Spis treści • *Contents*

RECENZOWANE ARTYKUŁY NAUKOWE
REVIEWED SCIENTIFIC ARTICLES

OD REDAKTORA NACZELNEGO

Dr inż. Tomasz Długosz
Redaktor naczelny
Biuletyn Wrocławskiej
Wyższej Szkoły Informatyki Stosowanej
ul. Ks. Marcina Lutra 4
54-239 Wrocław
http://wydawnictwo.horyzont.eu

Drodzy Czytelnicy!

Mają Państwo przed sobą kolejny numer *Biuletynu Wrocławskiej Wyższej Szkoły Informatyki Stosowanej. Informatyka*. W tym wydaniu znajdą Państwo osiem artykułów.
Pierwsza praca autorstwa Pawła Iljaszewicza zatytułowana *„Analiza dopasowania modelu do obiektu po etapie segmentacji obrazu wykorzystującego działy wodne (ang. watershed) przy rozpoznawaniu barw obrazów”* poświęcona jest tematyce skutecznej metody detekcji obiektów. Autor w swojej pracy wykorzystuje fakt, że poszukiwany obiekt powinien wskazywać podobieństwo geometryczne dotyczące m. in. kształtu, rozmiaru, czy położenia. Zaproponowany w pracy algorytm umożliwia dokładne dopasowanie zadanego modelu do poszukiwanego obiektu.
Druga praca *„How To Not Compare Results In Bioelectromagnetic Studies”*, której autorem jest Tomasz Długosz, przedstawia problematykę jednoznaczności wyników badań bioelektromagnetycznych uzyskiwanych w różnych laboratoriach. W pracy wykorzystano różne metody numeryczne do prezentacji zjawisk mających ogromny wpływ na powtarzalność eksperymentów, co jest kluczowym aspektem w każdej dziedzinie nauki.
Kolejny artykuł autorstwa Michała Zubelewicza i Ewy Kardasz pt.: „*Human Skills, czyli motywowanie pracowników i delegowanie zadań”* przedstawia zagadnienie motywowania pracowników i delegowania zadań z uwzględnieniem aspektów informatycznych.
W czwartej pracy zamieszczonej w tym wydaniu Biuletynu „*Dekompozycja ciągu uczącego dla rozproszonej bazy danych”* Swietłany Lebiediewy sformułowany został problem dekompozycji ciągu uczącego dla rozproszonej bazy danych. W artykule tym zdefiniowano trzy rodzaje dekompozycji i zaprezentowano twierdzenia dotyczące zajętości pamięci przez zdekomponowany ciąg uczący.
Kolejny artykuł autorstwa Piotra Kardasza „*Współczesna automatyzacja i robotyzacja a człowiek*”, w którym przedstawiono rozwój automatyzacji i robotyzacji z uwzględnieniem szans i zagrożeń związanych z tym zjawiskiem.
Następny artykuł zatytułowany jest *„Wykorzystanie nowoczesnych technologii w zarządzaniu fir-*

mą ochroniarską – skuteczność dronów w działaniach monitorujących”, a jego autorami są Piotr Dąbrowski, Jędrzej Dąbrowski i Ewa Kardasz. Artykuł ten poświęcony jest aktualnej sytuacji firm ochroniarskich na rynku pracy. Opisuje zadania firm ochroniarskich, a także czynniki składające się na dobry wizerunek owych przedsiębiorstw. Dalej mówi o potrzebie wprowadzania innowacyjnych technologicznych rozwiązań w firmach ochroniarskich. Wymienia i charakteryzuje tego typu rozwiązania, takie jak: kamery wizyjne, kamery termowizyjne, kamery z funkcją identyfikacji twarzy, czujniki ruchu, aplikacje mobilne oraz bezzałogowe statki powietrzne. Kolejno opisuje liczne możliwości wykorzystania dronów przez firmy ochroniarskie, a także wykazuje, że urządzenia te pozwalają na realne oszczędności w przedsiębiorstwach.

Ostatnie dwie prace tego numeru, to artykuły, których autorami są przede wszystkim studenci Wrocławskiej Wyższej Szkoły Informatyki Stosowanej. W pracy „*Komentarze w kodach wybranych programów*”, którego autorami są Sviatoslaw Skhut, Kateryna Iholkina i Piotr Kardasz zwrócono szczególną uwagę na fakt, że nie tylko sam kod programu jest ważny, ale równie istotne są stosowane komentarze i dokumentacja, które powinny być prowadzone w taki sposób, aby kolejni programiści pracujący nad kodem nie mieli trudności z rozeznaniem się w jego treści i funkcjonalności. Artykuł ten stanowi kompendium wiedzy na temat dobrych zasad w programowaniu..

To wydanie Biuletynu zamyka artykuł zatytułowany „*Środowiska i narzędzia związane z wytwarzaniem oprogramowania*” autorstwa Michała Kulisia, Jarosława Jazienickiego i Patryka Stachury porusza problematykę i kwestie związane ze środowiskami, narzędziami oraz szeroko pojętym wytwarzaniem oprogramowania. W artykule przedstawiono także przykłady środowisk i narzędzi, które są obecnie najczęściej wykorzystywane

Korzystając z okazji chciałem poinformować, zwłaszcza Studentów WWSIS, że planuję wydać specjalny numer poświęcony pracom wykonanym przez naszych Studentów. Jeśli realizujecie projekty z obszaru szeroko rozumianej informatyki, to zachęcam do publikowania ich rezultatów na łamach naszego wspólnego, uczelnianego Czasopisma.

Serdecznie zachęcam do nadsyłania artykułów do kolejnego numeru.

Tomasz Długosz

Analiza dopasowania modelu do obiektu po etapie segmentacji obrazu wykorzystującego działy wodne (*ang. Watershed*) przy rozpoznawaniu barw obrazów.

Analysis of the pattern matching to the object after the Watershed image segmentation for image color recognition.

Paweł Iljaszewicz[1]

Streszczenie: W artykule przedstawiono segmentację obrazu bazująca na wykorzystaniu algorytmu działów wodnych (*ang. Watershed algorithm*), które zdobywa w ostatnich latach coraz szersze uznanie, jako skuteczna metoda detekcji obiektów. Wykorzystujemy fakt, iż poszukiwany obiekt powinien wykazywać podobieństwo geometryczne dotyczące kształtu, rozmiaru i położenia. Następnie połączenie tych cech z pobranym obrazem przetwarzanym na bieżąco pozwoli na zwrócenie współrzędnych prostokąta, który otoczy największy obiekt o wybranym kolorze. Celem proponowanego algorytmu będzie umożliwienie dokładnego dopasowania zadanego modelu do poszukiwanego obiektu, co w konsekwencji pozwoli na skuteczną detekcję na wszystkich obrazach z danej serii tematycznej. Omówiono również model wokselowy obiektu. Badania oparto na bibliotekach OpenCV.

Abstract: The article deals with the segmentation and recognition of the image based on the use of Watershed algorithm Watershed algorithm has gained ever greater recognition as an effective method of detecting objects in recent years. We use the fact that the object sought should show the geometric similarity of the shape, Size and position. Then combining these features with the downloaded image to be processed will allow you to return the coordinates of the rectangle, which will the largest object of the selected color. The purpose of the proposed algorithm will be to allow an exact match of the specified model to the object you are looking for, which will result in effective detection of all images from the given series. Also discusses the voxel object model. The study was based on OpenCV libraries.

Słowa kluczowe: OpenCV, analiza obrazu, działy wodne, thermography, dron, RGB, HSV, Model wokselowy obiektu.

Keywords: OpenCV, image analysis, watershed, termovision, drone, RGB, HSV, Voxel OpenCV model,

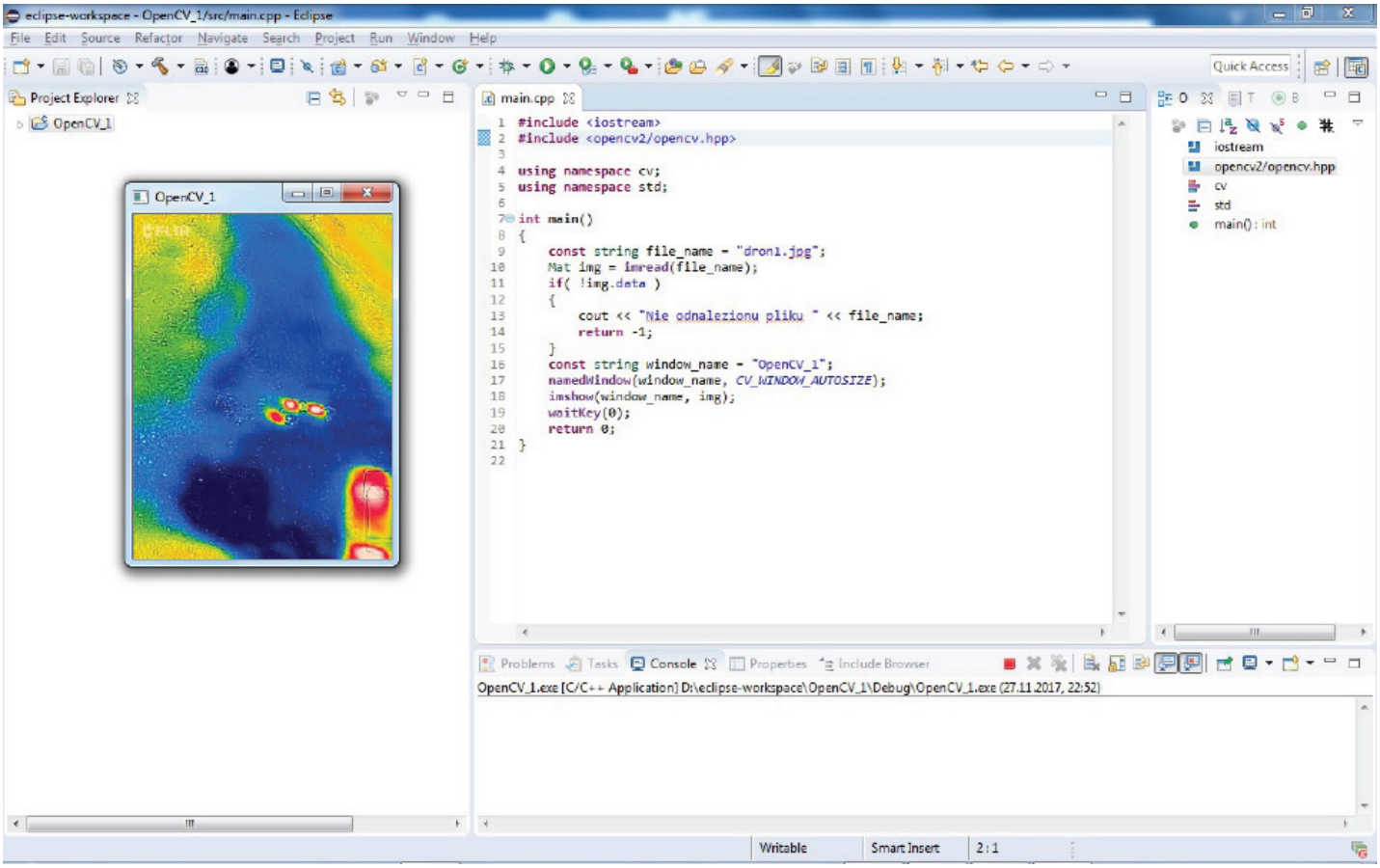
Wstęp

Celem niniejszej publikacji jest przedstawienie zastosowania bibliotek OpenCV wspomagających analizę barw obrazów uzyskiwanych podczas lotów dronem na różnych wysokościach w zależności od ich rozdzielczości. Film został wykonany kamerą termowizyjną.
Zaprezentowane zostaną przekształcenia RGB na HSV wykorzystujące bibliotekę opengl. Szczegółowo opisano mechanizm przejścia z jednej przestrzeni RGB do drugiej HSV. Przedstawiono wyniki konwersji na zdjęciu pobranym z filmu. Następnie
Zastosowanie modelu HSV i jego przekształcenie z RGB w OpenCV
OpenCV (*ang. Open Source Computer Vision Library*) - biblioteka algorytmów wizyjnych komputerowych, przetwarzania obrazu i algorytmów numerycznych do ogólnego przeznaczenia open source. Realizowane w C / C ++, jest również opracowany dla Python, Java, Ruby, Matlab, Lua i innych językach. [1] Można go swobodnie używać w celach akademickich i komercyjnych - rozpowszechniany jest na warunkach licencji BSD . Pozwala na optymalizowanie aplikacji w czasie rzeczywistym. Jest niezależna od systemu, sprzętu ani menadżera okienek. Zawiera nisko i wysokopoziomowy interfejs programowania aplikacji API (*ang. application programming interface*).[2]

1.1 Wczytanie obrazu

Korzystamy ze środowiska Eclipse[3] Pełny opis znajdziemy w dokumentacji OpenCV[4].
Program pisany jest w języku C++. Na rysunku poniżej przedstawiono wynik wczytania obrazu będącego klatką z filmu z drona.

1. Instytut Techniki Cieplnej Politechniki Śląskiej, ul. Konarskiego 22 44-100 Gliwice, email: pawel.iljaszewicz@polsl.pl
2. http://www.intel.com/technology/computing/opencv/
3. http://www.eclipse.org/
4. https://docs.opencv.org/2.4/doc/tutorials/introduction/linux_eclipse/linux_eclipse.html

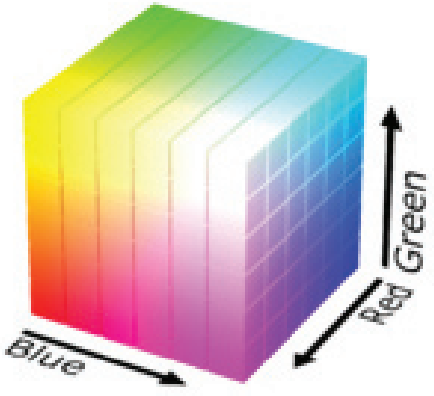


Ryc. 1 Wczytywanie obrazu OpenCV

W OpenCV obraz jest przechowywany w klasach Mat (Mat img = imread(file_name), które mają budowę wzorowaną na macierzach. W klasach można przechowywać również macierze a OpenCV udostępnia szereg funkcji pozwalających na wykonywanie działań na nich. Kolor zapisywany jest w formacie BGR odwrotnym od RGB (*ang. Red, Green, Blue*). Model ten jest rozpowszechniony przy zapisie i wyświetlaniu obrazów w telewizorach, kamerach aparatach itp.[4]

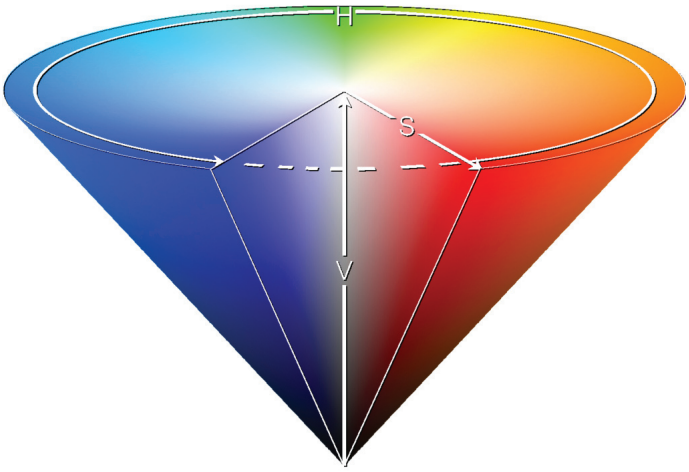
1.2 Przejście z przestrzeni RGB do HSV

Model RBG to przestrzeń o trzech wymiarach, gdzie każdemu punktowi odpowiada współrzędna definiowana względem osi R, G, B. Na rysunku poniżej przedstawiono osie i odpowiadające im punkty kolorów.



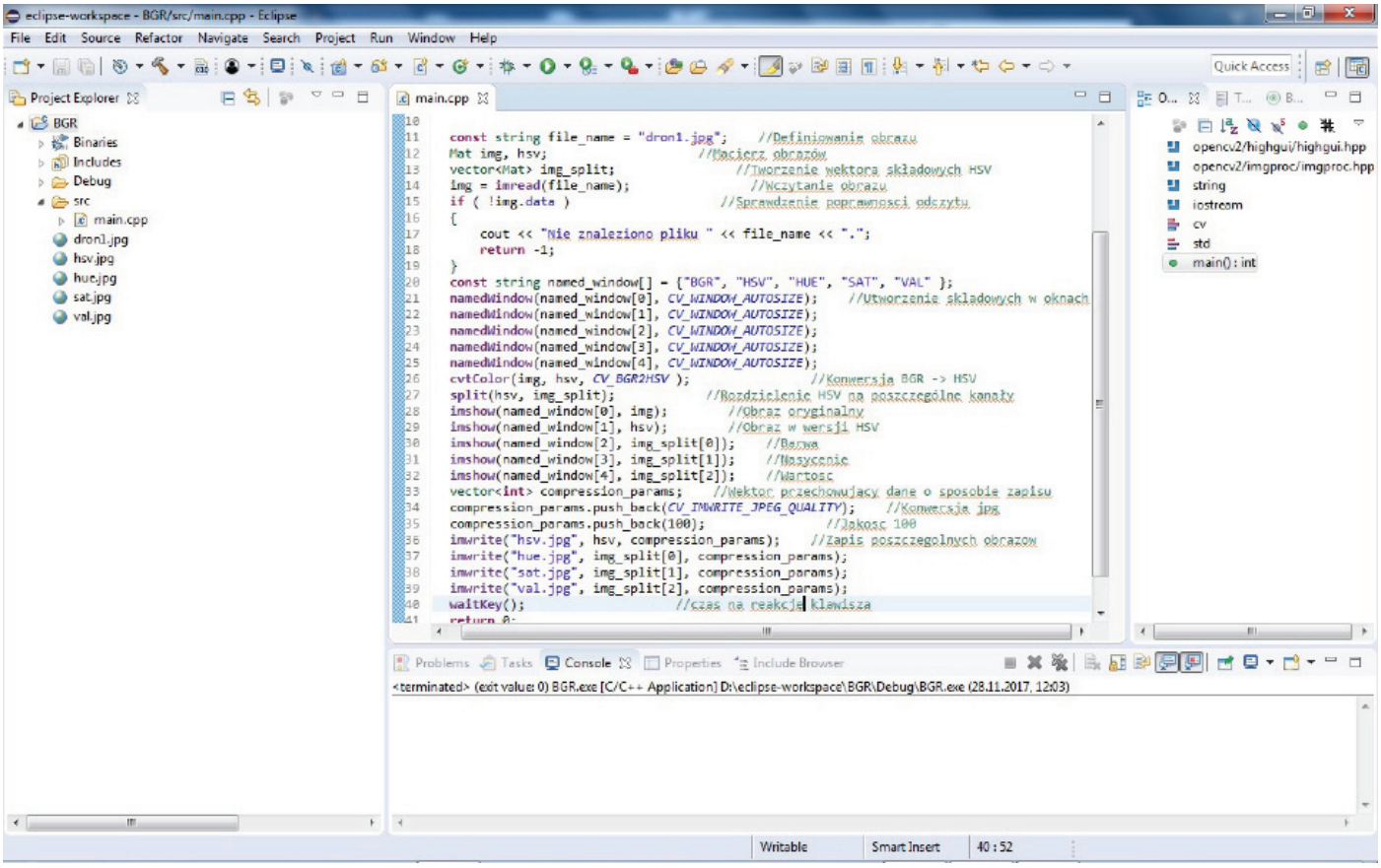
Ryc. 2 Model RGB

Widzimy, że aby zdefiniować kolor elementy macierzy Mat są definiowane przez 3 wartości odpowiadające kolorom B, G, R.
W przypadku modelu HSV (*ang. Hue Saturation Value*) przedstawionym na rysunku poniżej



Ryc. 3 Stożek przestrzeni barw HSV

składowa H barwa (hue) określana jest przez kąt od 0 do 360 stopni. Składowa S nasycenie (saturation), przez odległość od środka na promieniu podstawy, im mniejsza wartość tym bardziej biały kolor. Składowa V wartość (value) definiowana przez wysokość stożka, określa jasność koloru, mniejsza wartość odpowiada ciemniejszemu kolorowi. [3]



Ryc. 5 OpenCV konwertuje RGB na HSV

Jak widać, aby określić kolor wystarczy jedna składowa H. Pokazuje to rzut podstawy stożka.

