny lub bezpłatny.
2) *Wieloplatformowość*. Cel: czy program jest dostępny dla Windows, Linux, MacOS.
3) *Zajętość pamięci na dysku*. Cel: zbadać ile miejsca potrzebuje program na dysku twardym.
4) *Wspomaganie programisty*. Cel: jak przydatne są helpy i autopodpowiedzi środowiska.
5) *Przyjazność środowiska – łatwość pisania aplikacji.* Cel: napisać program z GUI dla rozwiązania równania kwadratowego i ocenić stopień łatwości jego tworzenia.

Dla każdego kryterium będą wystawione oceny w odpowiedniej skali. Każdemu kryterium będzie przypisana odpowiednia waga, określająca jego ważność w procesie wyboru. Dla podjęcia decyzji będzie zastosowana metoda rozwiązywania problemów decyzyjnych – ELECTRE I. Metoda ELECTRE I zakłada, że porównując dwa warianty decyzyjne a i b możemy mieć do czynienia z jedną i tylko jedną z następujących sytuacji: a jest uznawane za równoważne b, a jest preferowane w stosunku do b, b jest preferowane w stosunku do a.

Ostatnie kryterium: przyjazność środowiska zostanie oceniona za pomocą następującego eksperymentu: wybrani studenci (10 osób) napiszą aplikacje graficzne rozwiązujace równanie kwadratowe w każdym środowisku i ocenią procesy ich tworzenia. Dla studentów przygotowano instrukcje opracowania aplikacji oraz kwestionariusze samooceny znajomości środowisk i oceny procesu tworzenia aplikacji pod kątem przyjazności środowiska.

Metody ELECTRE stosowane są w przypadkach szczególnie złożonych sytuacji decyzyjnych. Metody te wymagają zastosowania co najmniej trzech lub więcej kryteriów oceny. Są użyteczne zwłaszcza w sytuacjach, gdy zbiór dobranych kryteriów ma charakter heterogeniczny, tzn. gdy brane pod uwagę cechy różnią się istotnie między sobą [10].

Metoda ELECTRE I (***El****imination* ***e****t* ***C****hoice* ***T****ranslating* ***Re****ality)* – wykorzystuje koncepcję relacji przewyższenia A_k -> A_l, która mówi, że nawet jeśli dwa warianty nie dominują się wzajemnie, to decydent akceptuje ryzyko traktowania wariantu A_k jako prawie na pewno lepszego od wariantu A_l. Metoda ta opiera się na porównaniach parami wszystkich wariantów decyzyjnych, co ma doprowadzić do ujawnienia częściowego uporządkowania wariantów zgodnie z preferencjami decydenta [11].

Metoda wykorzystuje test zgodności i niezgodności:

- **zgodność** – czyli stopień w jakim wagi preferencji są w zgodzie z relacją dominacji par;
- **niezgodność** – czyli stopień w jakim obliczenia wagowe różnią się między sobą.

Metoda do utworzenia końcowego rankingu wykorzystuje dwie macierze:

- **macierz zgodności** *(concordance matrix)* **–** macierz wartości obliczanych dla każdej pary kryteriów, informujących w jakim stopniu jedna alternatywa jest co najmniej tak dobra jak druga.
- **macierz niezgodności** *(discordance matrix)* **–** macierz wartości obliczanych dla każdej pary kryteriów, informujących w jakim stopniu jedna alternatywa jest gorsza od drugiej.

Badane środowiska (alternatywy) a także kryteria oceny środowisk i ich wagi w skali od 1 (mała) do 9 (duża) przedstawiono w tabeli 1.

Tabela 1. Alternatywy i kryteria oceny środowisk

Alternatywa:		Kryterium:		waga
A1	Visual Studio	Q1	Licencja	6
A2	QT Creator	Q2	Wieloplatformowość	3
A3	C++ Builder	Q3	Zajętość pamięci	3
		Q4	Wspomaganie programisty	7
		Q5	Przyjazność środowiska	8
			suma	27

Kryterium Q5 - Przyjazność środowiska będzie ocenione podczas eksperymentu ze studentami na podstawie wypełnionych kwestionariuszy według kryteriów przedstawionych w tabeli 2.

Tabela 2. Alternatywy i kryteria oceny przyjazności środowiska

Alternatywa:		Kryterium:	
A1	Visual Studio	Q1	Znajomość języka C++
A2	QT Creator	Q2	Czas stworzenia formularza
A3	C++ Builder	Q3	Całkowity czas pisania programu
		Q4	Łatwość pisania aplikacji w danym środowisku
		Q5	Trudność adaptacji do składni kodu
		Q6	Intuicyjność interfejsu

5. Wyniki badań

Cztery pierwsze kryteria oceny wybranych środowisk wizualnego programowania w języku C++ (licencja, wieloplatformowość, zajętość pamięci, wspomaganie programisty) oceniono na podstawie danych producenta. Piąte kryterium – przyjazność interfejsu na podstawie eksperymentu – badania przeprowadzonego na grupie 10 studentów. Każdemu studentowi dano za zadanie napisać prosty program dla rozwiązania równania kwadratowego w trzech środowiskach programistycznych: Visual Studio, QT Creator, C++ Builder. Każdy student otrzymał szczegółową instrukcję w każdym środowisku niezbędną do napisania programu. Po wykonaniu zadania studenci

wypełniali kwestionariusz odpowiadając na pytania o łatwości użycia każdego środowiska.

Wyniki każdego studenta przeanalizowane zostały za pomocą metody wielokryterialnego wyboru – ELECTRE I. Dzięki niej oceniono jakie środowisko każdy student uważa za najlepsze. Na podstawie ponownego wykorzystania metody ELECTRE I dla 5 wybranych kryteriów wybrano środowisko programistyczne które najbardziej pasuje dla wizualnego programowania w języku C++.

