INFRASTRUKTURA I EKOLOGIA TERENÓW WIEJSKICH
INFRASTRUCTURE AND ECOLOGY OF RURAL AREAS

Nr 2/III/2013, POLSKA AKADEMIA NAUK, Oddział w Krakowie, s. 95–103
Komisja Technicznej Infrastruktury Wsi

WYKORZYSTANIE WIEDZY Z KURSU PROGRAMOWANIA W PRAKTYCE INŻYNIERSKIEJ

Jacek Derwisz
Wyższa Szkoła Inżynieryjno Ekonomiczna z siedzibą w Rzeszowie

APPLYING KNOWLEDGE GAINED FROM A COMPUTER PROGRAMMING COURSE TO ENGINEERING PRACTICE

Streszczenie

W rozporządzeniu Ministra Nauki i Szkolnictwa Wyższego znaleźć można załącznik, który określa standardy kształcenia dla kierunku studiów „Geodezja i Kartografia" stopnia I, między innymi w zakresie informatyki. Autor od wielu lat łączy nauczanie programowania z praktyką inżynierską i nie ma wątpliwości co do celowości wzbogacenia umiejętności inżyniera geodety o tworzenie programów komputerowych, zdarza się jednak, że celowość nauczania programowania poddawana jest w wątpliwość i to zarówno przez studentów jak i nauczycieli akademickich. Opisany w niniejszym sposób rozwiązania problemów, które pojawiły się podczas wykonywania modernizacji mapy numerycznej, stanowi w opinii autora silny argument popierający tezę o celowości kształcenia w tym zakresie. Przedsiębiorstwo, którym opiekował się autor, podjęło się wykonania prac, których celem była poprawa kartometryczności mapy numerycznej zawierającej wybrane obiekty bazy danych geoprzestrzennych. Dotrzymanie terminu modernizacji mapy było by niemożliwe bez stworzenia szeregu drobnych aplikacji usprawniających wykonywanie prac. Jak się okazało po zakończeniu prac, aplikacje te skróciły czas potrzebny na wykonanie modernizacji o 1/3 a co najważniejsze, wiedza niezbędna do ich napisania w znacznej części pokrywa się z tą, która przekazywana jest na prowadzonych przez autora zajęciach.

Summary

In one of the Regulations of the Ministry of Science and Higher Education one can find an appendix which defines standards of education for the Geodesy and Cartography of the first level of studies, also in terms of computer programming. The author of the article has for many years combined teaching program-

ming with engineering practice and he has no doubts that it is essential to teach future engineer surveyors how to write computer programs. However, the advisability of teaching programming sometimes raises concerns both among some students and academic teachers. The method of solving problems which was used during the process of numerical map modernization, described in the article, is in the author's opinion a strong argument in favour of advisability of computer programming education . The company run by the author has performed works which aim was to improve the cartometricity of a numerical map which contained selected objects of geospatial data. Meeting a deadline of the works would have been impossible without some tiny applications facilitating unnecessary works. After the works had been completed, it turned out that the applications shortened the time of making the modernization by 1/3 and what is more, knowledge unnecessary to make the applications is mainly the same that students can get during the computer programming course run by the author.