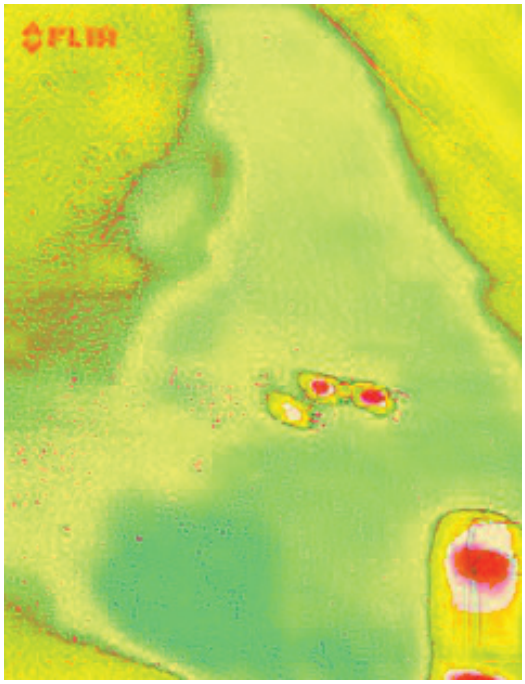
Ryc. 4 Definiowanie koloru żółtego, jako zakres kąta

Gdzie kolor żółty jest definiowany przez odpowiedni zakres kąta.

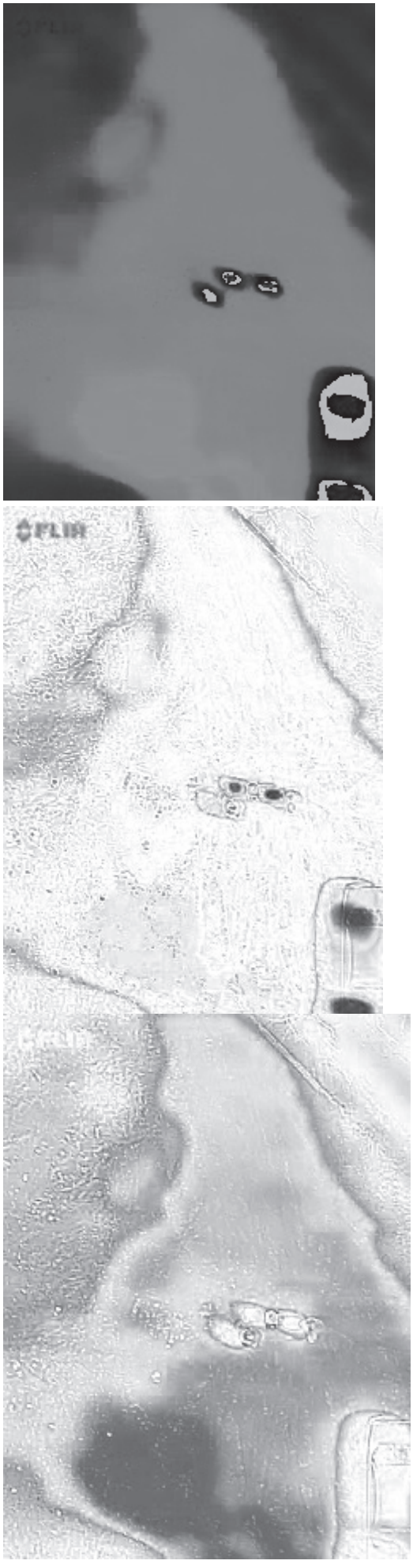
1.3 OpenCV przejście przestrzeni barw

W OpenCV w prosty sposób dokonuje się konwersji. Odpowiedzialna za to funkcja cvtColor(img, hsv, CV_BGR2HSV); pozwala przechodzić pomiędzy obrazami. Pierwszy argument img to obraz, który poddajemy przekształceniu, drugi argument oznacza obraz po konwersji natomiast trzeci CV_BGR2HSV oznacza przejście z BGR na HSV. Oczywiście istnieją inne konwersje szczegółowo opisane w dokumentacji[5]. Na rysunku powyżej pokazano zrzut ekranu pokazujący program konwertujący RGB na HSV.

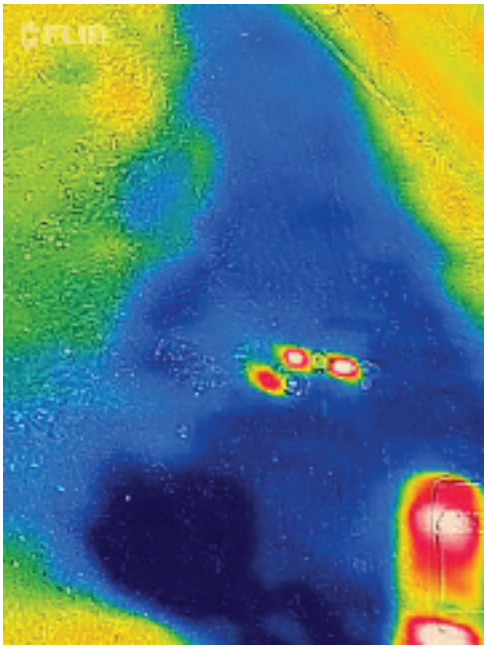
Funkcja split (hsv, img_split); rozdziela obraz na składowe wynikowe przedstawione na rysunkach poniżej. Widzimy obraz wyjściowy a następnie składowe po konwersji. Pierwszy obraz przedstawia obraz *HSV*, drugi składową *hue*, trzeci składową *sat*, czwarty składową *val*.



5. https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html



Ryc. 6 Konwersja obrazu wejściowego na składowe HSV, hue, sat i val.



Ryc. 7 Obraz wejściowy img

Warto zwrócić uwagę, iż różne odcienie koloru niebieskiego stanowią ten sam odcień na obrazie reprezentującym barwę *hue*. Te formuły odzwierciedlają pewne osobliwości wartości HSV. Jeśli R= G = B, to wartość H nie ma znaczenia i jest ustalona przez definicję H = 0. Jest to oczywiste. Ponieważ jeśli S = 0 (kolor nienasycony), to chromatyczność znajduje się w środkowej szarej linii, Barwa jest tak nieistotna i nie może być określona w żaden znaczący sposób. Jeśli R = G = B = 0, to S nie ma znaczenia, a z definicji S = 0 jest ustalona. Jeśli wszystkie trzy wartości RGB są "zerowe", to jest obraz czarny i nasycenia koloru traci znaczenie. To samo dotyczy, jednak nie w przypadku MAX = MIN = 1, czyli biały, wartość podana tutaj przez formułę 0 ma znaczenie elementarne, jak widać w powyższej formie stożka. Niezdefiniowane wartości są wypełnione, jako "zero" ze względów obliczeniowych.
W równaniach poniżej przedstawiono podstawowe zależności pomiędzy RGB a HSV

$$H := \begin{cases} 0, & jeśli\ MAX = MIN \Leftrightarrow R = G = B \\ 60^{\circ} \cdot \left(0 + \dfrac{G-B}{MAX-MIN}\right), & jeśli\ MAX = R \\ 60^{\circ} \cdot \left(2 + \dfrac{B-R}{MAX-MIN}\right), & jeśli\ MAX = G \\ 60^{\circ} \cdot \left(4 + \dfrac{R-G}{MAX-MIN}\right), & jeśli\ MAX = B \end{cases} \quad (1)$$

$$S_{\mathrm{HSV}} := \begin{cases} 0, & jesli\ MAX = 0 \Leftrightarrow R = G = B = 0 \\ \dfrac{MAX-MIN}{MAX}, & inaczej \end{cases} \quad (2)$$

$$V := MAX$$
$$L := \frac{MAX + MIN}{2} \quad (3)$$

Możemy powiedzieć, że kolor przestrzeni jest kombinacją modelu kolorów i funkcji mapowania (ta definicja jest dobrze znana). Dla modelu kolorów, możemy zrozumieć każdy model matematyczny, który może być używany do reprezentowania kolory, jako liczby (np. RGB (255, 0, 0) reprezentuje kolor czerwony). Dla funkcji mapowania, możemy zrozumieć każdą funkcję, która może mapować model kolorów do absolutnej przestrzeni kolorów, w celu podłączenia tego systemu kolorów do rzeczywistego świata, dzięki czemu można go używać.
Naszym celem jest wizualizacja każdego z trzech kanałów tych przestrzeni kolorystycznych: RGB, HSV. Ogólnie żaden z nich nie jest w stanie bezwzględne odtworzyć rzeczywisty kolor. Są to jedynie inne systemy kodowania informacji RGB. Nasze obrazy będą odczytywane w BGR (niebiesko-zielono-czerwony), ze względu na domyślne ustawienia OpenCV. Dla każdej z tych przestrzeni kolorów jest funkcja mapowania i można je znaleźć dokumentacji OpenCV cvtColor.[6]

Metoda modelu wokselowego

Woksel to najmniejszy element przestrzeni w grafice 3D (*ang. Voxel volumetric pixel element*) Voxel reprezentuje wartość na regularnej siatce w przestrzeni trójwymiarowej[7]. Podobnie jak piksele w bitmapie, same woksele zazwyczaj nie mają swojej pozycji (współrzędnych) wyraźnie zakodowanej wraz z ich wartościami. Zamiast tego systemy renderujące określają pozycję woksela na podstawie jego pozycji względem innych wokseli (tj. jego pozycja w strukturze danych, która tworzy pojedynczy obraz wolumetryczny). W przeciwieństwie do pikseli i wokseli, punkty i wielokąty często są jawnie reprezentowane przez współrzędne ich wierzchołków. Bezpośrednią konsekwencją tej różnicy jest to, że wielokąty mogą efektywnie reprezentować proste struktury 3D z dużą ilością pustej lub jednorodnie wypełnionej przestrzeni, podczas gdy woksele doskonale reprezentują regularnie próbkowane przestrzenie, które są niejednorodnie wypełnione.[2]
Metoda implementująca woksele jest techniką śledzenia wielosegmentowego obiektu w przestrzeni trójwymiarowej. Każdy z obiektów ma jednoznacznie określoną objętość. Pozwala to na jego wysegmentowanie, pomimo, iż może być przesłonięty przez inne obiekty. Znając przestrzeń zajmowaną przez obiekt łatwo można go rzutować na zdefiniowaną płaszczyznę. Łatwo można wyselekcjonować i śledzić obiekt gdyż ma przypisany do siebie indywidualny numer.
Obiekt jest na tej podstawie śledzony przez odpowiedni mechanizm, dopóki nie opuści zakładanego obszaru sceny. Segmentacja sceny z wykorzystaniem modelu wokselowego w prosty sposób identyfikuje obiekt i przypisuje mu indywidualny znacznik, w odróżnieniu do innych przekształceń macierzy homograficznych i klasycznych metodach detekcji obszaru poruszającego się i jego klasyfikacji.
Przykładowa implementacja używająca biblioteki OpenCV opengl suport gdzie chmura punktów jest reprezentowana w Voxel OpenCV. Każda z 8 pozycji definiuje trójwymiarowy wektor uzyskany podczas segmentacji obiektu.[8]

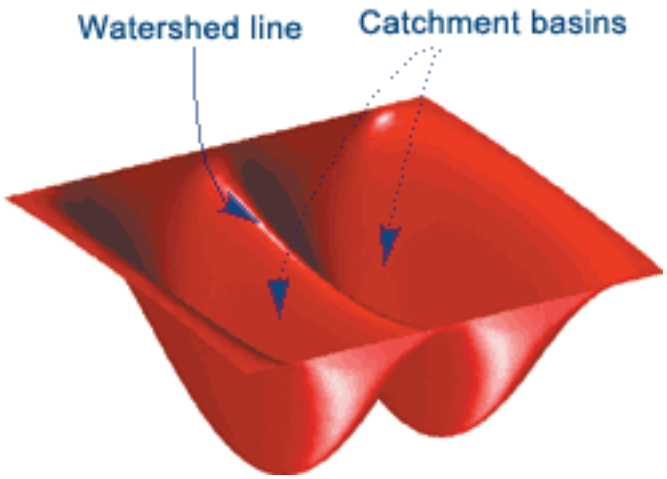
Tab. 1 Implementacja chmury punktów Voxel OpenCV

```
#include <opencv2/opencv.hpp>
// shame, needs windows.h on win, else problem with
APIENTRY
//#include <windows.h>
#include <GL/gl.h>

using namespace std;
using namespace cv;

float pts[8*3] = {
   0.3255598,0.2123329,0.4342422,
   0.234444,0.4323432,0.4324234,
   0.668886,0.387868,0.51884323,
   0.686878,0.567577,0.7675777,
   0.234565,0.2675675,0.356777,
   0.3456567,0.523435,0.3446577,
   0.237577,0.2343434,0.324344433,
   0.24353,0.35765756,0.2455345
};

void on_opengl(void* param)
{
   glLoadIdentity();
   glPointSize(3);
   glBegin(GL_POINTS);
   for ( int i=0; i<8; i++ )
   {
      glColor3ub( i*20, 100+i*10, i*42 );
      glVertex3fv(&pts[i*3]);
   }
   glEnd();
}

int main(int argc, char *argv[])
{
      namedWindow("3d",CV_WINDOW_OPENGL|CV_
WINDOW_NORMAL);
   setOpenGlDrawCallback("3d",on_opengl,0);
   waitKey(0);
   return 0;
}
```

6. www.opencv.org
7. https://github.com/clickteam-plugin/OpenCV/blob/master/Extension/Main/Voxel.h-
8. https://docs.opencv.org/2.4.13.2/modules/core/doc/basic_structures.html

Algorytm działów wodnych

Algorytm działu wodnego jest klasycznym algorytmem służącym do segmentacji i jest szczególnie przydatny przy wydobywaniu z obrazu obiektów, dotykających lub nakładających się. Każdy obraz w skali szarości może być postrzegany jako powierzchnia topograficzna, gdzie wysoka intensywność oznacza szczyty i wzgórza, a niska intensywność oznacza doliny. W algorytmie zaczyna się wypełniać każdą izolowaną dolinę (lokalne minima) różnokolorową wodą (etykiety) tworząc tzw. zlewiska (ang. catchment basins), gdy woda wznosi się, w zależności od pobliskich szczytów, wody z różnych dolin, oczywiście z różnymi kolorami zaczynają się łączyć. Aby tego uniknąć, buduje się bariery w miejscach, gdzie woda się łączy. Kontynuuje się pracę nad napełnianiem wody i budowaniem barier, dopóki wszystkie szczyty nie znajdą się pod wodą. Następnie utworzone bariery dają wynik segmentacji. To jest "filozofia" stojąca za działem wodnym.
Możesz odwiedzić stronę CMM w dziale wodnym, aby zrozumieć ją za pomocą animacji.
Ale to podejście daje wynik przekrojowy z powodu szumu lub innych nieprawidłowości w obrazie. Tak, więc OpenCV wdrożył algorytm wodny oparty na markerach, w którym określa się, które punkty doliny mają zostać połączone, a które nie. Jest to interaktywna segmentacja obrazu. To, co robimy, to dawanie różnych etykiet dla naszego znanego nam obiektu. Oznacza się region, którego jesteśmy pewni, że jest pierwszym planem lub obiektem o jednym kolorze (lub intensywności), oznacza się region, z którego jesteśmy pewni, że jest tłem lub nie jest obiektem o innym kolorze, i ostatecznie region, którego nie jesteśmy pewni, oznacz to, jako 0. To jest nasz marker. Następnie zastosuj algorytm działu wodnego. Następnie nasz marker zostanie zaktualizowany o etykiety, które daliśmy, a granice obiektów będą miały wartość -1.



Ryc. 8 Wypełnienia algorytmu działem wodnym i zlewiskiem

Poniżej przykład algorytmu[9]. Rozpatrujemy pojedynczy obraz. W pierwszej kolejności ładowany jest obraz „image”.

Tab. 2 Algorytm działów wodnych OpenCV

```
#include <opencv2/core/utility.hpp>
#include "opencv2/imgproc.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include <cstdio>
#include <iostream>
using namespace cv;
using namespace std;
static void help()
{
cout << "\nThis program demonstrates the famous water-
shed segmentation algorithm in OpenCV: watershed()\n"
"Usage:\n"
"./watershed [image_name -- default is ../data/fruits.jpg]\n"
<< endl;
cout << "Hot keys: \n"
"\tESC - quit the program\n"
"\tr - restore the original image\n"
"\tw or SPACE - run watershed segmentation
algorithm\n"
"\t\t(before running it, *roughly* mark the areas to seg-
ment on the image)\n"
"\t (before that, roughly outline several markers on the
image)\n";
}
Mat markerMask, img;
Point prevPt(-1, -1);
static void onMouse( int event, int x, int y, int flags, void*
)
{
if( x < 0 || x >= img.cols || y < 0 || y >= img.rows )
return;
if( event == EVENT_LBUTTONUP || !(flags & EVENT_
FLAG_LBUTTON) )
prevPt = Point(-1,-1);
else if( event == EVENT_LBUTTONDOWN )
prevPt = Point(x,y);
else if( event == EVENT_MOUSEMOVE && (flags &
EVENT_FLAG_LBUTTON) )
{
Point pt(x, y);
if( prevPt.x < 0 )
prevPt = pt;
line( markerMask, prevPt, pt, Scalar::all(255), 5, 8, 0 );
line( img, prevPt, pt, Scalar::all(255), 5, 8, 0 );
prevPt = pt;
imshow("image", img);
}
}
int main( int argc, char** argv )
{
cv::CommandLineParser parser(argc, argv, "{help h | | }{
@input | ../data/dron1.jpg | }");
if (parser.has("help"))
{
help();
```

9. https://docs.opencv.org/3.3.1/d8/da9/watershed_8cpp-example.html

```
return 0;
}
string filename = parser.get<string>("@input");
Mat img0 = imread(filename, 1), imgGray;
if( img0.empty() )
{
cout << "Couldn'g open image " << filename << ". Usage:
watershed <image_name>\n";
return 0;
}
help();
namedWindow( "image", 1 );
img0.copyTo(img);
cvtColor(img, markerMask, COLOR_BGR2GRAY);
cvtColor(markerMask, imgGray, COLOR_GRAY2BGR);
markerMask = Scalar::all(0);
imshow( "image", img );
setMouseCallback( "image", onMouse, 0 );
for(;;)
{
char c = (char)waitKey(0);
if( c == 27 )
break;
if( c == 'r' )
{
markerMask = Scalar::all(0);
img0.copyTo(img);
imshow( "image", img );
}
if( c == 'w' || c == ' ' )
{
int i, j, compCount = 0;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
findContours(markerMask, contours, hierarchy, RETR_
CCOMP, CHAIN_APPROX_SIMPLE);
if( contours.empty() )
continue;
Mat markers(markerMask.size(), CV_32S);
markers = Scalar::all(0);
int idx = 0;
for( ; idx >= 0; idx = hierarchy[idx][0], compCount++ )
drawContours(markers, contours, idx, Scalar::all(comp-
Count+1), -1, 8, hierarchy, INT_MAX);
if( compCount == 0 )
continue;
vector<Vec3b> colorTab;
for( i = 0; i < compCount; i++ )
{
int b = theRNG().uniform(0, 255);
int g = theRNG().uniform(0, 255);
int r = theRNG().uniform(0, 255);
colorTab.push_back(Vec3b((uchar)b, (uchar)g, (uchar)r));
}
double t = (double)getTickCount();
watershed( img0, markers );
t = (double)getTickCount() - t;
printf( "execution time = %gms\n", t*1000./getTickFrequ-
ency() );
Mat wshed(markers.size(), CV_8UC3);
// paint the watershed image
for( i = 0; i < markers.rows; i++ )
for( j = 0; j < markers.cols; j++ )
{
int index = markers.at<int>(i,j);
if( index == -1 )
wshed.at<Vec3b>(i,j) = Vec3b(255,255,255);
else if( index <= 0 || index > compCount )
wshed.at<Vec3b>(i,j) = Vec3b(0,0,0);
else
wshed.at<Vec3b>(i,j) = colorTab[index - 1];
}
wshed = wshed*0.5 + imgGray*0.5;
imshow( "watershed transform", wshed );
}
}
return 0;
}
}
```

Wnioski

Zastosowany algorytm pozwala na wyodrębnienie poszczególnych obiektów. Niestety przy zdjęciach termowizyjnych obszary przedstawiające obiekty o tej samej temperaturze mają identyczną barwę. Należałoby więc połączyć segmentację opartą na badaniu konturu razem z zastosowanym algorytmem. Celem niniejszej publikacji było jedynie pokazanie zastosowań bibliotek OpenCV.