Tabela 3. Podsumowanie wyników eksperymentu ze studentami

Nr studenta	Znajomość języka C++			Znajomość środowiska			Najlepsze środowisko
	zaawansowana	średni	podstawowa	Visual studio	QT Creator	C++ Builder	
1		X		X			Visual studio
2		X		X			Visual studio
3			X			X	C++ Builder
4			X		X		C++ Builder
5			X		X		QT Creator
6		X		X			Visual studio
7		X				X	C++ Builder
8			X		X		QT Creator
9	X					X	C++ Builder
10	X			X			QT Creator

W tabeli 3 przedstawiono wybór każdego studenta. Środowisko C++ Builder otrzymało najwięcej punktów – 4. Środowiska QT Creator i Visual Studio po 3 punkty.

Jak wynika z rezultatów badań – większość studentów zaznacza, że najlepszym środowiskiem dla wizualnego programowania jest to, z którego korzystali oni wcześniej.

Zgodnie z metodą ELECTRE I policzono współczynniki macierzy zgodności (tabela 4) i niezgodności (tabela 5). Zgodnie z zasadami indeksu zgody oraz indeksu niezgody ustawiono progi współczynnika zgodności c1 na 0,48 i współczynnika niezgodności d1 na 0,7.

Tabela 4. Macierz zgodności

Cij	A1	A2	A3
A1	–	0,2592592593	0,4814814815
A2	0,7407407407	–	0,3333333333
A3	0,5185185185	0,6666666667	–

Tabela 5. Macierz niezgodności

Cij	A1	A2	A3
A1	–	0,6666666667	0,6666666667
A2	0,6666666667	–	0,6666666667
A3	0,6666666667	0,6666666667	–

Tabela zalet alternatyw odnośnie innych dla badania na podstawie danych producenta i wyników eksperymentu ze studentami (tabela 6) zbudowana została w następujący sposób: jeżeli indeks zgodności Aij > c1 i indeks niezgodności Aij < d1 to alternatywa A1 jest lepsza od alternatywy A2 i ma znaczenie „+” w innym przypadku ma znaczenie „–”. Najlepszą alternatywą jest ta, która ma najwięcej „+”.

Tabela 6. Zalety alternatyw odnośnie innych

Cij	A1	A2	A3
A1	*	–	+
A2	+	*	-
A3	+	+	*

Najwięcej zalet ma alternatywa A3 – środowisko C++ Builder.

Zgodnie z wynikami badań analizy na podstawie przyjętych kryteriów i ich wag najlepszym środowiskiem dla wizualnego programowania okazało się środowisko C++ Builder. Przyjęta hipoteza ***nie została potwierdzona.***

6. Wnioski

Narzędzia do programowania wizualnego znacznie ułatwiają tworzenie aplikacji. Środowiska te mają na celu maksymalizację wydajności programisty. Można administrować programem w trakcie jego tworzenia za pomocą dwóch widoków: zakładki kodu i zakładki widoku obiektów graficznych.

W przebiegu badań miały znaczenie wagi kryteriów, co pozwoliło zaznaczyć, na ile ważne jest każde kryterium w stosunku do innych. Po testach zostały wystawione oceny dla każdej charakterystyki. Biorąc pod uwagę małą różnicę między ocenami i różnymi wagami kryteriów do podjęcia decyzji była wybrana metoda rozwiązywania problemów decyzyjnych – ELECTRE I, która pozwoliła policzyć wyniki uwzględniając wagi kryteriów.

Na podstawie przeprowadzonych badań najlepszym środowiskiem dla wizualnego programowania w języku C++ jest C++ Builder.

Studenci z podstawową znajomością języka C++ najczęściej wybierają środowisko C++ Builder. Dla studentów z lepszą znajomością języka C++ nie jest trudne korzystanie z żadnego środowiska i wybierają te z którego korzystali już wcześniej. Ośmiu z 10 studentów stwierdziło, że najlepszym środowiskiem jest to z jakiego korzystali wcześniej, dwu z 10 studentów zmienilo swoje zdanie.

Literatura

[1] К.А. Хайдаров, 4GL-Технологии. Основы визуаль-ного программирования, http://bourabai.kz/einf/4gl. htm, 02.05.2016

[2] К.А. Хайдаров, RAD – технология быстрого программирования, http://bourabai.kz/C-Builder/RAD. htm, 14.04.2016

[3] S. Vohra, Object Oriented Programming with C++: OOPC, bookrent.in Impression, 2015.

[4] K. Kuczmarski, Kurs C++, Helion, Xion, 2004.

[5] J. Matulewski, Podstawy programowania RAD z użyciem narzędzi Borland Delphi/C++ Builder, ćwiczenia, Helion, Gliwice, 2003.

[6] А.Я. Архангельский, Программирование в Borland C++, Москва, Бином, 2001.

[7] У.Р. Алексеев, Г.Г. Злобин, Программирование на языке C++ в среде QT Creator, Москва, ALT Linux, 2015.

[8] О. Вальпа, Borland C++ Builder - Экспресс-курс, Петербург, БХВ, 2006.

[9] R. Simiński, Programowanie w środowiskach RAD, http://programowanie.siminskionline.pl/resource/pwsz/qt_ w02.pdf, [30.09.2016].

[10] ELECTRE, http://brasil.cel.agh.edu.pl/~13sustrojny/ electre-iii/, [30.09.2016].

[11] О.И. Ларичев, Метод ELECTRE, http://bugabooks.com/book/ 264-teoriya-i-metody-prinyatiya-reshenij-a-takzhe-xronika-sobytij-v-volshebnyx-stranax/86-4-metod-electre-i.html, [30.09.2016].