WSTĘP

W rozporządzeniu Ministra Nauki i Szkolnictwa Wyższego z dnia 12 lipca 2007 roku, w sprawie standardów kształcenia dla poszczególnych kierunków oraz poziomów kształcenia, a także trybu tworzenia i warunków, jakie musi spełniać uczelnia, by prowadzić studia międzykierunkowe oraz makrokierunki (Dz. U. z dnia 13 września 2007 r.), znaleźć można załącznik numer 36, który określa standardy kształcenia dla kierunku studiów „Geodezja i Kartografia" stopnia I. W punkcie 4 zapisane zostały treści kształcenia w zakresie informatyki, między innymi „*...Projektowanie aplikacji. ... Elementy programowania obiektowego. Oprogramowanie wspomagające wykonywanie obliczeń geodezyjnych ...*" [*Rozporządzeniu MNiSW z dnia 12 lipca 2007*]. Autor od wielu lat łączy nauczanie programowania z praktyką inżynierską i nie ma wątpliwości co do celowości wzbogacenia umiejętności inżyniera geodety o tworzenie programów komputerowych. Zdarza się jednak, że w dyskusjach – zwłaszcza tych mniej formalnych - celowość nauczania programowania poddawana jest w wątpliwość i to zarówno przez studentów jak i nauczycieli akademickich. Artykuł ten jest głosem w dyskusji na temat przydatności wiedzy zdobywanej na kursie programowania a powstał podczas trwających dwa miesiące geodezyjnych prac kameralnych, których celem była poprawa jakości mapy numerycznej. Autor celowo pominął większość szczegółów dotyczących danych i zastosowanego oprogramowania, koncentrując się na wykazaniu bezpośredniego związku umiejętności, które zdobyć można na zajęciach z przedmiotów „Podstawy programowania" oraz „Programowanie specjalistyczne" z ekonomicznym efektem ich wykorzystania w geodezyjnej praktyce inżynierskiej.

OPIS WYKONANYCH PRAC

Przedsiębiorstwo, które podlegało nadzorowi autora, podjęło się wykonania prac, których celem była poprawa kartometryczności mapy numerycznej. Prace trwały od października do grudnia 2012 roku. Materiałem, na którym wykonywane były prace, był rysunek wektorowy zawierający wybrane obiekty bazy danych geoprzestrzennych. Korekta położenia obiektów wykonywana była na podstawie aktualnego i skalibrowanego materiału rastrowego (ortofotomapy). Ponieważ zweryfikowany rysunek wektorowy miał z powrotem zasilić bazę danych należało podczas prac zastosować reżim technologiczny, który uniemożliwiłby utratę informacji, redundancję lub pojawienie się nowych obiektów, których nie było w bazie danych. Podczas prac nie mogły również ulec uszkodzeniu informacje o związku obiektów z rekordami bazy danych. Inaczej mówiąc, na rysunku wektorowym nie mógł się pojawić żaden nowy obiekt, żaden obiekt nie mógł z niego zniknąć i nie mogły ulec usunięciu czy modyfikacji informacje bazodanowe związane z elementami rysunku. Obiektom rysunku wektorowego, które powinny zostać usunięte z bazy danych zmieniano status zaś nowe obiekty kreślone były w osobnych plikach. Korekta rysunku wektorowego polegała na przesuwaniu węzłów konstrukcyjnych elementów rysunku tak, aby zgodnie ze szczegółowymi wytycznymi zleceniodawcy, pokrywały się z obiektami na rastrze. Przesunięcie każdego węzła konstrukcyjnego wymagało wskazania go kursorem myszy i przesunięcia w odpowiednie miejsce. Nie była to więc czynność wymagająca wiedzy czy doświadczenia z zakresu geodezji, dlatego do prac wybrane zostały osoby o bardzo dobrej znajomości programu, w którym wykonywane były prace i dużej wprawie w posługiwaniu się klawiaturą i myszką.

DANE I NARZĘDZIA ZASTOSOWANE W PRACACH

Zbiór danych, które zostały przekazane przedsiębiorstwu przez zleceniodawcę, składał się z czterech plików wektorowych w formacie „dgn" (natywny format programu Bentley MicroStation), z których każdy posiadał podłączony referencyjnie zestaw rastrów, zapisanych w plikach formatu „tiff" zawierających dane georeferencyjne. Łączna objętość zbiorów danych nieznacznie przekroczyła 0,4TB i co oczywiste, główną część tej wielkości stanowiły pliki rastrowe. Jednym z powodów, dla których opisywane tu przedsiębiorstwo podjęło się wykonania prac, był fakt, że posiadało ono sprzęt oraz i oprogramowanie (co rzadkie a warte podkreślenia – licencjonowane) zainstalowanego na odpowied-

niej liczbie stanowisk. Program MicroStation należy do nielicznej grupy programów, które bez problemów radzą sobie ze zbiorami o takiej wielkości, a jak się ponadto okazało w czasie wykonywania prac, jest wyposażony w narzędzie bez którego dotrzymanie terminu prac nie byłoby możliwe – język programowania - Visual Basic for Application, który daje dostęp do składników rysunku, pozwalając na tworzenie programów (makropoleceń) usprawniających pracę.