Literatura

[1]. Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue. pp. 40:40–40:56. doi:10.1145/2181796.2206309 CHENG, H. D. et al. Color image segmentation: advances and prospects. Pattern Recognition, v. 34, p. 2259-228, 2001.

[2]. Kaufman, A. and Shimony, E., '3D Scan-Conversion Algorithms for Voxel-Based Graphics', Proc. ACM Workshop on Interactive 3D Graphics, Chapel Hill, NC, October 1986, 45-76.

[3]. FR patent 841335, Valensi, Georges, "Procédé de télévision en couleurs", published 1939-05-17, issued 1939-02-06 [4].THEODORIDIS, S.; K, K. Pattern Recognition. [S.l.]: Elsevier Inc, 2009.

[5]. PAL, N. R.; PAL, S. A review on image segmentation techniques. Pattern Recognition, v. 26, p. 1277-1294, 1993.

[6]. C. Rother, V. Kolmogorov, and A. Blake.“GrabCut”—Interactive foreground extraction using iterated graph cuts. In SIGGRAPH, pages 309–314, 2004.

[7]. A. K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker

which yields a new algorithm. In ICCV, 2007
Strony internetowe:
[8]. https://docs.opencv.org/3.3.1/d8/da9/watershed_8cpp-
-example.html.
[9]. https://github.com/clickteam-plugin/OpenCV/blob/
master/Extension/Main/Voxel.h-
[10]. www.opencv.org
[11]. https://docs.opencv.org/2.4/doc/tutorials/introduc-
tion/linux_eclipse/linux_eclipse.html
[12]. http://www.eclipse.org
[13]. http://www.intel.com/technology/computing/
opencv/
[14]. https://docs.opencv.org/2.4.13.2/modules/core/doc/
basic_structures.html
[15]. https://docs.opencv.org/2.4/modules/imgproc/doc/
miscellaneous_transformations.html

Jak nie porównywać wyników badań bioelektromagnetycznych

HOW TO NOT COMPARE RESULTS IN BIOELECTROMAGNETIC STUDIES

Tomasz Długosz[1]

Streszczenie: Artykuł poświęcony jest problemom związanym z porównywaniem wyników badań bioelektromagnetycznych. Można zauważyć, że badania poświęcone wpływom pola elektromagnetycznego, zwłaszcza na obiekty biologiczne, dają niejednoznaczne wyniki, które ciężko porównać między różnymi ośrodkami naukowymi. Wiele błędów wynika z technicznych aspektów tego typu eksperymentów. W pracy przedstawiono wybrane z nich. W badaniach nie wykorzystywano obiektów biologicznych, a jedynie ich numeryczne modele.

Abstract: The paper is devoted to the problem of difficulty in results comparision of bioelectromeganetic experiments. It may be noticed that many studies which are devoted to biological effects of electromagnetic field exposure on biological objects can not be compared to each other. There may be many reasons responisble for that state. Certainly one of them are different sources of uncertainty resulting from the technical aspects of this type of research. One such source is influence of exposure system on tested object.

Słowa kluczowe: badania bioelektromagnetyczne, metody numeryczne, pole elektromagnetyczne, dokładność badań, układy ekspozycyjne

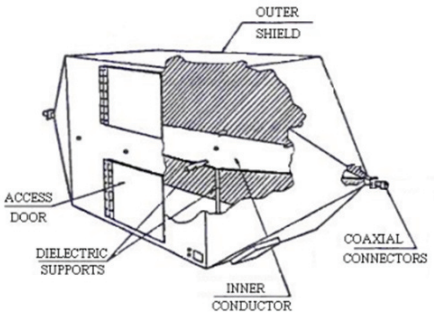
Keywords: bioelectromagnetic studies, electromagnetic field, accuracy of experiments, exposure systems, numerical methods

Introduction

Increasing usage of electronic equipment and wireless telecommunication systems in almost all aspects of our live has coused interested of society about electromagnetic field (EMF). Whole environment is intentionally or unintentionally exposed to EMFs. This exposure creates a risk which is current subject of studies in bioelectromagnetics experiments. Especially important issues are biomedical studies exploring the effects of EMFs on human [1]-[7].
One of the most important problem in bioelectromagnetics studies is accuracy one. Bioelectromagnetic research is one of the least accurate and difficult to perform. In many cases the tests are performed when the EMF exposure is significantly different from the one to which objects are exposed to in real life. In order to improve the accuracy it is necessary to increase the comparability of results obtained in different labs. Estimates made by the author show that due to interaction between the tested objects and the exposure system and among objects themselves, errors may exceed even 100% [8][9]. These phenomena are the reason for significant differences in the results of research done in different research centres.
Tha aim of this paper is to focus attention of experimenters on one phenomenon that is not taken into account in the majority of experiments and may lead to complete falisification results of experiments.

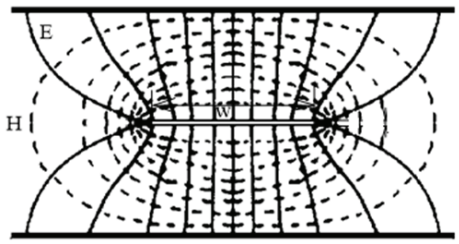
Object under test inside exposure system

Transverse Electro-Magnetic (TEM) line (Fig. 1) is one of the most popular exposure system that is very often used in electromagnetic compatibility tests or in bioelectromagnetics experiments [9][10]. It may be used for antenna's calibration, electromagnetic compatibility investigations and biomedical studies.
TEM cells have many advantages, like the wide frequency range from DC to hundreds megahertz, good isolation from external environment, frequency independent field intensity, relatively small costs. But there are also some limitations: influence of line on object, mutual interactions between the cell and object, problems with larger objects testing at high frequencies, non-ideal EMF distribution, resonances and the presence of higher modes.



a)

1. Wrocław School of Information Technology ks. M. Lutra 4, 54-239 Wrocław, Poland e-mail: tdlugosz@horyzont.eu



b)

Fig. 1. TEM line as exposure system: a) construction, b) a cross section of opensided line

Placing any object with conductivity different than zero in the EMF causes certain losses. If the values of the electric field intensity and conduction current density are known, then the power loss that is absorbed by the objects is described by the formula

$$P_{abs} = \int_V EJ\,dV \tag{1}$$

where:
P_{abs} – absorbed power,

E – electric field density vector,

J – current density vector,

V – volume of the object.

Substituting into (1) as J

$$J = \sigma E \tag{2}$$

where:

σ - conductivity of tested object

and making simple transformations, we get the power absorbed by an object placed in the EMF, given by

$$P_{abs} = \sigma \int_V E\,dV \tag{3}$$

Formula (3) is true if the tested object is homogeneous. Otherwise, a formula that takes into account the quasihomogenous volumes VN in all N areas of different conductivity is used

$$P_{abs} = \sigma_1 \int_{V_1} |E_1|^2\,dV_1 + \sigma_2 \int_{V_2} |E_2|^2\,dV_2 + \mathsf{K} + \sigma_N \int_{V_N} |E_N|^2\,dV_N \tag{4}$$

Calculating absorbed power allows to see how exposure system influences on tested object [9][11].

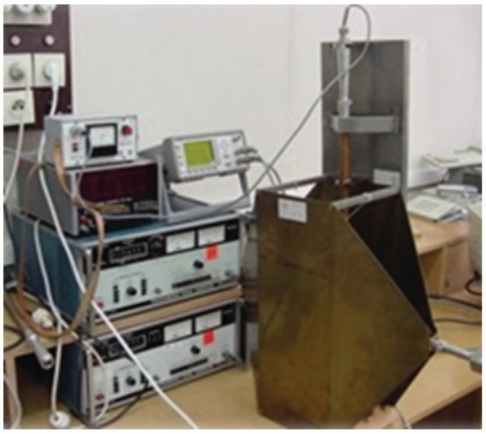
Materials and methods

It is well known that the primary tool for quantitative research is hands-on experimentation and measurements. Unfortunately, the tests are not always possible due to high complexity of the studied objects, lack of appropriate sensors or their inaccuracy. This is especially important in the measurement of EMF. It is worth mentioning that any physical quantity measured (i.e.: frequency) are performed with 10^{10}% accuracy, whereas the error in creating a standard EMF equals 5%-10%. That influences the test tools' accuracy whose error can't exceed the one of creating EMF. Further appears the question of ethics of such tests. Experiments examining EMF's influence on human body are acceptable with person's consent, but still controversial. The same applies to the use of animals for this type of research. Above arguments show that bioelectromagnetic testing is a challenge, and is often impossible to perform. This is where use of mathematical models and computer programs based on numeric methods comes in handy. These tools give us some insight on the expected results. Similar results from different numerical methods can be considered as exemplary and reliable

The most important feature and the biggest advantage of computer simulations using numerical methods is their ability to predict the behavior of the actual object based on its mathematical model. It is much easier and faster to perform computer simulations, rather than perform the measurements in real life conditions. Computer simulations are also extremely useful when the experiments are too dangerous to perform, i.e.: when the researched EMF can cause health issues or death of tested objects. Major drawbacks of computer simulations are restraints of computing resources and long duration of the calculations.

All presented in this paper results were obtained by Finite Element Method (FEM) and Finite Difference Time Domain method (FDTD) [12][13].

In the above simulations, real TEM line (Fig. 2a) was replaced by two conductive surfaces (Fig. 2b). Six models (I-VI) of different sizes of TEM line were used. They varied one dimension – distance (d) between plates (Fig. 3).



a)

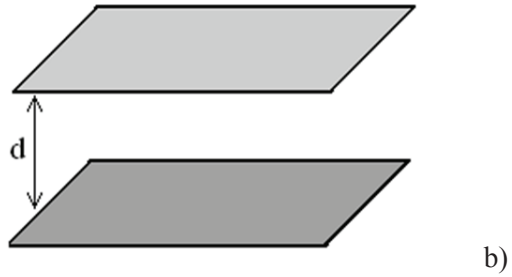
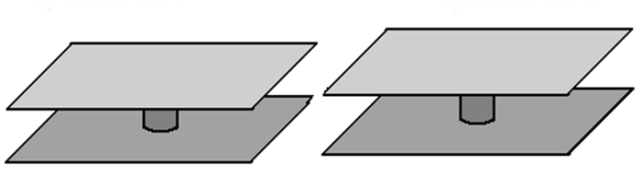
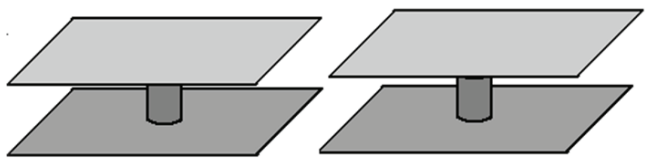


Fig. 2. TEM line: a) real exposure system b) simplified model

a) model no. I b) model no. II



c) model no. III d) model no. IV



e) model no. V f) model no. VI

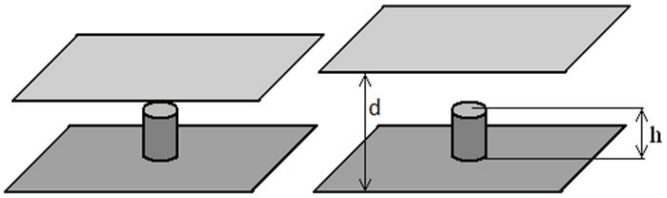


Fig. 3. TEM cell with tested object in function of distance between plates: a) d/h=1,0, b) d/h=1,2, c) d/h=1,4, d) d/h=1,6, e) d/h-1,8, f) d/h=2,0

In each case electric field inside E was the same 1 V/m. Inside those exposure systems an tested object was placed. It was simplified cylindrical heterogeneous model of a human. Its electrical parameters equal $\varepsilon = 80$, $\sigma = 0.84$ S/m.

Results

Results of calculations are shown in Fig. 4. It may be noticed that the size of the exposure system has a significant impact on the quantity of absorbed energy. The same tested object placed in the same EMF's conditions absorbed different portion of energy in each of exposure system.

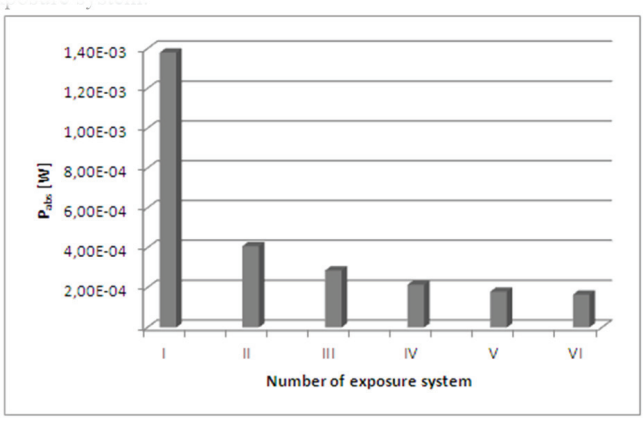


Fig. 4. Power absorbed by the same object placed in the same EMF within exposure system of different sizes

The results of changes in the power absorption as a function of the exposure and system's size are shown in Fig. 5. It is worth mentioning that when the plates of TEM line are close to the object the power absorbed is 30 times higher compared to the conditions of free space. Increase in the d/h ratio causes the absorbed power to decrease and approach asymptotically the value of absorbed power in free space, where the presence of metal plates is negligible. This condition is met for d/h ≈ 2.

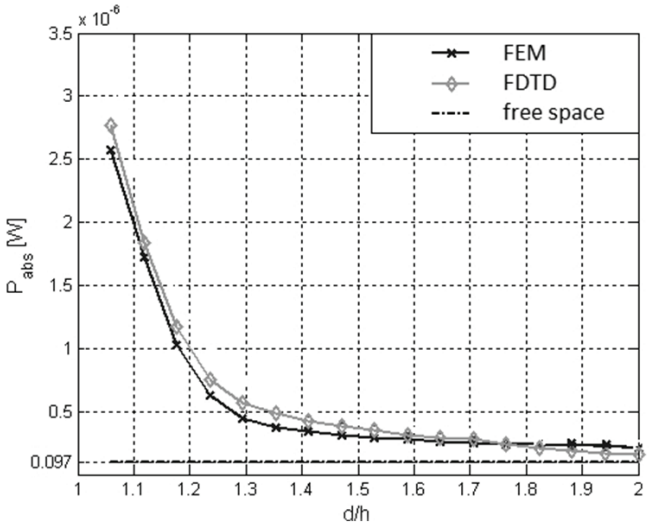


Fig. 5. The results of calculations of absorbed power by the cylindrical model of a human placed perpendicular to the plates of the walls

The estimations are the most primitive ones, however, they show a role of the conducting plates presence upon the absorption. Apart from the presence of couplings with the exposure system (plates) the same effect exists between objects (if more than one). Effect increases with frequency and complexity of tested object. The phenomenon losses it's importance for d≈2h.

Presented results show clearly that not only EMF parameters should be the same when we want to compare results. Also dimensions of exposure system play important role. When different sizes exposure systems are used than significant errors are made (Fig. 6).

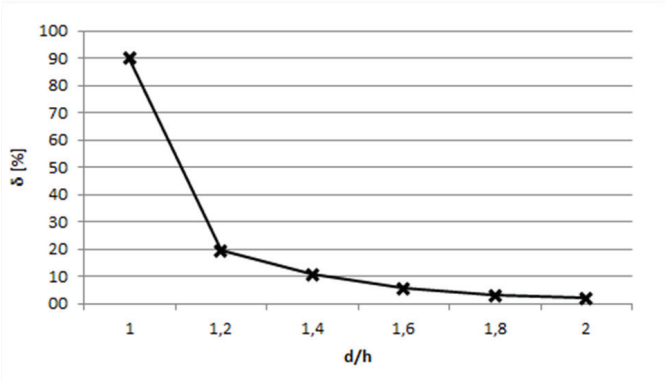


Fig. 6. Accuracy estimations of bioelectromagnetic experiment versus size of exposure system

Conclusion

A lot of studies is currently devoted to biological effects as a result of EMF exposure but very often they are irreproducible and contradictory. One of the reasons may be not taking into account uncertainty of such experiments. There are a lot of sources of errors in that kind studies. One of them is influence of exposure system on tested object what was presented in this paper.

It is important, to all of us, to start cooperation between biologists, physicians and engineers, because sources of errors in such experiemnts are twofold: technical [8] and biological [14].

REFERENCES

[1] Ibey, B. L., Roth, C.C., Ledwig, P. B., Payne J.A., Amato A.L., Dalzell D.R., Bernhard J.A., Doroski M.W., Mylacraine K.S., Seaman R.L., Nelson G.S., Woods C.W., "Cellular effects of acute exposure to high peak power microwave systems: Morphology and toxicology", *Bioelectromagnetics*, Vol. 37, Issue 3, 2016, pp. 141-151.

[2] Calvente, I., Pérez-Lobato, R., Núñez, M.I, Ramos R., Guxens, M., Villalba, J., Olea, N., Fernandez, M.F., "Does exposure to environmental radiofrequency electromagnetic fields cause cognitive and behavioral effects in 10-year-old boys?", *Bioelectromagnetics*, Vol. 37, Issue 1, 2016, pp. 25-36.

[3] Wuschech, H., Hehn, U., Mikus, E., Funk, R. H."Effects of PEMF on patients with osteoarthritis: results of a prospective, placebo-controlled, double-blind study", *Bioelectromagnetics*, Vol. 36, Issue 8, 2015, pp. 576-585.

[4] Varsier, N., Plets, D., Corre, Y., Vermeeren, G., Joseph, W., Aerts, S., Martens, L., Wiart, J., "A novel methods to assess human population exposure induced by a wireless cellular network", *Bioelectromagnetics*, Vol. 36, Issue 6, 2015, pp. 451-463.

[5] Markov, M. S., "Electromagnetic Fields in Biology and Medicine", *CRC Press*, 2015.

[6] Scientific Committee on Emerging Newly Identified Health Risks, "Opinion on Potential Helath Effetcs of Exposure to Electromagnetic Fields", *Bioelectromagnetics*, Vol. 36, Issue 6, 2015, pp. 480-484.

[7] Boriraksantikul, N., Bhattacharyya K. D., Whiteside, P.J.D., O'Brien, C., Kirawanich, P., Viator, J.A., Islam, N.E., "Case Study of High Blood Glucose Concentration Effects of 850 MHz Electromagnetic Fields Using Gtem Cell", *Progress In Electromagnetics Research B*, Vol. 50, 2012, pp. 55-44.

[8] Dlugosz, T., "Uncertainity Analysis of selected Sources of Errors in Bioelectromagnetic Investigations", *Bio-Medical Materials and Engineering* 1, 2014, pp. 609-617.

[9] Dlugosz, T., Trzaska, H., Mutual interactions in bioelectromagnetics, *The Environmentalist* 4, 2007, pp. 403-409.

[10] Crawford, M., L., "Generation of Standard EM Field Using TEM Transmission Cell", *IEEE Transactions on Electromagnetic Compatibility*, Vol. EMC-16, Issue 4, 1974, pp. 189-195.

[11] Dlugosz, T., Trzaska, H., "Proximity Effects in the Near-field EMF Metrology", *IEEE Transaction on Instrumentation and Measurement*, Vol. 57, Issue 11, 2008, pp. 626-630.

[12] Huebner, K.,H., Thornton, E., A., "The Finite Element Method for Engineers", *John Wiley and Sons*, 1982.

[13] Taflove, A., "Computational Electrodynamics: The Finite-Difference Time-Domain Method", *Artech House Antennas and Propagation Library*, 1996.

[14] Buchachenko, A., "Why Magnetic and Electromagnetic Effects in Biology Are Irreproducable and Contradictory?", *Bioelectromagnetics*, Vol. 37, Issue 1, 2016, pp. 1-13.

Human skills, czyli motywowanie pracowników i delegowanie zadań

„Human Skills - motivating employees and delegating task"

Michał Zubelewicz[1], Ewa Kardasz[2]

Streszczenie: Artykuł opowiada o zarządzaniu zasobami ludzkimi. Charakteryzuje takie pojęcia jak: *human skills*, motywowanie pracowników oraz delegowanie zadań. Podaje kilka różnych definicji motywowania pracowników. Opisuje sposoby na zmotywowanie pracowników (tu: środki przymusu, środki zachęty: materialne i niematerialne, środki perswazyjne). Przedstawia podstawowe kwestie związane z delegowaniem zadań. Kolejno artykuł wymienia sześć podstawowych porad dotyczących delegowania zadań, a także opisuje zalety delegowania zadań. W zakończeniu zostały podsumowane wszystkie rozważania oraz podane wnioski.

Słowa kluczowe:*human skills*, zarządzanie zasobami ludzkimi, motywowanie pracowników, delegowanie zadań, sposoby na zmotywowanie podwładnych, środki przymusu, środki zachęty, środki perswazji

Wstęp

Pojęcie *human skills* nierozerwalnie łączy się z pojęciem zarządzania zasobami ludzkimi. Jest to strategiczne i całościowe podejście do zarządzania najbardziej wartościowymi aktywami danego przedsiębiorstwa[9]. Za najbardziej wartościowy uznaje się czynnik ludzki – pracowników indywidualnych, a także zbiorowych (grupy pracownicze), bez których realizacja wyznaczonych celów nie byłaby możliwa [6]. Zarządzanie zasobami ludzkimi składa się na: przyjmowanie do pracy, dbanie o rozwój, adekwatne do pracy i umiejętności nagradzanie, a także kształtowanie odpowiednich relacji między pracownikami oraz między pracownikami a przełożonymi [4]. Istotnym elementem zarządzania zasobami ludzkimi jest motywowanie pracowników i delegowanie zadań, co szczegółowo opisane zostało poniżej. Tego typu kwestiami zajmują się menadżerowie, liderzy zespołów, w dużych firmach również specjaliści od zarządzania zasobami ludzkimi [4].

Motywowanie pracowników – co to takiego?

Motywowanie pracowników jest procesem, a także funkcją zarządzania, która reguluje zachowania pracowników tak, aby swoimi działaniami przyczyniali się do osiągania celów i sukcesu danej organizacji [2]. Jest to złożony proces regulujący i uruchamiający zachowania i działalność danego podwładnego [2]. Motywowanie pracowników jest pewnego rodzaju układem sił, które skłaniają osobę do zachowywania się w konkretny sposób. Jest to bardzo ważny czynnik, wpływający na to, czy pracownik: realizuje zadania przy włożeniu maksymalnego wysiłku; realizuje zadania w taki sposób, by nie zostać upomnianym przez przełożonego, realizuje zadania, ograniczając swoje starania do minimum [2].

Według Zdzisława Jasińskiego „Motywowanie to oddziaływanie przez rozmaite formy i środki na pracowników tak, aby ich zachowania były zgodne z wolą kierującego, aby zmierzały do zrealizowania postawionych przed nim zadań. Istotą motywacji jest kojarzenie subiektywnych dążeń pracowników z procesem realizacji zadań organizacji" [3]. Z kolei Elżbieta Skrzypek stwierdza, że „Motywacja stanowi siłę wewnętrzną, która pozwala osiągnąć sukces w każdej dziedzinie życia. Gwarantuje wspaniałą współpracę w zespole i możliwość realizacji trudnych celów. Motywowanie to umiejętność wykorzystywania energii dla rozwoju człowieka" [3]. Jeszcze inaczej proces motywowania przedstawia Tadeusz Kotarbiński „Chodzi o to, by człowiek robił chętnie to, co robić musi, by tego, co robić musi, nie robił dlatego, że musi, by w robieniu tego, co musi, znalazł upodobanie i dzięki temu pracę swą usprawnił wielokrotnie, okazując hojność w oddaniu się jej" [3].

Sposoby na zmotywowanie podwładnych

Wyróżnia się trzy podstawowe sposoby na zmotywowanie podwładnych. Są to: środki przymusu, środki zachęty oraz środki perswazji [5]. Środki przymusu to wszelkiego rodzaju rozkazy, czyli nakazy i zakazy. Zalicza się do nich również polecenia i rady przełożonego oraz zobowiązania własne pracownika, czyli zadania, które przyjął na siebie dobrowolnie, w chwili podjęcia pracy. Różnią się one od

1. OVB Michał Zubelewicz, Wczasowa 7, 58-310, Szczawno-Zdrój
2. Wyższa Szkoła Zarządzania i Coachingu Wydział Inżynieryjny al. Śląska 1 54-118 Wrocław

siebie rangą, sposobami oraz metodami realizacji. Środki przymusu są związane z podporządkowywaniem się woli przełożonego. W ich wykorzystywaniu nie bierze się pod uwagę oczekiwań i interesów pracownika. Ich wadą jest to, że są oparte na strachu i karaniu. Środki te nie powinny być często stosowane. Przynoszą one efekt przede wszystkich w sytuacjach zagrożenia, czyli takich, które wymagają szybkiego działania, pracy pod presją czasu [5].
Środki zachęty to innymi słowy obietnice dawane pracownikom, gdzie powiedziane zostaje, że w przypadku zastosowania się do danego zalecenia, terminowego zrealizowania jakiegoś zadania, nastąpią jakieś pozytywne konsekwencje, które nie będą równoznaczne z podstawowym wynagrodzeniem pracownika (np. będą oznaczały premię w postaci wyższego wynagrodzenia). Intencją środków zachęty jest zainteresowanie osoby motywowanej do lepszego wykonywania zadań, które prowadzić ma do realizacji jego założeń (tu: awans, podwyżka), a także do ulepszenia sprawności pracy organizacji. Środki te to wszystkie procesy wpływające na zmodyfikowanie i wykształtowanie pożądanych przez przełożonego zachowań oraz wzorców. Środki zachęty dzielą się na środki materialne (tu: awans, podwyżka, premia, świadczenia) i niematerialne (tu: awans, pochwała, uznanie, prestiż) [5].
Środki perswazji odwołują się do motywacji wewnętrznej. Polegają na negocjacjach pracodawcy czy przełożonego z pracownikiem. Perswazja może stanowić oddziaływanie jednostronne (kiedy opiera się na ingerencji w sferę emocjonalną pracownika) lub – co zalecane – dwustronne (kiedy opiera się na wzajemnym przekonywaniu się dwóch stron do swoich racji i motywacji). Środki perswazyjne przyjmują zazwyczaj formę apelu, doradztwa, sugestii, konsultacji, akceptacji, negocjacji. Środki te mogą jednak łatwo przejść w manipulację, wtedy ich formami są wmawianie, wymuszanie czy propaganda. Rola środków perswazyjnych wzrasta wraz z podnoszeniem kwalifikacji i świadomości osoby pracującej – jest to zjawisko pozytywne. Jednak w warunkach autokratycznego stylu zarządzania firmą występuje spore niebezpieczeństwo przerodzenia się środków perswazyjnych w środki przymusu, a precyzyjniej w nakazy, co należy uznać za zjawisko negatywne [5].

Delegowanie zadań – co to takiego?

Delegowanie zadań to przydzielanie zadań pracownikom przez menadżera, lidera lub innego przełożonego (w wypadku dużych firm) lub właściciela przedsiębiorstwa (w wypadku małych i średnich firm) [7]. Jest to jedno z najważniejszych narzędzi, jakie posiada menadżer, lider zespołu. To od delegowania zadań zależy wzrost lub spadek efektywności pracy całego zespołu [7]. Delegowanie nie polega na powierzaniu spraw, które mają małe znaczenie. Powinno ono bowiem obejmować czynności cykliczne bądź takie, które dotyczą ważnych, dużych projektów. Delegujący powinien aktywnie uczestniczyć w procesie realizacji tych projektów [1].
Precyzyjnie wskazanie zadań i określenie czasu na ich realizację to tylko wstęp w procesie delegowania. Istotne jest nieustanne komunikowanie się z członkami zespołu, monitorowanie występujących problemów i osiągniętych rezultatów, dawanie rad i udzielanie szeroko pojętej pomocy [1]. Poprzez tego typu działania osoby przydzielone do konkretnych zadań będą czuły się członkami zespołu, bez których całość projektu nie mogłaby zostać zrealizowana [1]. Zadaniem osoby delegującej zadania jest także bieżące komentowanie i podsumowywanie efektów tego, co udało się zrealizować zespołowi [7]. Szczegółowe wskazówki dotyczące delegowania zadań zostały przedstawione poniżej.

Jak skutecznie delegować zadania?

Aby skutecznie delegować zadania należy przestrzegać kilku podstawowych wytycznych. Po pierwsze trzeba wyznaczać pracowników do zadań ze stosownym wyprzedzeniem czasowym. W przeciwnym wypadku terminowe wykonanie projektu może okazać się niemożliwe do zrealizowania bądź też wykonanie go będzie się wiązało z presją czasu wywieraną na pracownikach. Dobrym rozwiązaniem jest tak zwane buforowanie zadań, które polega na informowaniu pracowników, że dane zadanie powinno zostać wykonane na kilka godzin, dni, tygodni wcześniej (w zależności od skali zadania) niż w rzeczywistości [7].
Po drugie istotny jest wybór pracowników posiadających stosowne umiejętności, kompetencje i potencjał. Przy podejmowaniu decyzji o podziale obowiązków, menadżer powinien wytypować osoby, które mogą je wykonać (np. dlatego, że w przyszłości realizowały podobne zadania i posiadają doświadczenie). Można również wyznaczyć pracowników, którzy szybko się uczą i łatwo nabywają nowe umiejętności [7].
Po trzecie należy precyzyjnie określać zadnia i sposób ich wykonywania. Obowiązkiem menadżera jest zapoznanie pracowników z etapami realizacji projektu, potencjalnymi przeszkodami, oczekiwanymi rezultatami. Dodatkowo menadżer powinien sprawdzić czy i jak podwładny zrozumiał zadanie. W komunikacji ludzkiej pełne zrozumienie intencji nadawcy jest niemalże niemożliwe, dlatego warto poprosić podwładnego o przedstawienie etapów realizacji zadania własnymi słowami [7].
Po czwarte – trzeba wyjaśnić nadrzędny sens zadania. Przedstawienie jasnego celu, intencji projektu jest bardzo ważną kwestią, o której niejednokrotnie menadżerowie zapominają. Jest to istotne szczególnie wtedy, gdy powierzane zostają żmudne, niemotywujące zadania, których wykonanie jest jednak konieczne. Po objaśnieniu „wyższego celu" zadania pracownik jest zmotywowany do wydajniejszej pracy [1].
Po piąte – należy prowadzić rozmowy z pracownikami oraz motywować ich. W rozmowach powinny pojawiać się pytania, które są związane z dotyczącymi projektu pomysłami i wizjami pracowników. Tego rodzaju dialog i zainteresowanie przełożonego będzie miało wpływ na za-

angażowanie pracownika, a także na zbudowanie dobrej relacji między obojgiem [1].
Wreszcie po szóste – dobry lider czy menadżer nie zapomina o uczczeniu sukcesu pracownika bądź całego zespołu. Po wykonaniu zadania i sprawdzeniu jego jakości, dobrze widziane jest podziękowanie wszystkim zaangażowanym w zadania projektowe oraz zmotywowanie do dalszych działań. To pole jest najczęściej pomijane przez liderów i menadżerów, którzy zapominają o docenieniu wkładu każdego poszczególnego pracownika [7].

Jakie są zalety delegowania zadań?

Abstrahując od terminowego ukończenia projektu, delegowanie zadań wiąże się z innymi korzyściami dla przedsiębiorstwa. Po pierwsze menedżer będzie miał sposobność podzielenia się z innymi pracownikami firmy swoją wiedzą teoretyczną oraz praktycznymi umiejętnościami. Poprzez to podwładni mają szansę na podniesienie własnych kompetencji. Po drugie – nowe wyzwania pozwolą menedżerowi na obserwowanie i sprawdzenie zdolności podwładnych bądź potwierdzenie wcześniejszych spostrzeżeń i opinii. Delegowanie zadań może także przyczynić się do powstawania nowych koncepcji oraz rozwiązań. Poprawi to stosunki menedżera z podwładnymi, jak również wpłynie na pozytywne odczucia pracowników, na przykład wzrośnie ich zadowolenie z wykonywanej pracy [7].

Zakończenie

System motywowania wymaga skrupulatności, uwagi oraz znajomości specyfiki i potrzeb danego przedsiębiorstwa. Dzieje się tak przez wzgląd na to, że jest on procesem złożonym. Składają się na niego: indywidualna motywacja pracownika, wzajemna motywacja pracowników oraz motywacja firmy [8]. Tylko połączenie tych składników może przynieść pożądane rezultaty. Dobre i – co nie mniej istotne – efektywne zarządzanie firmą nie jest możliwe w dzisiejszych czasach bez znajomości i rozumienia takich pojęć jak human skills, motywowanie pracowników oraz delegowanie zadań [8]. Odpowiednie motywowanie pracowników oraz delegowanie zadań jest prymarnym zadaniem każdego menadżera, lidera, przełożonego [8]. Bez zwrócenia uwagi na czynnik ludzki żadne przedsiębiorstwo, nawet najlepiej rokujące, nie osiągnie oczekiwanych i długoterminowych rezultatów w swojej branży.

Literatura

[1] Dittmer Robert, McFarland Stephanie, „Podejmowanie decyzji i delegowanie zadań. 151 błyskotliwych rozwiązań”, Wydawnictwo Helion, Gliwice 2014.
[2] Kopertyńska Maria Wanda, „Motywowanie pracowników. Teoria i praktyka”, Agencja Wydawnicza Placet, Kraków 2015.
[3] Kozłowski Waldemar, „Zarządzanie motywacją pracowników”, Wydawnictwa Fachowe CeDeWu, Lublin 2009.
[4] Oleksyn Tadeusz, „Zarządzanie zasobami ludzkimi w organizacji”, Wydawnictwo Wolters Kluwer, Warszawa 2008.
[5] Pietroń-Pyszczek Agata, „Motywowanie pracowników. Wskazówki dla menadżerów”, Wydawnictwo Marina, Łódź 2015.
[6] Selden Bob, „Jak być dobrym szefem”, Dom Wydawniczy Rebis, Łódź 2009.
[7] Tracy Brian, „Delegowanie i kontrolowanie”, Wydawnictwo MT Biznes, Warszawa 2017.
[8] Tracy Brian, „Motywowanie. 21 sposobów zwiększenia wydajności sprawdzonych w praktyce”, Wydawnictwo MT Biznes, Warszawa 2016.
[9] Kardasz Piotr, Ewa Jaworska, „Dotacje bezzwrotne” ISBN 978-83-946273-0-0, Wrocław 2016r

Dekompozycja ciągu uczącego dla rozproszonej bazy danych

TEACHING SEQUENCE DECOMPOSITION FOR DISTRIBUTED DATABASE

Swietłana Lebiediewa[1]

Abstract: The problem of decomposition of a teaching sequence (TS) for distributed database is formulated. Three types of TS decomposition for a distributed database are formulated. Theorems concerning memory occupancy by the TS after decomposition are proved. The results of a calculation experiment are presented, that illustrate the memory occupancy depending upon the decomposition type, the redundancy of features in the tree and upon the type of the dispersion.

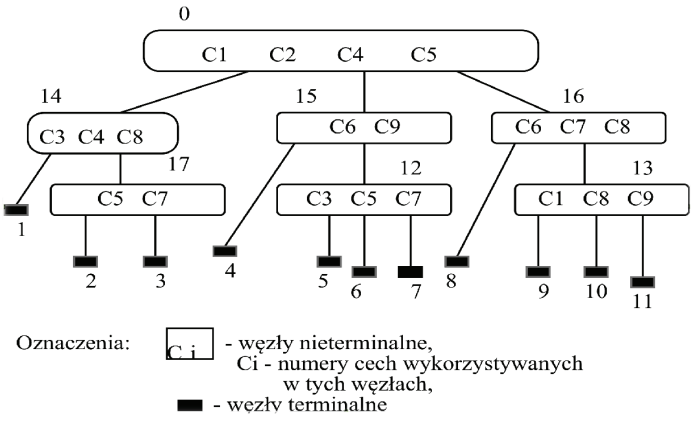
Streszczenie: Sformułowano problem dekompozycji ciągu uczącego (CU) dla rozproszonej bazy danych. Zdefiniowano trzy rodzaje dekompozycji. Przedstawiono twierdzenia dotyczące zajętości pamięci przez zdekomponowany ciąg uczący. Zaprezentowano wyniki badań symulacyjnych ilustrujących zajętość pamięci w zależności od typu rozproszenia i rodzaju dekompozycji.

Słowa kluczowe: rozproszona baza danych, rozpoznawanie wieloetapowe, ciąg uczący, dekompozycja

1. Sformułowanie problemu

Rozpoznawanie wielostopniowe polega na *dekompozycji* problemu decyzyjnego, czyli zastępowaniu jednorazowego rozpoznawania sekwencją tzw. rozpoznawań lokalnych przeprowadzanych w poszczególnych węzłach zgodne z zadaną konstrukcją drzewa decyzyjnego (DD), rys. 1. Żeby zaklasyfikować rozpoznawany obiekt do jednej z klas, należy przejść ścieżką w DD startując od korzenia. W każdym napotykanym węźle wewnętrznym (**decyzyjnym**) należy podjąć decyzję o dalszej drodze w grafie, aż osiągnięty zostanie węzeł terminalny. Klasa z nim związana reprezentuje końcowy wynik rozpoznawania wielostopniowego. Tak więc w trakcie klasyfikacji wielostopniowej rozpoznawany obiekt podlega sekwencji decyzji na ścieżce od korzenia do węzła terminalnego.

Algorytmy rozpoznawania z uczeniem korzystają z ciągu uczącego (CU) [1,2,3,4]. CU jest ciągiem poprawnie zaklasyfikowanych obiektów, elementami CU są pary postaci (x_k, k), gdzie x jest wektorem wartości cech obiektu, k – numerem klasy. Drzewo decyzyjne (DD) przedstawiono na rys.1, a odpowiadający mu CU – na rys. 2.



Rys. 1. Drzewo decyzyjne NR1a

W CU przedstawionym na rys. 2 w miejscu wartości niektórych cech występuje symbol *, symbol ten oznacza, że dana cecha jest cechą nieistotną dla rozpoznawanego obiektu i nie jest brana pod uwagę przez algorytm rozpoznawania. Tak więc w procesie rozpoznawania wielostopniowego na różnych etapach rozpoznawania wykorzystuje się tylko niektóre elementy wektora cech obiektów, a pewne cechy wcale nie występują na poszczególnych ścieżkach.