OPIS PROBLEMU.

Wspomniane przedsiębiorstwo posiadało, jak wynika z przedstawionego opisu, wszystko co było niezbędne do sprawnego wykonania pracy: wykwalifikowany personel, sprzęt i oprogramowanie. Jedynym problemem był bardzo krótki termin realizacji prac. Po przejęciu kompletu danych oszacowano czas niezbędny do wykonania pracy a wówczas okazało się, że dotrzymanie terminu możliwe jest tylko pod warunkiem, że niemal cały przewidziany na wykonanie pracy czas zostanie poświęcony na korygowanie położenia węzłów konstrukcyjnych obiektów. Jedynym rozwiązaniem było zatem zredukowanie do minimum czasu potrzebnego na pozostałe czynności, takie jak:

- podłączanie i odłączanie plików rastrowych,
- modyfikowanie statusu obiektu i zaznaczania obiektów, które zostały zweryfikowane,
- ustawienie koloru, warstwy i przygotowanie narzędzi do kreślenia,
- przerysowywanie obiektów narysowanych niewłaściwymi narzędziami,
- kontrola spójności rysunku (odpowiednie obiekty na odpowiednich warstwach),
- poszukiwanie obiektów z uszkodzonymi informacjami bazodanowymi,
- poszukiwanie obiektów bez informacji bazodanowych,
- przerysowanie obiektu z zastosowaniem poprawnych narzędzi,
- ustawianie okna widokowego nad wybranym obiektem (np. błędnym),
- sprawdzanie czy składniki obiektu mają identyczne informacje bazodanowe oraz status,
- wykonywanie kopii bezpieczeństwa i kopii archiwalnych,
- kontrolę postępu prac.

Oczywiście wpływ niektórych z tych operacji na realizację prac był mniejszy, gdyż wykonywane były one sporadycznie (np. kopie zapasowe czy kontrola postępu prac) ale większość wykonywana była bardzo często (modyfikacja statusu, kontrola uszkodzeń, przygotowanie koloru itp.).

ROZWIĄZANIE PROBLEMU.

Wykonawca postanowił podzielić cztery główne pliki na 10 stanowisk. Zwiększało to szanse na dotrzymanie wyznaczonego terminu, choć równocześnie zwiększało ilość pracy niezbędną do podzielenia zbiorów, skoordynowania prac na stykach i na koniec ponownego połączenia zbiorów. Prace nad programami usprawniającymi ruszyły równocześnie z pracami nad korektą rysunków wektorowych. Taka metoda wdrażania oprogramowania niesie ogromne ryzyko niepowodzenia ale napięty harmonogram prac nie stwarzał możliwości wyboru. Aby zminimalizować ryzyko uszkodzenia danych, każdy nowo napisany program był najpierw testowany przez programistę na osobnym stanowisku ze zbiorem danych przeznaczonym wyłącznie do testów, później testowany był przez jednego z pracowników i dopiero po pozytywnym przejściu obu testów, włączany był do produkcji. Przed włączeniem do produkcji programów, które znacząco modyfikowały dane, wykonywane były dodatkowe kopie bezpieczeństwa przetwarzanych plików. Czas potrzebny na wykonanie pracy przed i po wprowadzeniu nowego programu był mierzony i notowany. Takie rozwiązanie zapewniło bezpieczeństwo danych podczas wdrażania programów i dało możliwość porównania stopnia przyspieszenia (bądź spowolnienia) tempa prac. Podczas trwających dwa miesiące prac, napisanych zostało kilkanaście niewielkich programów, z których 10 zostało wdrożonych do produkcji. Jeden z nich zostanie opisany w rozdziale związanym z przykładem zastosowania opracowanego oprogramowania.