1.Wrocławska Wyższa Szkoła Informatyki Stosowanej „Horyzont” (‘Horyzont’ Wroclaw School of Information Technology) swietlana@lebiediewa.com

C1	C2	C3	C4	C5	C6	C7	C8	C9	NR KLASY
16	3,44	3	6,55	40,0	48	116	*	14,9	7
15	3,45	2	8,45	40,0	*	*	120	*	1
14	2,4	1	8,33	38,1	*	110	*	*	2
18	3,0	*	7,5	35,0	40	*	*	15,0	4
15	2,7	*	6,0	20,0	35	110	112	*	8

Rys.2. Fragment ciągu uczącego dla DD NR1a
Figure 2. A fragment of the teaching sequence for DT No. 1a

W celu oszczędności pamięci proponuje się dekompozycję CU. Z algorytmów rozpoznawania przedstawionych w [2,3] widać, że czas pracy AR w każdym z węzłów składa się z czasu potrzebnego na utworzenie segmentu danych oraz czasu pracy algorytmów rozpoznawania. Odpowiednia postać CU może skrócić czas potrzebny na utworzenie segmentu danych. Celem dekompozycji jest rozbicie CU na takie podciągi, które zarówno minimalizują zajętość pamięci przez CU jak i zmniejszają czas potrzebny rozpoznawania obiektów [5,6]. W pewnych przypadkach wydaje się uzasadnione rozbicie procesu rozpoznawania na kilka lokalizacji, w których znajdują się lokalne bazy danych (LBD), które połączone siecią komunikacyjną stanowią rozproszoną bazę danych (RBD). Model konceptualny scentralizowanej bazy danych (SBD) przedstawiono w [5, 7]. Model konceptualny BD zawiera informacje o rozpoznawanym obiekcie a także informację o CU SEQUENCE i relacje opisujące strukturę DD KLASY, WĘZŁY i CECHY. W relacji SEQUENCE znajduje się informacja o CU. Relacja KLASY opisuje zależność pomiędzy numerem klasy a numerem węzła będącego bezpośrednim poprzednikiem tej klasy. dla każdej klasy wskazany jest jej poprzednik. Relacja WĘZŁY zawiera informacje o numerach każdego węzła, o bezpośrednich następnikach i poprzednikach węzła. W relacji CECHY zawarta jest informacja, w których węzłach wykorzystywane są poszczególne cechy [5, 6]. Model konceptualny RBD różni się od modelu konceptualnego SBD tym tylko, że w opisie relacji WĘZŁY występuje dodatkowy atrybut LOKALIZACJA, wskazujący na lokalizację węzła nieterminalnego [7]. Dekompozycję CU dla SBD przedstawiono w [6]. Przedstawimy dekompozycję CU dla RBD.

2. Dekompozycja ciągu uczącego dla rozproszonej bazy danych

Dla rozproszonej bazy danych (RBD) możemy wyróżnić trzy rodzaje dekompozycji ciągu uczącego: dekompozycję *naturalną*, zmodyfikowaną dekompozycję 1. rodzaju i dekompozycję 2. rodzaju. Ze względu na wysoki koszt transmisji danych wydaje się zasadnym przyjąć założenie, iż CU jest dekomponowany w taki sposób, ażeby algorytm rozpoznawania nie musiał korzystać z informacji zawartej w CU znajdującym się w innej lokalizacji. Można przyjąć, że w komputerze głównym pozostawia się cały CU zdekomponowany lub nie, ze względu na to, że w węźle będącym korzeniem wykorzystywane są cechy występujące we wszystkich elementach CU, można też założyć całkowite rozproszenie CU. W dalszym ciągu zakładamy całkowite rozproszenie CU (jeżeli założyć zachowanie CU w głównej lokalizacji, zajętość pamięci będzie nie mniejsza niż przy całkowitym rozproszeniu). Aby umożliwić ponowne otrzymanie niezdekomponowanego CU, do każdego wiersza CU zdekomponowanego dodajemy identyfikator.
Dekompozycja naturalna polega na tym, że z CU usuwa się wiersze, które w kolumnie NR KLASY nie zawierają numerów klas nieosiągalnych z węzłów nieterminalnych lokalnej bazy danych (LBD), następnie usuwa się kolumny zawierające wartości cech, nie wykorzystywanych w danej lokalizacji. Metoda dekompozycji 1. rodzaju dla RBD jest tak zmodyfikowana, że każdy podciąg (lub jego fragment) odpowiadający węzłowi pośredniemu znajdującemu się poza główną lokalizacją, zawiera tylko wartości cech wykorzystywanych w tej lokalizacji. Podciąg ciągu uczącego budujemy dla każdego węzła nieterminalnego, który jest bądź bezpośrednim poprzednikiem węzła nieterminalnego(klasy), bądź jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji. W [6] przedstawiono metodę dekompozycji 1. rodzaju dla SBD i metodę dekompozycji 2. rodzaju dla SBD.
Algorytmy dekompozycji korzystają z informacji o CU i strukturze DD zawartej w relacjach SEQUENCE, KLASY, WĘZŁY, CECHY modelu konceptualnego BD.

ALGORYTM 1. (algorytm naturalnej dekompozycji CU dla RBD)
Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.
Wyznaczyć: CU zawierający wartości cech, wykorzystywanych w danej lokalizacji do rozpoznawania klas osiągalnych z węzłów nieterminalnych należących do tej lokalizacji.
KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.
KROK 1. Dla każdej LBD z relacji WĘZŁY pobierz numery węzłów nieterminalnych znajdujących się w LBD.
KROK 2. Z relacji NASTĘPNIKI wybierz numery klas osiągalnych z węzłów nieterminalnych należących do LBD.
KROK 3. Dla każdej LBD z relacji przechowującej CU wybierz wiersze, dla których
NR KLASY = "klasa osiągalna z węzła nieterminalnego należącego do tej lokalizacji";
KROK 4. Z otrzymanej relacji usuń atrybuty zawierające wartości cech, które nie są wykorzystywane w tej lokalizacji. **STOP**.

Oznaczmy przez *ZPDN* zajętość pamięci przez CU otrzymany w wyniku dekompozycji naturalnej, przez i – numer LBD, przez LC_i – liczbę cech wykorzystywanych w lokalizacji i, przez k_i – liczbę klas osiągalnych z węzłów nieterminalnych należących do lokalizacji i, przez L – liczbę LBD. Zajętość pamięci przez CU dla całej RBD wyraża wzór:

$$ZPDN = \sum_{i=1}^{L} (LC_i + 2) * K_i \quad (1)$$

ALGORYTM 2. (zmodyfikowany algorytm dekompozycji pierwszego rodzaju CU dla RBD)
Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.
Wyznaczyć: dla każdego węzła nieterminalnego będącego bądź poprzednikiem klas, bądź poprzednikiem węzłów nieterminalnych znajdujących się w innej lokalizacji wyznaczyć taki podciąg ciągu uczącego, który zawiera wartości cech wykorzystywanych na ścieżce od „korzenia" poddrzewa do tego węzła.

KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.
KROK 1. Z relacji KLASY i WĘZŁY pobierz numery węzłów należących do LBD, które są bądź bezpośrednimi poprzednikami węzłów terminalnych bądź bezpośrednimi poprzednikami węzłów nieterminalnych należących do innej lokalizacji.
KROK 2. Dla każdego takiego węzła utwórz relację zawierającą podciąg uczący dla wszystkich klas bezpośrednio osiągalnych z tego węzła i zawierający wartości cech wykorzystywanych w tej lokalizacji (na ścieżce od "korzenia" poddrzewa do tego węzła).
KROK 3. Jeżeli węzeł jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji, utwórz podciąg CU zawierający wartości cech wykorzystywanych w tej lokalizacji do rozpoznawania klas będących następnikami tego węzła, otrzymany podciąg dodaj do podciągu otrzymanego w kroku drugim.
STOP.

Niech L, jak wyżej, oznacza liczbę LBD, a i – numer lokalizacji. Oznaczmy przez J_i liczbę węzłów nieterminalnych w lokalizacji i, przez LC_j– liczbę cech wykorzystywanych w danej lokalizacji przez algorytmy rozpoznawania na ścieżce do węzła j, przez K_j – łączną liczbę klas bądź będących bezpośrednimi następnikami węzła j, bądź osiągalnych z węzła – bezpośredniego następnika j należącego w innej lokalizacji. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji pierwszego rodzaju (*RZPD1*) dla RBD wyraża wzór:

$$RZPD1 = \sum_{i=1}^{L} \sum_{j=1}^{J_I} \left(LC_j + 2\right) * k_j \quad (2)$$

Algorytm dekompozycji drugiego rodzaju dla RBD nie różni się od algorytmu dekompozycji drugiego rodzaju dla SBD:
ALGORYTM 3. (algorytm dekompozycji 2. rodzaju)
Dane: model konceptualny RBD – relacje SEQUENCE, WĘZŁY, CECHY.
Wyznaczyć: dla każdego węzła nieterminalnego utworzyć taki podciąg ciągu uczącego, który zawiera wartości tylko tych cech, które są wykorzystywane w tym węźle.
KROK 1. Do relacji SEQUENCE przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.
KROK 2. Z relacji WĘZŁY pobierz numery wszystkich węzłów nieterminalnych.
KROK 3. Dla każdego węzła nieterminalnego utwórz podciąg uczący (z CU wybierz wiersze, dla których NR KLASY = "klasa osiągalna z danego węzła", z otrzymanej relacji usuń atrybuty zawierające wartości cech nie wykorzystywanych w tym węźle). **STOP**.

Niech J oznacza liczbę wszystkich węzłów nieterminalnych w bazie danych, j – kolejny numer węzła, LC_j – liczbę cech wykorzystywanych w węźle j, K_j – liczbę klas osiągalnych z węzła j. Wzór na zajętość pamięci przy dekompozycji drugiego rodzaju dla RBD (*RZPD2*) ma postać:

$$RZPD2 = \sum_{j=1}^{J} \left(LC_j + 1\right) * K_j \quad (3)$$

Wniosek Dla zmodyfikowanej dekompozycji 1. rodzaju dla RBD redundancja cech występujących na ścieżce nie ma wpływu na zajętość pamięci, jeżeli występuje w węzłach znajdujących się w jednej lokalizacji, natomiast zwiększa zajętość pamięci, jeżeli występuje w węzłach leżących w różnych lokalizacjach.
Z Algorytmu 2 wynika
Twierdzenie 1. Zajętość pamięci przy dekompozycji drugiego rodzaju nie zależy od sposobu rozproszenia.
Mają miejsce następujące lematy:
Lemat 1. Przy całkowitym rozproszeniu (każdy z węzłów nieterminalnych znajduje się w innej LBD) ma miejsce następujący wzór:

$$ZPDN = RZPD1 = RZPD2$$

Lemat 2. Jeżeli w LBD żadna para węzłów nieterminalnych nie znajduje się w relacji poprzednik–następnik, ma miejsce równość:

$$RZPD1 = RZPD2,$$

w przeciwnym przypadku

$$RZPD1 < RZPD2$$

Z lematów 1. i 2. wynika

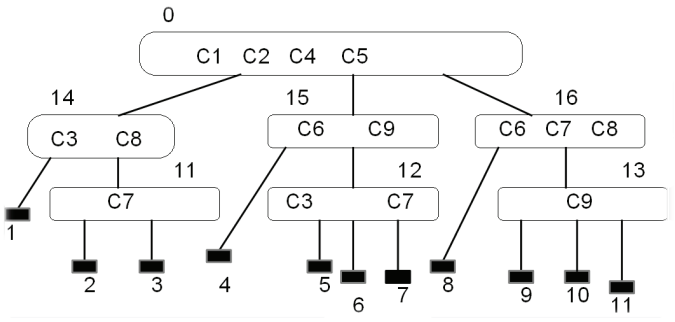
Twierdzenie *2*. Dla RBD ma miejsce zależność:

$$\text{RZPD1} \leq \text{RZPD2}$$

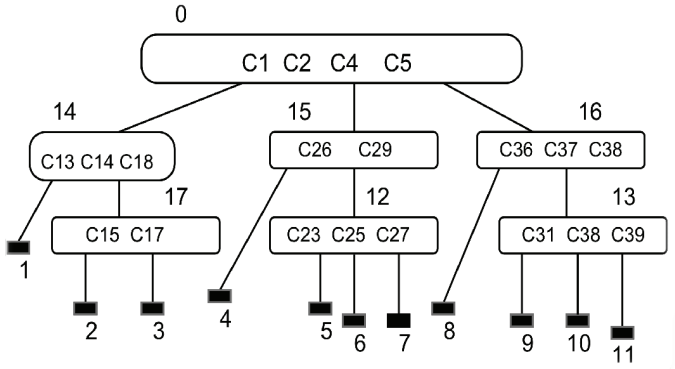
3. Eksperyment obliczeniowy

Przedstawimy wyniki eksperymentu obliczeniowego ilustrującego zajętość pamięci CU otrzymanych w wyniku

zmodyfikowanej dekompozycji pierwszego i drugiego oraz dekompozycji naturalnej dla rozproszonej bazy danych. Rozpatrujemy zajętość pamięci dla drzew 1a (Rys. 1), 1b (Rys 3.), i
1c (Rys 4.) w zależności od rodzaju dekompozycji i typu rozproszenia.



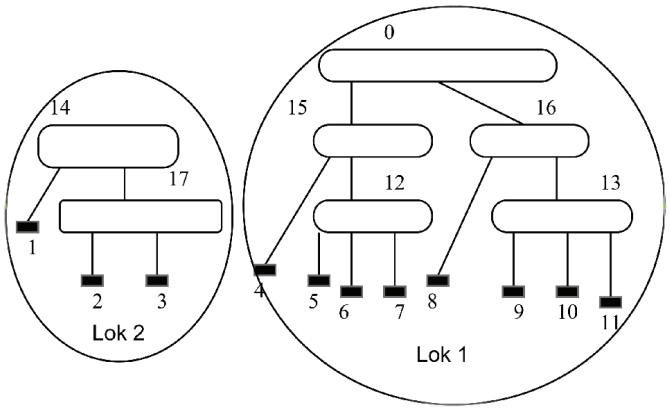
Rys. 3 Drzewo decyzyjne 1b
Figure 3. Decision tree No. 1b



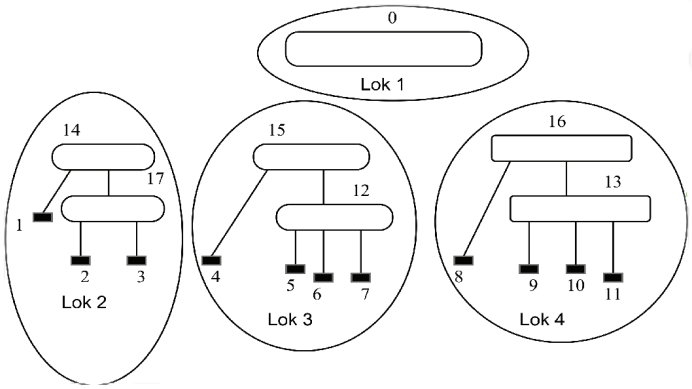
Rys. 4. Drzewo decyzyjne 1c
Figure 4. Decision tree No. 1c

Struktura drzew 1a, 1b i 1c jest identyczna, drzewo 1b różni się od drzewa 1a tym, że na poszczególnych ścieżkach nie ma redundancji cech, a w drzewie 1c w każdym z węzłów występują inne cechy.

Rozpatrzmy dwa rodzaje rozproszenia dla drzew 1a-1c: rozproszenie typu A i rozproszenie typu B, rys.5. W przypadku rozproszenia typu A rozproszona baza danych składa się z dwóch LBD. Do pierwszej LBD należą węzły 0, 16, 13, 15 i 12, do drugiej – węzły 14 i 17. W przypadku rozproszenia typu B rozproszona baza danych składa się z czterech LBD, do pierwszej LBD należy węzeł numer 0 (korzeń drzewa), do drugiej – węzły 14 i 17; do trzeciej – węzły 15 i 12; do czwartej – węzły 16 i 13. Rysunki 6 i 7 ilustrują zajętość pamięci przez CU dla drzew 1a-1c odpowiednio dla rozproszenia typu A i rozproszenia typu B. Jak widać różnice w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego rodzaju i dekompozycji drugiego rodzaju zmniejszają się w miarę wzrostu rozproszenia aż do zaniku różnic (por. Lemat 1).

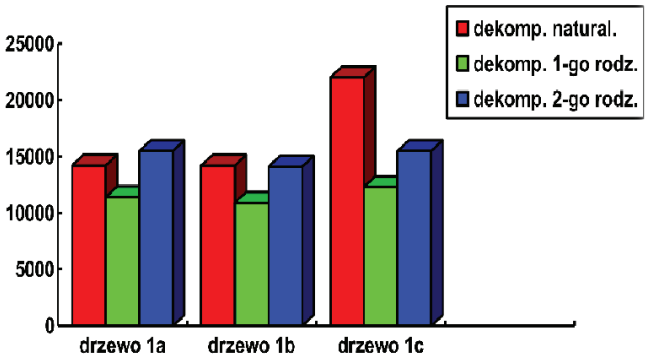


Rozproszenie typu A

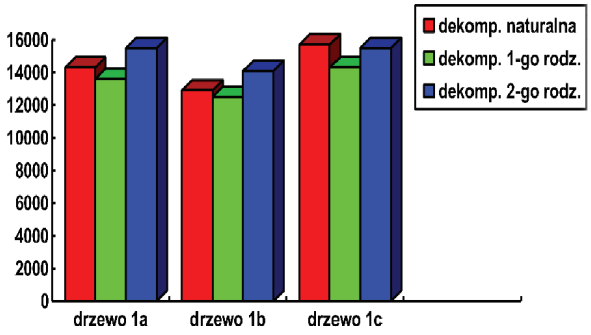


Rozproszenie typu B

Rys. 5. Rozproszenie typu A i B dla drzew 1a – 1c.
Figure 5. The dispersion of type A and B for the trees 1a - 1c.



Rys. 6. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji naturalnej dla drzew 1a-1c, rozproszenie typu A
Figure 6. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type A



Rys. 7. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji natu-

ralnej dla drzew 1a-1c, rozproszenie typu B
Figure 7. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type B

Najlepsze rezultaty daje dekompozycja 1-go rodzaju dla rozproszenia typu A. Dla CU otrzymanych w wyniku dekompozycji 1-go rodzaju wzrost zajętości pamięci następuje wraz ze wzrostem rozproszenia i wzrostem redundancji cech. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju wzrasta wraz ze wzrostem redundancji. Rodzaj rozproszenia nie ma wpływu na zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju. Dekompozycja naturalna daje najlepsze rezultaty w przypadku braku redundancji i rozproszenia typu C. Najgorsze rezultaty daje dekompozycja naturalna w przypadku rozproszenia typu B (większa zajętość pamięci niż w przypadku CU nie zdekomponowanego).
Dla każdej dekompozycji korzystny wpływ na zajętość CU ma rozpoznanie części klas na wcześniejszych etapach.

Wnioski:

Dla RBD zmodyfikowana dekompozycja 1. rodzaju daje wyraźnie lepsze wyniki, gdy liczba etapów jest duża i w poszczególnych lokalizacjach znajdują się węzły należące do długich ścieżek. Im większe rozproszenie, tym mniejsza różnica między rodzajami rozproszenia. Przy całkowitym rozproszeniu każda z dekompozycji daje ten sam rezultat. Dekompozycja 2. rodzaju nie zawsze daje oszczędność pamięci. Dekompozycja 2. rodzaju na ogół potrzebuje więcej pamięci niż dekompozycja 1. rodzaju, a w niektórych przypadkach – także więcej niż dekompozycja naturalna. Wraz z wzrostem rozproszenia (wzrostem liczby lokalizacji) maleje różnica w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego i drugiego rodzaju. Przy dużym rozproszeniu najbardziej korzystna wydaje się być dekompozycja 2-go rodzaju, ponieważ CU otrzymane w wyniku dekompozycji 2. rodzaju są identyczne z fragmentami CU wykorzystywanych przez algorytmy rozpoznawania w węźle, dlatego w przypadku stosowania dekompozycji 2. rodzaju nie ma potrzeby tworzenia specjalnej relacji *Fragment CU w węźle* przy tworzeniu modelu zewnętrznego, a różnica w zajętości pamięci przez CU otrzymane w wyniku dekompozycji 1-go i 2-go rodzajów jest nieznaczna lub żadna w przypadku całkowitego rozproszenia.

References

[1] Bubnicki, Z. ‘Knowledge-Based Approach as a Generalization of Pattern Recognition Problems and Methods’. Systems Science Vol. 19, No 2 (1993), pp. 5–21
[2] Flasiński, M. Wstęp do sztucznej inteligencji. Warsaw: PWN, 2011
[3] Kurzyński, M. Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne. Wrocław: Wyd. PWr., 1987
[4] Józefczyk, J. ‘Rozpoznawanie i zastosowania biomedyczne’ [in:] Problemy automatyki i informatyki, Wrocław: Wyd. Ossolineum, 1998, pp. 45–58
[5] Lebiediewa, S, ‘System informatyczny dla wieloetapowego rozpoznawania obiektów’. Biuletyn Naukowy WWIS. Informatyka, 2014
[6] Lebiediewa, S, ‘Training sequence decomposition’. Biuletyn Naukowy WWIS. Informatyka, 2015
[7] Lebiediewa, S. Metodologia projektowania problemowo zorientowanych baz danych do systemów wielostopniowego podejmowania decyzji. Wrocław: Wyd. Pwr., 1998

Współczesna automatyzacja i robotyzacja a człowiek

Modern automation and robotics vs man

Autorzy: Piotr Kardasz[1],

Streszczenie : Artykuł przedstawia wiele informacji na temat działania, postępującej automatyzacji i robotyzacji w świecie, a także szans i zagrożeń z tym związanych. Omówiono również kwestie opodatkowana maszyn i robotów..

Abstract : The article presents a lot of information about the operation, progressive automation and robotics in the world, as well as the opportunities and threats associated with it. The issues of taxed machines and robots were also discussed.

Słowa kluczowe: automatyzacja, robotyzacja, podatki

Keywords: automation, robotics, taxes

Czym są automatyzacja i robotyzacja

Współczesny świat charakteryzuje się przede wszystkim jako technologicznie prężnie rozwijający się. Już od końca XIX wieku, co około dwadzieścia lat, pojawiają się wynalazki, które zmieniają rzeczywistość nie do poznania. Pod koniec XIX wieku zaistniała konsumencka elektryczność, początek XX wieku to czas wynalezienia samolotu oraz radia, przełomowe lata 30. to powstanie i rozwój telewizji, lata 50. – tranzystor, lata 70. – mikrokomputery, lata 90. – internet.

Postępująca automatyzacja i robotyzacja nieustannie i w coraz to nowych obszarach zastępują pracę człowieka. Zanim jednak opisane i ocenione zostaną konsekwencje owego rozwoju, należy wyjaśnić, czym właściwie są automatyzacja i robotyzacja. Automatyzacja to ograniczanie bądź zastępowanie ludzkiej pracy, zarówno w sferze fizycznej jak i umysłowej, poprzez wykorzystanie pracy maszyn. Maszyny te mają funkcję samoregulacji, mogą więc wykonywać dane czynności bez pomocy człowieka[2] . Robotyzacja natomiast to podobny proces, w którym pracę ludzką zastępuje się pracą robotów[3] . Obie opisane dziedziny nauki są coraz częściej stosowane w branżach usługowych. Automatyzowanie mechanizmów usługowych przez zastosowanie aplikacji informatycznych, czyli robotów nazywa się „zrobotyzowaną automatyzacją procesów" (*ang. Robotic Process Automation – RPA*)[3] .

Lista dziesięciu elektronicznych technologii przyszłości

Korzystanie z powstałych innowacyjnych rozwiązań, udogodnień i rozrywek to za mało dla miłośników technologii. W związku z tym już dziś podaje się długą listę urządzeń, bez których najprawdopodobniej w przyszłości nie będziemy mogli wyobrazić sobie funkcjonowania. I tak na przykład w Davos w 2014 roku podczas Światowego Forum Ekonomicznego wyróżniono dziesięć elektronicznych technologii, które mogą kształtować społeczeństwo przyszłości[5] .

Pierwszą z nich jest urządzenie łączące mózg z komputerem. Gdyby taki sprzęt faktycznie powstał, pojawiłaby się także możliwość pisania na komputerze bez używania tradycyjnej klawiatury i, co istotniejsze, ułatwiłoby to lub całkowicie umożliwiło pracę z komputerem osobom niepełnosprawnym. Drugi wynalazek przyszłości z listy stanowią auta tworzone z włókna węglowego, czyli nanostruktur zamiast z metolowych części. Pojazdy wyprodukowane z włókna węglowego miałby wiele zalet: byłyby do czterdziestu procent lżejsze od samochodów metalowych, cechowałby się większą trwałością i łatwiejszą możliwością recyklingu, a także znacząco mniejszym zapotrzebowaniem na paliwo. Numerem trzy są magazyny energii, które dałyby możliwość przechowywania energii elektrycznej pochodzącej z odnawialnych źródeł, takich jak słońce czy wiatr. Aktualnie pozyskiwana energia z wy-

1. Piotr Kardasz – Wrocławska Wyższa Szkoła Informatyki Stosowanej we Wrocławiu, Wydział Automatyki i Robotyki
2. A. K. Gupta, Industrial Automation and Robotics. Laxmi Publications (P) Ltd., 2007, s. 1.
3. A. Grycuk, Klastry a rozwój regionalny. Klaster usług biznesowych w Krakowie, Krakós 2017, s. 145–146.
4. Tamże.
5. http://tvn24bis.pl/wiadomosci-gospodarcze,71/dziesiec-technologii-przyszlosci-ktore-zmienia-nasze-zycie,402504.html [dostęp: 08.09.2017 r.].

żej wymienionych źródeł jest od razu użytkowana w sieci, a jej nadwyżki pozostają niewykorzystane. Na czwartym miejscu plasuje się tak zwana „zielona energia", czyli wspomniane pozyskiwanie energii słonecznej bądź wietrznej, co pozwoliłoby nie tylko na jej magazynowanie, lecz także na produkowanie. W rezultacie energia kosztowałaby mniej i stałaby się bardziej przyjazna dla atmosfery. Piąta technologia jutra to implanty, których zadaniem byłoby informowanie ludzi o ich aktualnym stanie zdrowia. Nosiłoby się ją na skórze bądź pod skórą, co pozwoliłoby na monitorowanie tętna, poziomu stresu i innych[6] .

Kolejnym, szóstym pomysłem, są krzemowe akumulatory, które podobnie jak w wypadku samochodów z włókna węglowego wykorzystałyby nanostruktury. Akumulatory składają się z niewielkich cząsteczek krzemu, co w efekcie pozwala im na dłuższe działanie do nawet trzech razy i znacznie sprawniejsze niż dotychczas ładowanie. Siódmym wynalazkiem są bezekranowe wyświetlacze, które w zamierzeniu mają projektować rzeczywistość 3D w przestrzeni i w rezultacie pozwolą zaoszczędzić energię. Ósme miejsce zajmuje leczenie mikroorganizmami, czyli wdrożenie terapii opartych o badania mikrobiomów występujących w ludzkim ciele, na przykład w jamie ustnej oraz w przewodzie pokarmowym. Mikrobiomy te mają w założeniu działać leczniczo. Dziewiąta technologia przyszłości to inteligentne leki, których skład opierać ma się na kwasach rybonukleinowych (RNA) występujących w jądrach komórkowych. Technologia ta przynosi pespektywę walki z wieloma chorobami, w tym z nowotworem. Ostatnie, dziesiąte miejsce, zajmuje chemia mająca moc oczyszczania ścieków oraz wody morskiej. Oczyszczanie na dużą skalę może stać się opłacalne poprzez chemiczne wykorzystywanie zanieczyszczeń. Przyczyniłoby się to także do rozwiązania problemu niewystarczających zasobów słodkiej wody na świecie[7] .

Podane przykłady urządzeń i rozwiązań zachwycają innowacyjnością i funkcjonalnością. Jedyną ich wadą może wydać się fakt, że pogłębią one masowe zastępowanie ludzi przez maszyny, nie pozostawiając dla siebie konkurencji. I tak na przykład technologia łącząca mózg z komputerem zastąpi copywriterów, implanty informujące o stanie zdrowia pacjenta lub zaawansowane i samodzielne leczenie mikroorganizmami przyniosą zmniejszenie częstotliwości wizyt pacjentów u lekarzy, a chemia samooczyszczająca ścieki i wodę morską zminimalizuje liczbę zatrudnień operatorów urządzeń oczyszczania ścieków. Czy w związku z tym możemy mówić o sztucznej inteligencji, która zaczyna mieć monopol na wszelkie zadania zarezerwowane dotychczas wyłącznie dla człowieka? Traktują o tym raport Białego Domu, którego głównym autorem jest ówczesny prezydent Stanów Zjednoczonych – Barack Obama (z grudnia 2016 roku) oraz apel współzałożyciela i byłego prezesa zarządu korporacji Microsoft – Billa Gatesa (z marca 2017 roku).

Raport Baracka Obamy w związku z rosnącą automatyzacją i robotyzacją

„Rozwój sztucznej inteligencji i postęp robotyzacji prowadzi do likwidowania miejsc pracy, zwłaszcza tych, które nie wymagają dużej wiedzy. Rozwój ten jest jednak niezbędny, choć może pogłębić przepaść między różnymi grupami społecznymi" – głosi raport Białego Domu zatytułowany „Sztuczna inteligencja, automatyzacja i ekonomia" szczegółowo omówiony przez „The Washington Post" – największą gazetę codzienną miasta Washington. Kolejno dokument traktuje o konieczności zwiększenia dostępu do edukacji technicznej, a także o potrzebie przynoszenia pomocy finansowej osobom bezrobotnym[8] .

Raport ten wysnuwa przypuszczenia, że coraz szersze wykorzystywanie sztucznej inteligencji, komputeryzacji oraz robotyzacji przyniesie najprędsze zmiany w transporcie i rolnictwie, a pejoratywnych rezultatów postępującej technologii przyjdzie doświadczać głównie osobom zarabiającym mniej niż dwadzieścia dolarów za godzinę pracy i tym, którzy nie ukończyli szkoły średniej. Wymienia także pozytywne skutki wdrażania nowoczesnych technologii oraz sztucznej inteligencji, jak na przykład to, że przyczynią się one do zwiększenia wydajności pracy – roboty zastępujące człowieka stają się bowiem coraz szybsze i sprawniejsze. Inteligentne maszyny można tez traktować jako sposób na obniżanie wydatków firm związanych ze podwyższaniem płac i kosztów opieki zdrowotnej. Roboty oczywiście wymagają konserwacji i zdarza się, że zawodzą, na przykład psując się, ale łatwiej nimi zarządzać niż ludźmi, a dodatkowo nie stwarzają problemów natury prawnej[9] .

Według Darrella Westa, dyrektora Centrum Innowacji Technologicznej w ośrodku Brookings Institution, automatyzacja procesów produkcji i usług będzie postępować bez względu na to, czy nowa administracja Stanów Zjednoczonych zachowa stawkę płacy minimalnej. Roboty będą coraz tańsze w eksploatacji. Boston Consulting Group prognozuje, że do 2025 roku koszty operacyjne robota-spawacza spadną poniżej dwóch dolarów za godzinę (dla porównania – wynagrodzenie spawacza wynosi dzisiaj w USA 25 dolarów za godzinę). Maszyny są coraz sprawniejsze nie tylko w wymienionej dziedzinie. Sprzęt oparty na sztucznej inteligencji zastępuje pracowników na przykład w restauracjach typu Fast food, centrach obsługi telefonicznej, na lotniskach, w sklepach, jak również w usługach transportowych. I tak na sytuację na rynku pracy wpłynie również wprowadzanie autonomicznych samochodów. Według zrzeszenia związków zawodowych transportowców American Trucking Associations w Stanach Zjednoczonych w 2010 roku około trzech milionów osób pracowało w zawodzie kierowców ciężarówek, a prawie siedem milionów pracowników zajmowało stanowiska powiązane z prowadzeniem działalności przewozo-

6. Tamże.
7. Tamże.
8. http://www.pap.pl/aktualnosci/news,753042,usa---raport-z-postepem-robotyzacji-znikaja-miejsca-pracy.html [dostęp: 08.09.2017 r.].
9. Tamże.

wej, także przy produkcji ciężarówek oraz ich obsłudze. W sumie jeden na piętnastu pracowników w USA był zatrudniony w sektorze transportu samochodowego. Według resortu pracy, około trzysta tysięcy osób jest taksówkarzami oraz kierowcy. Niebawem spora część z nich może stracić pracę [10].

W przeciągu następnej dekady technologia zdziesiątkuje miejsca pracy w wielu zawodach. Ludzie najsłabiej wykształceni, nawet jeśli nie zostaną bezrobotni, będą mogli liczyć na pracę mniej płatną niż obecnie i pozbawioną lub zapewniającą minimalne świadczenia. Za najbardziej bezpieczne miejsca pracy uznaje się aktualnie głównie stanowiska wymagające wysokiego poziomu kreatywności, myślenia analitycznego i dobrej komunikacji interpersonalnej, czyli predyspozycji, których na razie od maszyn wymagać nie możemy.

Apel Billa Gatesa w kwestii opodatkowania maszyn i robotów

Bill Gates – najbogatszy człowiek świata z 2016 roku (zgodnie z rankingiem amerykańskiego dwutygodnika o tematyce biznesowej „Forbes") w wywiadzie dla Quartza postulował opodatkowanie robotów. Stwierdził, że pieniądze z owych podatków mogłyby na przykład zasilić edukację lub działania aktywizacyjne. Dodał ponadto, że jeśli oczywiste dla społeczeństwa jest opodatkowanie pracy ludzkiej, tak samo niezaskakujące powinno stać się niebawem opodatkowanie pracy maszyn. Za opodatkowanie miałyby, według Gatesa, odpowiadać przedsiębiorstwa, które wykorzystują roboty[11] .

Współzałożyciel korporacji Microsoft mówił także o zadaniach, jakie stoją przed współczesnym światem. Wymienił między innymi pomoc dzieciom specjalnej troski oraz osobom starszym, która może być realizowana właśnie poprzez podatki pochodzące z pracy robotów. Zgodnie z myślą Gatesa istnieje masa sposobności na zwiększenie produktywności oraz wygenerowanie większych sum podatków. Dodał, że jeśli dziś zastąpienie pracownika przez maszynę powoduje wzrost efektywności i zwiększenie zysków przedsiębiorstwa, powinniśmy wprowadzać tego typu zmiany, nie zapominając przy tym o opodatkowywaniu robotów. Zauważył też potrzebę wszczęcia działań niwelujących pejoratywne społeczne skutki automatyzacji i robotyzacji tak, aby nowe technologie nie kojarzyły się ludziom z czymś negatywnym i by nie wywoływały lęków[12].

Refleksje Billa Gatesa w licznych kwestiach pokryły się z raportem Baracka Obamy zatytułowanym „Sztuczna inteligencja, automatyzacja i ekonomia" analizowanym powyżej. Biały Dom podkreślił bowiem, że automatyzacja i rozwój sztucznej inteligencji dotknie głównie warstwy słabo wykształcone i najmniej zarabiające. Zarówno Barack Obama, jak i Bill Gates zauważyli potrzebę systemowej pomocy osobom, których tyczyć się będą owe zagrożenia.

Podsumowanie

W dzisiejszym świecie już nie tylko cieszymy się powstałymi rozwiązaniami, lecz także wciąż pracujemy nad nowymi wynalazkami. XX i XXI wiek to czas prężnie rozwijających się innowacyjnych technologii, czas maszyn i robotów, które wypierają pracę człowieka na coraz to większą skalę. Automatyzacja i robotyzacja są procesami nie do zatrzymania. Społeczeństwo odczuwa więc lęki związane z utratą stanowisk i, co za tym idzie, dochodów. Mechanizmy zastępują na przykład copywriterów, pracowników służby zdrowia, operatorów urządzeń oczyszczania ścieków, pracowników restauracji, centrów obsługi telefonicznej, lotnisk, sklepów, usług transportowych i wielu innych.

Autorytety takie jak Barack Obama czy Bill Gates uspokajają, podając rozwiązania przyczyniające się do wzbogacania się skarbu państwa oraz społeczeństwa mimo, a dokładniej poprzez zautomatyzowanie świata. Takim rozwiązaniem jest na przykład opodatkowanie maszyn i robotów przez przedsiębiorców. Pozostaje jeszcze inna, nie mniej istotna kwestia – urządzenia technologiczne muszą zostać przez człowieka zaprojektowane, później zbudowane. Dodatkowo – tylko człowiek posiada zdolność monitorowania i nadzorowania ich. Trudno więc mówić o całkowitym zastąpieniu ludzi przez choćby najbardziej nowoczesne i rozwinięte technologie.

Bibliografia

Źródła pisane:

1) A. Grycuk, Klastry a rozwój regionalny. Klaster usług biznesowych w Krakowie, Kraków 2017, s. 145–146.

2) A. K. Gupta, Industrial Automation and Robotics. Laxmi Publications (P) Ltd., 2007, s. 1.

3) J. Szyłak, M. Skutnik, Rewolucje. Pełna automatyzacja, Warszawa 2006.

Źródła internetowe:

1) https://qz.com/911968/bill-gates-the-robot-that-takes-your-job-should-pay-taxes/ – tłum. własne [dostęp: 08.09.2017 r.].

2) http://www.pap.pl/aktualnosci/news,753042,usa---raport-z-postepem-robotyzacji-znikaja-miejsca-pracy.html [dostęp: 08.09.2017 r.].

3) http://tvn24bis.pl/wiadomosci-gospodarcze,71/dziesiec-technologii-przyszlosci-ktore-zmienia-nasze-zycie,402504.html [dostęp: 08.09.2017 r.].

10. Tamże.

11. https://qz.com/911968/bill-gates-the-robot-that-takes-your-job-should-pay-taxes/ [dostęp: 08.09.2017 r.].

12.Tamże.

Wykorzystanie nowoczesnych technologii w zarządzaniu firmą ochroniarską - skuteczność dronów w działaniach monitorujących

The use of modern technologies in the management of a security company - the effectiveness of drones in monitoring activities

Piotr Dąbrowski[1], Jędrzej Dąbrowski[1], Ewa Kardasz[2]

Streszczenie: Artykuł traktuje o aktualnej sytuacji firm ochroniarskich na rynku pracy. Opisuje zadania firm ochroniarskich, a także czynniki składające się na dobry wizerunek owych przedsiębiorstw. Dalej mówi o potrzebie wprowadzania innowacyjnych technologicznych rozwiązań w firmach ochroniarskich. Wymienia i charakteryzuje tego typu rozwiązania: kamery wizyjne, kamery termowizyjne, kamery z funkcją identyfikacji twarzy, czujniki ruchu, aplikacje mobilne oraz bezzałogowe statki powietrzne. Kolejno opisuje liczne możliwości wykorzystania dronów przez firmy ochroniarskie, a także wykazuje, że urządzenia te pozwalają na redukcję etatów w przedsiębiorstwach, a co za tym idzie – przynoszą oszczędności. W podsumowaniu tekstu znajduje się informacja, że bezzałogowe statki powietrzne znajdują coraz częściej zastosowanie w firmach ochroniarskich – aktualnie w wypadku obszarów zewnętrznych, w przyszłości – na terenach zewnętrznych.

Abstract: The article assesses the current situation of security companies in the labor market. Describes the tasks of security companies, as well as the factors that make up the good image of these enterprises, the need to introduce innovative technological solutions in security companies. Exchange and characterize such solutions: video cameras, thermal imaging cameras, cameras with face identification, motion detectors, mobile applications and unmanned aerial vehicles. Subsequently, it describes the numerous possibilities of using drones by security companies, and also shows that these devices allow the reduction of jobs in enterprises. Unmanned aerial vehicles are increasingly used in security companies - currently in the case of external areas, in the future - in external areas.