PRZYKŁAD ZASTOSOWANIA OPRACOWANEGO OPROGRAMOWANIA

Zadaniem, od którego rozpoczęto usprawnianie prac, była zmiana statusu i koloru obiektów zweryfikowanych. Każdemu obiektowi, którego położenie i kształt zostały zweryfikowane, zmieniano status i kolor. Oczywiście użyte oprogramowanie dawało bogaty zestaw narzędzi, którymi można wykonać te zmiany ale liczba operacji, które należało wykonać była spora a w związku z tym zajmowały one sporo czasu. Również z tego powodu prawdopodobieństwo popełnienia błędu było wysokie. Ponieważ obiektów w każdym pliku było około kilkaset tysięcy, to każdy zaoszczędzony ułamek sekundy przekładał się na znaczącą liczbę godzin. W tabeli 1 zestawione zostały działania, które należało wykonać aby zmienić status i atrybuty graficzne obiektu przy pomocy narzędzi programu (kolumna „Narzędzia") i napisanego programu (kolumna „Makropolecenie"). Podczas wykonywania zmian narzędziami programu, najbardziej czasochłonne i narażone na błędy były operacje numer 3,4 oraz 7.

Tabela 1. Operacje wykonywane podczas zmiany statusu i koloru obiektu

Nr	Narzędzia 3,0s	Makropolecenie 1,0s
1	wybierz narzędzie do zmiany statusu obiektu	zaznacz obiekt
2	zaznacz obiekt	uruchom program zmiany statusu i atrybutów
3	w tabelce statusu odnajdź właściwy wiersz	wciśnij klawisz zmiany atrybutu
4	wprowadź nową wartość statusu	
5	zamknij okno tabelki statusu	
6	wybierz funkcję zmiany atrybutów	
7	ustaw właściwy zestaw atrybutów i nowy kolor	
8	wskaż obiekt	
9	zatwierdź zmianę atrybutów	

Program, którego napisanie zajęło autorowi kilkanaście minut, został dołączony do zbioru funkcji uruchamianych skrótami klawiszowymi, dzięki czemu czas potrzebny na wykonanie wszystkich operacji (wskazanie, uruchomienie i wybranie statusu) spadł poniżej 1s. Pomiar dla obu metod zmiany statusu i koloru, wykonano podczas testów wdrożeniowych programu na kilkuset obiektach, poświęcając na to jeden dzień. Jak się okazało, zastosowanie programu pozwoliło zaoszczędzić 2 sekundy na każdym obiekcie, co może wydawać się niewielką oszczędnością czasu ale w przypadku jednego tylko pliku zawierającego około 100 tysięcy obiektów oznacza to niemal 7 dni! Poniżej zamieszczona została zasadnicza część programu zmiany statusu i koloru elementu rysunku. Dla poprawienia czytelności, program został przed umieszczeniem w artykule uzupełniony o komentarze (kolor zielony) a dla zilustrowania z jaką częścią kodu studenci zapoznają się podczas zajęć, te instrukcje, które nie są omawiane zostały zaznaczone na niebiesko.

```
' jeżeli użytkownik wcisnął odpowiedni klawisz...
If (KeyCode = 65) Or (KeyCode = 66) Then
  ' ...tworzymy zbiór zaznaczonych elementów rysunku „ZER”.
  Set ZER = ActiveModelReference.GetSelectedElements
  ' Dla każdego zaznaczonego elementu rysunku...
  Do While ZER.MoveNext
    ' ... sprawdzamy czy numer koloru jest równy 1, oraz...
    If (ZER.Current.Color = 1) Then
      ' ... czy obiekt narysowany jest właściwą linią.
      If (ZER.Current.IsLineElement)Or(ZER.Current.IsClosedElement) Then
        ' Jeżeli tak, to podstawiamy odnaleziony element po zmienną LN...
        Set LN = ZER.Current
            ' ... zmieniamy kolor elementu na kolor numer 7 i ...
        LN.Color = 7
```

```
        ' jeżeli element posiada dołączone informacje bazodanowe, to...
        If (LN.HasAnyTags) Then
          ' ... to pobieramy te informacje...
          TG = LN.GetTags
          TAGindex = -1
                    ' ...przeszukujemy je...
          For TAGtest = 0 To 2
            ' ...odnajdujemy (właściwą) pozycję o nazwie „AB”.
            If (TG(TAGtest).TagDefinitionName="AB") Then TAGindex=TAGtest
          Next TAGtest
          ' Jeżeli właściwa pozycja została odnaleziona, to...
          If (TAGindex >= 0) And (TAGindex <= 2) Then
            ' modyfikujemy ją w zależności od wciśniętego klawisza
            If (KeyCode = 65) Then TG(TAGindex).Value = "A"
            If (KeyCode = 66) Then TG(TAGindex).Value = "B"
                      ' aktualizujemy zestaw informacji bazodanowych
            TG(TAGindex).Rewrite
            ' powiększamy licznik ilości zmodyfikowanych obiektów
            Licznik = Licznik + 1
          End If
        End If
              ' aktualizujemy atrybuty graficzne obiektu
        LN.Rewrite
      End If
    End If
  ' i wracamy na początek pętli
  Loop
  ' na koniec wyświetlamy komunikaty i ostrzeżenia
  If (TAGindex < 0) Then
    MsgBox "Proszę skontrolować i zmienić status ręcznie."
  End If
  If (Licznik = 0) Then
    MsgBox "Żaden element nie został zmodyfikowany."
  End If
End If
```

Kod 1. Procedura zmiany statusu i koloru elementu

Zasadnicza część programu, to tylko kilkanaście linii, z których tylko cztery zawierają polecenia, które należało samodzielnie odnaleźć przy pomocy systemu kontekstowych podpowiedzi programu Bentley MicroStation lub wyszukiwarek sieci Internet. Dla porównania, przedstawiony zostanie teraz kod programu tworzonego przez studentów podczas ćwiczeń programowania. Program ten ma odnaleźć wśród zaznaczonych elementów rysunku, linię o kolorze numer 1 i zmienić go na kolor numer 2.

```
' Deklarujemy zmienną typu ElementEnumerator.
Dim EE As ElementEnumerator
' deklarujemy zmienną typu Element
Dim Poszukiwany As Element

' Wiążemy zmienną EE ze zbiorem zaznaczonych elementów.
Set EE = ActiveModelReference.GetSelectedElements
' Przeszukujemy zbiór zaznaczonych elementów w pętli Do While - Loop.
Do While EE.MoveNext
  ' sprawdzamy czy element jest linią o kolorze numer 1
  If (EE.Current.IsLineElement) and (EE.Current.Color = 1) Then
    ' Podstawiamy aktualny element pod zmienną Poszukiwany.
     Set Poszukiwany = EE.Current
    ' Zmieniamy kolor.
    Poszukiwany.Color = 2
    ' „Utrwalamy" kolor elementu.
    Poszukiwany.Rewrite
  End If
Loop
```

Kod 2. Procedura przeszukania rysunku i zmiany koloru elementu

Różnica pomiędzy programem, który należało napisać dla usprawnienia przebiegu prac (kod 1) a pisanym w ramach ćwiczeń (kod 2) sprowadza się do konstrukcji warunku logicznego w instrukcji poszukiwania danych oraz zestawu instrukcji do poszukiwania i modyfikacji informacji zapisanych w „metkach" (ang. tag) dołączonych do elementów. Stopień złożoności pozostałych programów napisanych podczas realizacji prac, był podobny do przedstawionego w przykładzie, a udział poleceń, które nie były omawiane na zajęciach do wszystkich użytych, w żadnym z programów nie przekroczył 20%.