Słowa kluczowe: firma ochroniarska a drony, ochrona o drony, ochrona osób oraz mienia a drony, zadania firm ochroniarskich, zadania ochroniarza, sytuacja firm ochroniarskich na rynku pracy, dobry wizerunek firm ochroniarskich, nowości w firmach ochroniarskich, innowacje w firmach ochroniarskich, nowe technologie w firmach ochroniarskich, wykorzystanie dronów w firmach ochroniarskich, zastosowanie bezzałogowych statków powietrznych w firmach ochroniarskich, kamery wizyjne, kamery termowizyjne, kamery z funkcją identyfikacji twarzy, czujniki ruchu, aplikacje mobilne

Keywords: security company and drones, protection of drones, protection of persons and property and drones, tasks of security companies, tasks of a security guard, situation of security companies on the labor market, good image of security companies, news in security companies, innovations in security companies, new technologies in security companies , use of drones in security companies, the use of unmanned aerial vehicles in security companies, video cameras, thermal imaging cameras, cameras with face identification, motion detectors, mobile applications

Zadania firm ochroniarskich

Firmy ochroniarskie to przedsiębiorstwa, których celem jest bezpośrednia ochrona fizyczna osób bądź mienia, a także montaż zabezpieczeń technicznych. Wymienione dwa segmenty są obowiązkowe w działalności ochroniarskiej. Muszą więc występować łącznie i nie mogą stanowić osobnych usług [2]. Działania firm ochroniarskich mogą być realizowane w postaci stałej lub doraźnej [6]. Do elementarnych zadań omawianych przedsiębiorstw należy [2]:

- Ochrona mienia.
- Monitorowanie sygnałów przesyłanych, gromadzonych, a także przetwarzanych w sprzętach elektronicznych.
- Monitorowanie sygnałów przesyłanych, gromadzonych, a także przetwarzanych w systemach alarmowych.

1. DB SYSTEM Wrocław, Marsz. Józefa Piłsudskiego 95, kod pocztowy 50-016
2. Wyższa Szkoła Zarządzania i Coachingu Wydział Inżynieryjny al. Śląska 1 54-118 Wrocław

- Zabezpieczenie techniczne.
- Transport pieniędzy.
- Transport przedmiotów wartościowych.
- Transport przedmiotów niebezpiecznych.
- Montowanie systemów alarmowych.
- Konserwowanie i naprawienia systemów alarmowych.
- Instalowanie sprzętów oraz systemów mechanicznego zabezpieczenia.

Sytuacja firm ochroniarskich na rynku pracy

Aktualnie w Polsce firmy ochroniarskie przeżywają znaczny rozwój. Polega on na zwiększeniu wymagań osób pracujących w firmach ochroniarskich, zwiększeniu wytycznych prawnych, a także wprowadzaniu innowacyjnych urządzeń i działań do przedsiębiorstw. Szacuje się, że w naszym kraju w 2016 roku liczba omawianych aktywnych podmiotów gospodarczych przekroczyła 2 tysiące. Liczba pracowników firm ochroniarskich szacowana jest na około 300 tysięcy osób, z kolei wykwalifikowani pracownicy to niespełna 20% z nich, czyli około 60 tysięcy osób. Szacowana wartość rynku ochrony to ponad 8 milionów złotych w ciągu jednego roku. Liczby pokazują, że zapotrzebowanie na rynku pracy na firmy ochroniarskie jest duże. Mówią także jednak o sporej konkurencyjności w owej branży [2].

Czynniki składające się na dobry wizerunek firm ochroniarskich

Aby firma ochroniarska miała dobry wizerunek, a co za tym idzie – cieszyła się zaufaniem oraz wieloma zleceniami, musi cechować się: 1) skutecznością; 2) profesjonalizmem w realizacji powierzonych zadań; 3) przyjaznym podejściem do klienta; 4) traktowaniem klienta „po partnersku"; 5) proaktywnością; 6) elastycznością; 7) mobilnością; 8) innowacyjnym podejściem; 9) rozbudowanym obszarem „Safety" będącym rozwinięciem „Security"; 10) umiejętnością przenoszenia akcentów wykonawczych z „twardej" ochrony na inne formy zabezpieczeń; 11) testowaniem nowoczesnych technologii; 12) wykorzystywaniem nowoczesnych i sprawdzonych technologii (tu na przykład bezzałogowych statków powietrznych powszechnie, znanych jako drony) [2].

Potrzeba wprowadzania innowacyjnych rozwiązań w firmach ochroniarskich

Właściciele firm ochroniarskich są zmuszeni do wykorzystywania innowacyjnych rozwiązań i urządzeń, aby móc utrzymać swoją pozycję na rynku. Technologia pomaga w widocznym zminimalizowaniu kosztów prowadzenia działalności, które wzrosły w ostatnich latach (stało się tak między innymi z powodu ozusowania umów zleceń oraz deficytu kadry pracowniczej). Dodatkowo, od stycznia 2018 roku podniesiona została minimalna stawka godzinowa dla pracowników firm ochroniarskich – aktualnie wynosi ona 13 zł brutto (jeszcze kilka, kilkanaście lat temu stawka ta była o ponad połowę mniejsza – wynosiła nawet 6 zł brutto). W związku z tą zmianą, utrudnione zostaną relacje z klientami, których świadomość i wymagania wciąż rosną. Z tego właśnie powodu branża od dłuższego czasu poszukuje innowacyjnych rozwiązań. Poprzez wykorzystywanie nowoczesnych technologii agencje ochroniarskie mogą ograniczyć liczbę pracowników o nawet 50% lub rozwiązać problem niedoborów kadrowych [2].

Przykłady innowacyjnych rozwiązań w firmach ochroniarskich

Profesjonalne usługi ochroniarskie podążają za prężnie rozwijającym się światem i reagują nie tylko na znane, lecz także na nowe, dopiero pojawiające się zagrożenia. Ochrona mienia nie polega więc już wyłącznie na ochroniarzu i nieskomplikowanym monitoringu przemysłowym. Innowacyjne rozwiązania pomagają w zabezpieczaniu mienia oraz pozwalają zniwelować niebezpieczeństwa pojawiające się w świecie wirtualnym. Takimi urządzeniami są na przykład kamery wizyjne, kamery termowizyjne, kamery z funkcją identyfikacji twarzy, czujniki ruchu, aplikacje mobilne. Ich skrótowy opis przedstawiony został poniżej.

- **Kamery wizyjne** – wysoko rozwinięte urządzenia, posiadające dużą liczbę pikseli. Kamery te są wykorzystywane od dawna, współcześnie jednak kładzie się nacisk na unowocześnienie i usprawnienie ich w taki sposób, by nie były zależne od pogody, oświetlenia i wielu innych czynników.
- **Kamery termowizyjne** – optoelektroniczne urządzenia obrazowe, które analizują temperaturowe promieniowanie podczerwieni. W wypadku firm ochroniarskich stosuje się kamery termowizyjne obserwacyjne (występują jeszcze wersje pomiarowe). Pozwalają one na monitorowanie obiektów w warunkach ograniczonej widzialności.
- **Kamery z funkcją identyfikacji twarzy** – program ten mierzy odległość między różnymi punktami na twarzy, na przykład pomiędzy oczami, brwiami, kącikami ust, długością między nosem a ustami. W skutek tego kamery z funkcją identyfikacji twarzy pozwalają na: wykrywanie wielu twarzy oraz ich rozpoznawanie; przesyłanie informacji związanych z identyfikacją do administratorów; proste dodawanie osób do bazy danych; przeszukiwanie bazy danych za pomocą rozmaitych filtrów; sposobność wybrania interfejsu bazy danych; sposobność zapisywania każdej twarzy w bazie danych; sposobność grupowania twarzy, rejestrowanie wszelkich obrazów twarzy z datą oraz indeksem czasowym; łatwe i nieczasochłonne przeszukiwanie historii rejestrowania przy użyciu filtrów.
- **Czujniki ruchu** – wykorzystuje się je zarówno w kamerach wizyjnych, jak i termowizyjnych monitorujących przestrzenie zamknięte. Czujniki stanowią wsparcie

dla operatora poprzez informowanie o wykryciu nieuzasadnionego ruchu oraz automatycznego przekazywania obrazu z alarmującej kamery. Kamery wyposażone w czujniki ruchu umieją rozpoznawać na przykład: samochody, zwierzęta, osoby, które biegną, osoby, które przepychają się w tłumie i inne. Opisywane urządzenie pozwala wyłonić niepokojący obraz na: lotniskach, dworcach, stadionach, ulicach i tym podobnych oraz wskazać go obserwującemu pracownikowi ochrony.

- Aplikacje mobilne – stanowią one ogromne ułatwienie dla pracowników ochrony. Pozwalają na monitorowanie ochrony fizycznej, raportowanie zadań zrealizowanych przez ochroniarzy, natychmiastowe przesyłanie informacji o zagrożeniu, a także oglądanie monitoringu z dowolnego miejsca i o dowolnej porze.

Wykorzystywanie dronów przez firmy ochroniarskie

Innym innowacyjnym rozwiązaniem coraz chętniej wykorzystywanym przez firmy ochroniarskie są drony. Bezzałogowe statki powietrzne kojarzone były niegdyś wyłącznie z działalnością militarną, dzisiaj znajdują wiele zastosowań komercyjnych. Jak więc można wykorzystać drony w działalności gospodarczej i jej systemach bezpieczeństwa? Sposobności wykorzystania jest wiele, przy czym należy dodać, że próby wykorzystania dronów zewnątrz pomieszczeń były dość liczne, inaczej jest natomiast z ich zastosowaniem wewnątrz obiektów [1].

Im bardziej dron jest rozbudowany oraz im większy ma zasięg, tym więcej jest jego potencjalnych zastosowań[7]. Zmodernizowana, nowoczesna konstrukcja, spora odporność na zjawiska atmosferyczne oraz sposobność precyzyjnego sterowania to tylko nieliczne z cech maksymalizujących potencjał bezzałogowych statków powietrznych w firmach ochroniarskich. Drony, szczególnie takie, które są wyposażone w systemy telemetryczne, kamery telewizyjne bądź kamery termowizyjne coraz częściej stosowane są w systemach bezpieczeństwa, takich jak ochrona obiektów [4]. Wyłącznie jeden bezzałogowy statek powietrzny jest w stanie patrolować obszar, do którego kontroli potrzebnych byłoby nawet kilkaset osób. Poprzez zastosowanie dronów można nie tylko zmniejszyć koszty prowadzonej działalności, lecz także zwiększyć efektywność patroli, a co za tym idzie – zmaksymalizować wykrywalność oraz zadowolenie klientów [1].

Nietrudno wywnioskować, że już niedługo bezzałogowe statki powietrzne będą wspomagać niemalże wszystkich operatorów zintegrowanych systemów bezpieczeństwa w ich pracy. W 2016 roku podczas międzynarodowych tragów Security Essen firma Ela-soft zaprezentowała technologiczną nowość w zakresie technik zabezpieczeń. Był to BSP z systemami bezpieczeństwa GEMOS. System ten posiad ponad 750 interfejsów służących do komunikacji z tradycyjnymi systemami zabezpieczeń. Są to na przykład systemy przeciwpożarowe, systemy sygnalizacji włamania, systemy sygnalizacji napadu, systemy kontroli dostępu, wizyjne systemy dozorowane i wiele innych. Możliwe jest także zintegrowanie systemu bezpieczeństwa GEMOS oraz dronów w taki sposób, by wywoływać interakcję wybranych podsystemów [3].

Grafika realizowana na podstawie współrzędnych daje możliwość wyznaczenia trasy przelotu automatycznego bez kontroli człowieka. Drony wyposażone w kamery mogą więc być wsparciem systemów ochrony obwodowej. Wyznaczając trasę bezzałogowego statku powietrznego, można użyć dronu do dodatkowego patrolowania. Dron przelatuje wtedy wyznaczoną wcześniej trasę, czyli obszar strzeżony i przekazuje natychmiastowo obraz operatorowi. Programując dany bezzałogowy statek powietrzny, można wyznaczyć dowolną trasę, którą urządzenie będzie podążało, a także dowolną liczbę punktów na tej trasie. Można również zaprogramować misje cykliczne, czyli stałe trasy programowe. Po przelocie zadaną trasą BSP wraca do stacji ładującej, co jest kolejnym dowodem na jego samodzielność i to, że przyczynia się on do redukcji zatrudnionych w firmie ochroniarskiej osób [3].

Ciekawą funkcją urządzenia jest także moduł analizy obrazu. Zaobserwowane anomalie zostają przedstawione operatorowi w czasie rzeczywistym. W sytuacji kryzysowej operator może wysłać dron w wybrane miejsce, aby przy użyciu kamery sprawdzić przyczynę alarmu. Jeżeli okaże się, że alarm jest prawdziwy, operator będzie miał możliwość sterowania dronem i obserwowania niepowołanych osób. Drony, które mają wbudowany miernik stężenia gazu, miernik temperatury bądź też kamerę termowizyjną mogą generować sygnał alarmowy. Wtedy, jeśli czujnik zareaguje na jakąś anomalię, natychmiastowo zaalarmuje operatora [5].

Podsumowanie

Sytuacja firm ochroniarskich na rynku pracy w ciągu kilku ostatnich lat znacząco się zmieniła. Stawki godzinowe dla pracowników zwiększyły się, a zapotrzebowanie na usługi wyżej wymienionych przedsiębiorstw wciąż wzrasta. W związku z tym firmy ochroniarskie wykorzystują coraz to nowsze, innowacyjne urządzania i rozwiązania, które pozwalają na zmniejszenie etatów oraz ponoszonych kosztów. Tego typu urządzaniami są: kamery wizyjne, kamery termowizyjne, kamery z funkcją identyfikacji twarzy, czujniki ruchu, aplikacje mobilne, a także coraz chętniej wykorzystywane – bezzałogowe statki powietrzne [4]. Aktualnie drony w firmach ochroniarskich stosuje się głównie do monitorowania obiektów zewnętrznych, rozwój techniki pozwala jednak przypuszczać, że za kilka lat znajdą one swoje zastosowanie w monitorowaniu pomieszczeń zamkniętych. Drony dają wiele korzyści właścicielom przedsiębiorstw. Jedną z nich jest redukcja kosztów, inną – wzrost wykrywalności, jeszcze kolejną – zwiększanie zadowolenia klientów. Wszystko to składa się na mnogość zleceń, umocnienie pozycji oraz rozwój firm ochroniarskich [4].

Literatura

[1] Audronis Ty, „Drony. Wprowadzenie", Wydawnictwo Helion, Gliwice 2015.
[2] Bejgier Waldemar, „Ochrona osób i mienia", Oficyna Wydawnicza Łośgraf, Kraków 2012.
[3] Cheng Eric, „Drony. Tajniki zdjęć i filmów lotniczych", Wydawnictwo Helion, Gliwice 2016.
[4] LaFay Mark, „Drony dla bystrzaków", Wydawnictwo Septem, Warszawa 2016.
[5] Killby Terry, Killby Belinda, „Make: Drony dla początkujących. Konstrukcja i dostosowanie własnego quadcoptera", Wydawnictwo Promise, Warszawa 2016.
[6] Pajorski Paweł, Kręgules Rafał, „Ustawa o ochronie osób i mienia. Komentarz", Wydawnictwo Wolters Kuwer, Warszawa 2015.
[7] Piotr Kardasz, Jacek Doskocz, Mateusz Hejduk, , Paweł Wiejkut, Hubert Zarzycki, „Drones and Possibilities of Their Using" Wydawnictwo OMICS, Civil Environ Eng 2016, 6:3 http://dx.doi.org/10.4172/2165-784X.1000233

Komentarze w kodach wybranych programów

Comments in the codes of selected programs

Sviatoslav Skhut[1], Kateryna Iholkina[2], Piotr Kardasz[3]

ABSTRACT : Writing comment as important as a code writing. The main purpose of using comments is improve readability of our code but frequently thoughtless comment writing decrease understandability of source code. Comments must be concise and precise simultaneously. Also, when our code is changed comments for this code must be changed too. Using comments in our code we must realize that if expressiveness of our programming language allows us to express clearly what we want in code there are no need comments at all. And if we decide to use comments they must be extremely accurate and understandable. Because another person must understand what we do here and most importantly why we do it.
Frequently comments can be replaced with good chosen name of variable, function or class. Also, we can replace our comments with assertions. The best way to use comments is clarification and explanation of intent. Copyrights and an authorship can be implemented using comments too. But our IDE can do these things automatically.
Classification of comments depends on their place in code, for which type of code they are attached and format.

Streszczenie : Artykuł odnosi się do analizy oraz komentarzy używanych podczas programowania. Komentarz jest ważnym elementem pisania kodu. Cel używania komentarzy tj. poprawa czytelności naszego kodu. Komentarze używane do opisywania kodu źródłowego muszą być zwięzłe i dokładne jednocześnie. Podczas zmiany kodu należy zmieniać także komentarze do tego kodu. Używając komentarzy w naszym kodzie, musimy zdać sobie sprawę, że jeśli ekspresja naszego języka programowania pozwala nam wyraźnie wyrazić to co najistotniejsze to nie ma potrzeby używania komentarzy. A jeśli zdecydujemy się użyć komentarzy, muszą one być niezwykle dokładne i zrozumiałe. Często komentarze można zastąpić dobraną nazwą zmiennej, funkcji lub klasy. Możemy również zastąpić nasze komentarze asercjami. Najlepszym sposobem wykorzystania komentarzy jest wyjaśnienie i wyjaśnienie intencji. Prawa autorskie i autorstwo można zaimplementować również za pomocą komentarzy. Klasyfikacja komentarzy zależy od ich miejsca w kodzie, rodzaju kodu, do którego są dołączone i formatu.

Keywords: source code, software and its engineering, documentation, software management, code comments

Słowa kluczowe : kod źródłowy, oprogramowanie i jego inżynieria, dokumentacja, zarządzanie oprogramowaniem, komentarze do kodu

INTRODUCTION

Improving code commenting techniques are important for programmers for several reasons: code maintenance takes about 40-80% of the lifetime cost of a piece of software [1], besides software is rarely maintained by its original author for its whole life. It's obvious that, the perception of the code author is very different from those of the following developers. Some moments and solutions, self-explanatory for the author, can create a lot of problems for programmers supporting or refactoring the program. Due to poor documentation and poor-quality comments, it is often even easier to write a new program than to change an old one. Thus, writing a code with useless, unreliable and inaccurate comments provides to large increase in cost.
Unfortunately, there is no general standards and conventions of writing and formatting code comments. In last twenty years there was published a lot of books, thesis and blog posts on the topic of programming style, clean code writing and code documentation. Namely, „The elements of programming style" by Brian W. Kernighan and P. J. Plauger, a study supporting point of view that source code should be, first of all, human readable. We should also note Robert C. Martin`s book „Clean Code. A Handbook of Agile Software Craftsmanship". A huge part of this study describes patterns and practices of writing maintainable

1. Wrocławska Wyższa Szkoła Informatyki Stosowanej „Horyzont", ul. ks. Marcina Lutra 4 (poprzednio ul. Wejherowska 28), 54 - 239 Wrocław
email: K_Iholkina@gmx.com
2. Wrocławska Wyższa Szkoła Informatyki Stosowanej „Horyzont", ul. ks. Marcina Lutra 4 (poprzednio ul. Wejherowska 28), 54 - 239 Wrocław
email: sgshkut@gmail.com
2. Piotr Kardasz – Wrocławska Wyższa Szkoła Informatyki Stosowanej we Wrocławiu, Wydział Automatyki i Robotyki

code and efficient comments. In addition to these books our essay is based on „Java code conventions", published by Sun Microsystems Inc. In 1997 and „C++ style guide" powered by Google Inc.
The aim of our paper is to treat a problem of code commenting in the source code of selected programs. In the first two chapters we pay attention on the different code comment`s classification based on their function, placement and format. In the third chapter we provide an analysis of costs and benefits of using comments. On the examples of different programs, we consider the best practices of code commenting and comments, that should be in each program, such as legal comments, clarifications and TODO comments and explanations of intent. Also, we study examples of useless and sometimes even dangerous comments, that should never appear in the source code. The last part of the third chapter is devoted to techniques that make the code more clear and readable without using comments.