PODSUMOWANIE I WNIOSKI

Wszystkie prace związane z realizacją projektu korekty map zajęły około 2950 godzin. Prace nad oprogramowaniem zajęły nieco ponad 10 godzin, czyli około 0,34% czasu przeznaczonego na wszystkie prace. Zastosowanie programów spowodowało średnie skrócenie czasu potrzebnego na wykonanie prac o 1/3 co oznacza skrócenie czasu realizacji projektu o ponad 1500 godzin. Po zamknięciu i podsumowaniu projektu, okazało się, że koszty pracy pochłonęły 50% przychodu, co oznacza, że dzięki zastosowaniu programów zysk wzrósł o nieco ponad 48%. Każda minuta pracy programisty zaoszczędziła 2,5 godziny czasu pracy kreślarzy. Podobne prace kameralne, są typowe dla współczesnego

rynku prac geodezyjnych, jednak podejmując ryzyko ich wykonania należy wziąć pod uwagę koszty zatrudnienia programisty, które są wyższe niż koszty zatrudnienia kreślarza oraz koszty testowania i wdrożenia programów, które mogą być znaczące, a które w tym projekcie były pomijalnie niskie. Trzeba również wziąć pod uwagę fakt, że z racji doświadczenia i wprawy autor napisał potrzebne programy szybciej niż zrobiłby to absolwent kurs programowania. Ważniejsze wydaje się być jednak to, że zastosowany w opisanych programach zestaw funkcji i metod programowania w 100% pokrywa się z zestawem omawianym na kursie podstaw programowania a w 80% z zestawem narzędzi VBA omawianych na kursie programowania specjalistycznego. Pozostałe 20% wiedzy, która niezbędna była do napisania programów, można znaleźć dzięki systemowi pomocy języka programowania lub na listach dyskusyjnych w Internecie a więc w sposób bardzo prosty i nie wymagający ponoszenia dodatkowych nakładów. Nie ma wątpliwości, że w opisanym przykładzie dotrzymanie terminu realizacji prac, przekładające się wprost na wynik finansowy przedsiębiorstwa oraz na obraz wykonawcy jako rzetelnego partnera, zostało w znacznym stopniu osiągnięte dzięki wiedzy, która przekazywana jest na zajęciach z przedmiotów „Podstawy programowania" oraz „Programowanie specjalistyczne". Pośrednim potwierdzeniem wartości tej wiedzy są ceny komercyjnych kursów programowania, które w przypadku porównywalnego zakresu materiału, sięgają kilku a nawet kilkunastu tysięcy złotych. Opisane tu efekty zastosowania w praktyce wiedzy z zakresu programowania, zdobytej podczas studiów nie pozostawiają żadnych wątpliwości co do potrzeby kształcenia w tym zakresie. Pozostaje jednak problem przekonania o tym tych, którzy powinni być tą wiedzą najbardziej zainteresowani.

BIBLIOGRAFIA

Rozporządzeniu Ministra Nauki i Szkolnictwa Wyższego z dnia 12 lipca 2007 roku, w sprawie standardów kształcenia dla poszczególnych kierunków oraz poziomów kształcenia, a także trybu tworzenia i warunków, jakie musi spełniać uczelnia, by prowadzić studia międzykierunkowe oraz makrokierunki (Dz. U. z dnia 13 września 2007 r.). Załącznik 36: standardy kształcenia dla kierunek studiów „Geodezja i Kartografia" stopnia I.

Dr inż. Jacek Derwisz
Katedra Geodezji i Geoinformatyki
Wyższa Szkoła Inżynieryjno-Ekonomiczna z siedzibą w Rzeszowie
e-mail: jacekderwisz@gmail.com