1. COMMENT TYPES

Comments can be used for different purposes in different parts of the program. They fall into one of three categories: header comments, block comments and trailing comments, which describe successively smaller areas of code.

1.1. Header comments.

This class of comments serves to help reader navigate, understand a general purpose of the code and use code it. This category of comments makes easier a maintenance of code. Thus, they should be included in any code planned to be in use more than a few weeks. The header comments usually occupy a number of lines (typically between 10 and 50) [2] and contains following elements:

- Filename
- Source control version history
- Creation date
- Revision history.
- Author's name
- Copyright notice
- License Summary
- Purpose
- Change history
- Restrictions.
- Special hardware requirements (e.g. Analog/Digital signal converters)

Header comments are placed at the beginning of the program which makes them stand out and easier to remove or copy. The recommended practice is dividing a comment block onto the section to ameliorate its readability. Using capital letters for the section headings and tabbing information out allows navigate and read them more quickly:

```
/*
 ...
 * GLOBAL DATA:
 *      int     DB_ErrStatus          Contains most recent database error
```

1.2. Class comments.

If we have a non-obvious class, there are comments required. These types of comments should describe what this class for and how it should be used. The class comments should provide such information as: interface of the class, multiple threads (if any) and if it possible a few small examples of code demonstrating its usage. When we separate implementation and declaration (e.g. .cpp and .h files), comments that describe use of the class should go together with declaration, comments that describe class operation and implementation should be insert into implementation file.

1.3. Function comments.

As with class function comments should appear when usage of the function is non-obvious. Comments attached to function declaration should describe the usage of the function. They shouldn't describe how the function performs its tusks, just tell reader in descriptive way what function must do, what inputs and outputs are, in which way user must free memory (if the function allocates memory) also function override should be described if it not trivial. Comments in function definition should describe operations. These types of comments describe in which way function works.

1.4. Variable comments.

The actual name of variable should be enough for description of what it is used for. Comment can be attached if this variable need additional clarification. In classes we have data members. They name must be enough descriptive too and comments required if there are some non-obvious instances. Global variables should have accompanying comments that describe why they need to be global.

1.5. Explanatory comments.

Well written and concise code will contain a lot of explanatory comments, which highly increase code readability and clearness. Even though explanatory comments are not necessary for each line of code, there are some items which definitely should have them.
For example, startup code, exit code, weird logic, regular expressions, sub routines and functions, long and complicated loops [3].
In startup code explanatory comments is a good practice to mention how the program is initialized, what #defines do, what arguments are expected etc.
In exit code explanatory comments, we should properly treat normal and abnormal exit situations, error codes etc.

Sub routing and function explanatory comment should first of all clearly explain its task and purpose of its using. It is important to comment function's arguments passed and returned with mentioning value`s format and limits on values expected, as this is one of the greatest sources of bugs [4]. Thus, this kind of comments written before sub routines greatly helps reader to gain a deep understanding of the following code.

2. COMMENT FORMATS

2.1. Block comments.

Block comments are generally found within functions, methods, data structures and algorithms. Block comments have two main purpose:

- Commenting out code
- Writing long comments

In C, C++ and Java languages block comments are began with special separator :"/*" and terminated with "*/" as shown in the example below. A common used practice is to start block comments by a blank line to separate them from the rest of code.

```
/* * Here is a block comment. */
```

The other technic of highlighting block comments is an enclosing them into a rectangle box of stars and dashes.
The example of boxed comment is:

```
/**********************
 Comment in a box!!
**********************/
```

The initial "/*" could be followed by other characters such as "=", "_" or "-".

2.2 Single-line comments.

Single-line comment is a short comment which appears on a single line indented of the code that follows. As well as block comments there is highly recommended to separate them from the rest of code by at least one blank line.
The example of single-line comment in Java code is:

```
if (condition) {
   /* Handle the condition. */
   ...
}
```

2.3. Trailing comments.

Trailing comments are very short comments which describe the action or use of a single line of code. They usually appear (and end) on the same line as the code they describe. For separating trailing comment from code, it is common practice to tab it out. The comment should be far enough away from the code.

For example:

```
SelectSides( Players );          /* choose partners and positions  */
```

3. PROS AND CONS OF COMMENTING CODE

3.1. Why comments are not always good.

Comment can be very helpful if it placed in correct place in right time. But frequently they just do mess or clarify the code which we can understand without them. We need comments for clarification our motives, why we write our code that way or even for warning about something. Nature of comments arises from low expressiveness of our programming languages. Sometimes we should write code that few people can understand and that way comments allow us to do it. The best comment is that we don't need at all. That means that we can chose a good name for variable or function, decide to write an extra line that make our code more readable and understandable.
Another case is evolving our code. Chunks of it can be moved in another place or delete or even rewrite. In this case we can face with outdated misleading comments. Of course, programmers can maintain these comments, but it takes a lot of time and the better way is to change a piece of code and deal with comment only we really need it in this place.
When we decide to write a comment first we need is thought about other people which will read it. And the second we should do our best with this comment. Time spending for writing a good comment will save a lot of time our code readers.
Sometimes we can see redundant comments. They do nothing except amass lines in our code. It means that there is no need single comment for every function or variable we declared and when we see a + b we already know what it does and comment no need again. Usually this type of comments just makes difficult to read the code.
Now we can see that comment not always helpful. There are a lot of cases when comments not need at all or when they just make code less readable.

3.1.1. Journal comments.

Some programmers add a kind of „historical" comments containing their names, time and date and changes made every time they edit code. For example:

```
// method name: pityTheFoo (included for the sake of redundancy)
// created: Feb 18, 2009 11:33PM
// Author: Bob
// Revisions: Sue (2/19/2009) - Lengthened monkey's arms
//            Bob (2/20/2009) - Solved drooling issue
 void pityTheFoo() {
```

```
    ...
}
```

Eventually, accumulate as a kind of journal of every modification has ever been made. There were some reasons to make log comments long-long ago, when there wasn`t a source control system making it automatically for us. Nowadays is more likely to use one of the source control system and just fill the check-in comments box on our commits [5].

3.1.2. Noise comments.

Sometimes comments are obvious and provide no new information. For example, each string of comments from code below can be discarded without loss of understandability:

```
/** The name. */
private String name;
/** The version. */
private String version;
/** The licenceName. */
private String licenceName;
/** The version. */
private String info;
```

Such code out commenting normally should be used in two cases: in code examples to teach the concept of programming language, or in the case when programming language isn`t easy human readable (Assembly).

3.2. Good comments.

There are cases when we can't avoid using comments such as corporal rules or copyright. But we will not consider them now. A good example of good comment is to do comment. This type of comments can appear near the functions which will implement in the future (or not). It just list of tasks that programmers want to do in future. Another example of good comments is warning comments. This type of comments warns other programmers about consequences of using code such as vulnerability or time of execution. Comments that describes our intents (why we decided to solve this problem this way or choose this data type etc.) or clarify our cod (when we really need it) are examples of good comments too.

3.2.1. Legal comments.

These comments should not be duplicate of contract or legal tome. Legal comments can include copyrights, refers to standard licenses, authorship. They also can refer to external document. This type of comments put at the beginning of source file.

3.2.2. Explanation of intent.

This type of comments explains why we have decided to use this implementation. It allows a developer to look at a piece of code and know why it exist. Also, it reduces situations where our intents aren't look clear at a glance.

3.2.3. Clarifications.

Sometimes the best way to tell developer about our code is write about it in readable form. For this reason, clarification comments can use. This type of comments helps us describe our obscure functions, returning values, non-obvious behavior etc.

3.2.4. TODO comments.

It is not a bad idea include some "todo" list in out code. It can be done with this type of comments. They can be connected to function or pieces of code that we want to implement in future. Todo comments show developers that this function does nothing except reminding.

3.3. Alternative to comments.

As we mentioned above the one of the worst practice is out commenting code. Obvious, annoying, trashy, redundant comments lead to incomprehensible hard to maintain code. Having considered what comments' strengths and weaknesses are, we will treat how they could be replaced by other tools.

3.3.1. Identifiers as comments.

Consider the example demonstrating how a typical comment can be encoded in an identifier:

Before:

```
 ++i;/* record another match of this expression */
```

After:

```
 ++number_of_expression_matches;
```

Huge part of source codes comments could be replaced by carefully and thoughtfully named variables, functions, methods and classes names. If a comment is intended to explain a complex expression, the expression should be split into understandable subexpressions using extract variable. If a comment explains a section of code, this section can be turned into a separate method via extract method.

However, identifiers could be completely misleading, if the programmer isn`t attentive when modifying code. This is the same problem as which appears when comments aren`t updated respectively to code modifications. So, when we refactor code we should be vigilant to change both comments and identifiers.

3.3.2. Replacement comments with assertions.

From time to time it is reasonable to refactor a comment into in assertion. For example:

Before:

```
// value must not be negative
 public double squareRootOf(int value) {
...square root algorithm...
}
```

After:

```
public double squareRootOf(int value) {
Assert.isTrue(value >= 0);
   ...square root algorithm...
}
```

The benefits of this solutions are: supporting better testing, making debugging easier, serving as understandable comment about preconditions [6]. However, assertions slow down our code and should make a program incorrect when they are used improperly. So, assertions have the advantage they are enforced as code and form programmable safeguards, but then also have all the disadvantages of code: expression of abstraction can be verbose and non-trivial

4. DOCUMETING SYSTEMS

There are several documenting systems available for various programming languages. These systems deal with the "Explanatory" type of comment. They create documentation out of comments from the code. Let's demonstrate these systems via Javadoc. This is the Java API contained in the JDK. It uses comments with specific tag /** to generate HTML pages with descriptions of all classes, interfaces, constructors etc. It also generates a tree with class hierarchy. For more details we can use documentation provided with JDK. PHP and C# also have documentation system, but the latter uses XML instead of HTML. These systems require from developers maintain not only the code but comments too.

CONCLUSIONS

When reading tricky code, there is nothing more helpful than well-written comment. However, when writing code, there is often nothing harder than write a well-placed, brief and clear comment.

From the one hand, plain English is always easier to read than code. Comments can explain things that couldn`t be easy express in programming language, besides, they don`t affect program execution speed. Writing good comments discipline programmer`s mind. Comments are shorter than the code they document and much easier to skim-read.

From the other hand, they reduce the readability of well-written code, in addition they are less precise that the code they document. Sometimes using a lot of comments encourage bad code and take up screen space and time to read. By the way, programmers often refactor code, but don`t update comments which provide to a high risk of spending hours tacking up a bug because you trusted a non-reliable comment.

Thus, programmers should use comments carefully, for preference when it is impossible to make a code self-explanatory. Before writing a comment, it is recommended practice to try to increase code expressiveness by introduce an explaining variable, extract a method, use more descriptive identifier, or replace a comment with assertion.

The questions „To write or not to write?", „How many?", „How detailed comments to write?" still hotly debated one.

SOURCES

[1] Sun Microsystems Inc, Java code conventions, 1997

[2] David Straker, C Style: Standards and Guidelines, 1991

[3] Bernhard Spuida, The fine Art of Commenting, 2002, Tech Notes, general Series

[4] Brian W. Kernighan, P. J. Plauger, The elements of programming style, 1978, McGraw-Hill Book Company,

[5] Robert C. Martin, Clean Code. A Handbook of Agile Software Craftsmanship, 2009, Prentice Hall,

[6] Dori Reuveni, Kevin Bourrillion, Code Health: to comment or not to comment, 2017, blog post

Środowiska i narzędzia związane z wytwarzaniem oprogramowania

Environment and tools related to software development

Michał Kuliś[1], Jarosław Jazienicki[2], Patryk Stachura[3]

Streszczenie: Niniejsza praca porusza problematykę i kwestie związane ze środowiskami, narzędziami oraz szeroko pojętym wytwarzaniem oprogramowania. Przedstawia również przykłady środowisk i narzędzi, które są na dzień dzisiejszy najczęściej wykorzystywane. Dzisiejsze tempo życia, wpływ globalizacji oraz bardzo szybki rozwój technologiczny budzą coraz to większe wymagania społeczeństwa w stosunku do jakości wytwarzanych usług, w tym także oprogramowań różnego rodzaju urządzeń. Zakres oferowanych środowisk oraz narzędzi, które mają za główne zadanie ułatwiać oraz zwiększać możliwości jest coraz szerszy i bardziej rozbudowany. Społeczeństwo wymaga coraz to bardziej zaawansowanych oraz bardziej skutecznych pod względem innowacyjności usług i rozwiązań problemów, które są związane z życiem codziennym. Wymusza to ogromną presję na koncernach, korporacjach, firmach, które zajmują się właśnie wytwarzaniem oprogramowania. Obawy te związane są ze skutecznością zaspokojenia popytu na dane usługi.

Abstract: This work addresses issues and issues related to environments, tools and the broadly understood software development. It also presents examples of environments and tools that are mostly used today. Today's pace of life, the impact of globalization and very rapid technological development are raising the society's ever-increasing demands in relation to the quality of services produced, including software of various types of devices. The range of offered environments and tools that have the main task to facilitate and increase the opportunities is growing wider and more extensive. Society requires more and more advanced and more effective in terms of innovation services and solutions to problems that are related to everyday life. This forces huge pressure on corporations, corporations, and companies that are involved in the production of software. These fears are related to the effectiveness of satisfying the demand for given services.

Słowa kluczowe: oprogramowanie, środowisko programistyczne, IDE, narzędzie wytwarzania oprogramowania, język programowania.

Keywords: software, development environment, IDE, software development tool, programming language.

Wstęp

Na samym początku warto się zaznajomić z podstawowymi pojęciami związanymi z wytwarzaniem programowania, takimi jak:

- Oprogramowanie,
- Proces wytwórczy oprogramowania,
- Narzędzie programistyczne,
- Zintegrowane środowisko programistyczne,
- Język programowania.

Oprogramowanie - całość informacji w postaci zestawu instrukcji, zaimplementowanych interfejsów i zintegrowanych danych przeznaczonych dla komputera do realizacji wyznaczonych celów. Celem oprogramowania jest przetwarzanie danych w określonym przez twórcę zakresie. Oprogramowanie tworzą programiści w procesie programowania. Oprogramowanie pisane jest zazwyczaj przy użyciu różnych języków programowania z wykorzystaniem algorytmów. Programy przekształcające oprogramowanie z postaci źródłowej na binarną to kompilatory. Niektóre oprogramowanie, np. napisane w całości w językach interpretowanych, może występować tylko w jednej postaci, spełniającej oba zadania [1].

Proces wytwórczy oprogramowania - proces mający na celu wytworzenie oprogramowania. Oprogramowanie wytwarzane jest od stosunkowo niedawna, dlatego procesy wytwórcze oprogramowania szybko się zmieniają w czasie, zmienia się też często opinia na temat jakości i efektywności poszczególnych procesów [2].

Narzędzie programistyczne - program komputerowy służący do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. Do narzędzi programistycznych należą: kompilatory, asemblery, debuggery i zintegrowane

1. Student, Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Adres, mail michalkulis1993@gmail.com
2. Student, Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Adres, mail jarek.jazienicki@gmail.com
3. Student, Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Adres, mail stachura.patryk@yahoo.com

środowiska programistyczne [3].

Zintegrowane środowisko programistyczne, IDE - aplikacja lub zespół aplikacji (środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. Aplikacje będące zintegrowanymi środowiskami programistycznymi charakteryzują się tym, że udostępniają złożoną, wieloraką funkcjonalność obejmującą edycję kodu źródłowego, kompilowanie kodu źródłowego, tworzenie zasobów programu, tworzenie baz danych, komponentów i innych [4].

Język programowania - zbiór zasad określających, kiedy ciąg symboli tworzy program komputerowy oraz jakie obliczenia opisuje [5].

Na wstępie warto również zaznaczyć, iż w dobie dzisiejszego tempa rozwoju technologii wszelakie środowiska oraz narzędzie programistyczne ulegają diametralnej zmianie każdego dnia. Codziennie powstają nowe frameworki, mające na celu wspomaganie danych komponentów języków programowania, nowe narzędzia oraz nowe środowiska. Same języki programowania nie ulegają tak szybkiej zmianie, lecz są równie szybko rozbudowywane o nowe funkcjonalności i frameworki, o których wspomnieliśmy powyżej. Także śmiało możemy pokusić się o tezę, iż również proces wytwarzania ulega zmianom, ponieważ jest znacząco uzależniony od komponentów, które się na niego składają. Proces tworzenia oprogramowania składa się z poszczególnych etapów, jest to także uzależnione od modelu procesu, który się zaimplementuje w projekcie.

Problematyka wytwarzania oprogramowania

Współczesne metodyki wytwarzania oprogramowania wykorzystują wiele artefaktów takich jak: raporty o błędach, kod źródłowy, harmonogramy, dokumentacja czy testy. Zintegrowane środowiska oraz narzędzia wspomagają zarządzanie i tworzenie tych artefaktów. Bardzo często pochodzą one od innych producentów dlatego różnią się one zarówno zakresem pokrywania procesu produkcji oprogramowania jak i funkcjonalnością, które oferują Zarówno narzędzia jak i środowiska te są wyspecjalizowane pod względem funkcji jakie oferują. Ich główne przeznaczenie to m.in.: zarządzanie wymaganiami, tworzenie dokumentacji, modelowanie oprogramowania, wytwarzanie dokumentacji itp. Bardzo rzadkim zjawiskiem jest sytuacja, gdzie producent w swojej ofercie posiada produkty, które pokrywają cały proces wytwarzania oprogramowania wraz z możliwością komunikowania się oraz wymiany danych, aby ułatwić ten proces programistyczny.

W dzisiejszych czasach palącym problemem inżynierii oprogramowania jest utrzymywanie spójności oraz zgodności pomiędzy artefaktami projektu. Za przykład może nam tu posłużyć pamiętanie o odzwierciedlaniu zmian w kodzie źródłowym oraz w przedstawiających go diagramach UML, co jest koszmarem dla większości programistów. Wyjściem z tej sytuacji jest wsparcie tej czynności za pomocą narzędzi CASE, które wyposażone są w możliwość reinżynierii danego oprogramowania i integrację ze środowiskiem programistycznym. Omawiany problem występuje również przy takich procesach jak: aktualizacja dokumentacji, harmonogramów, testów oraz innych elementów projektu. Jest to trudność ogólna dotycząca reagowania na zmiany oraz propagowania tych zmian w projekcie, w taki sposób aby zachować spójność jego elementów. Istotny jest fakt, iż koszt który jest wymagany przez dokładną analizę zależności tzn. Formalny proces propagacji zmian, przeważnie jest większy niż potencjalne zyski. W związku z tym zauważalne jest, iż bardzo duży odsetek projektów informatycznych rezygnuje z wszelakiej dokumentacji i modeli UML, w momencie przekroczenia pewnego stopnia skomplikowania projektu. W myśl integracji narzędzi programistycznych oraz środowisk zauważalna jest częściowa możliwość zautomatyzowania większości nużących i pracochłonnych czynności, które są związane z propagacją zmian, co prowadzi do poprawienia jakości projektu. Aby rozwiązać powyższy problem integracji środowisk i narzędzi mamy możliwe do zastosowania dwie opcje. Pierwszą z opcji jest wyprodukowanie lub powiązanie ze sobą takiego zestawu narzędzi, który da możliwość zbudowania rozproszonego I ujednoliconego środowiska zarządzającego wszystkimi elementami danego projektu oraz zintegrowania ich ze sobą. Poprzez rozproszenie rozumiane są tutaj zarówno geograficzne rozproszenie wykonawców projektu jak I zarówno komponentów system. Natomiast drugą możliwością jest próba wytworzenia szkieletowej architektury, co dało by możliwość zestawienia istniejących narzędzi a także stworzenia z nich jednej spójnej platform, która służyła by to wytwarzania oprogramowania [6].

Podstawowe procesy wytwarzania oprogramowania

Jak już wspominaliśmy we wstępie proces wytwarzania oprogramowania bardzo silnie opiera się o model procesu wytwarzania. Do podstawowych modeli wytwarzania możemy zaliczyć m.in.:

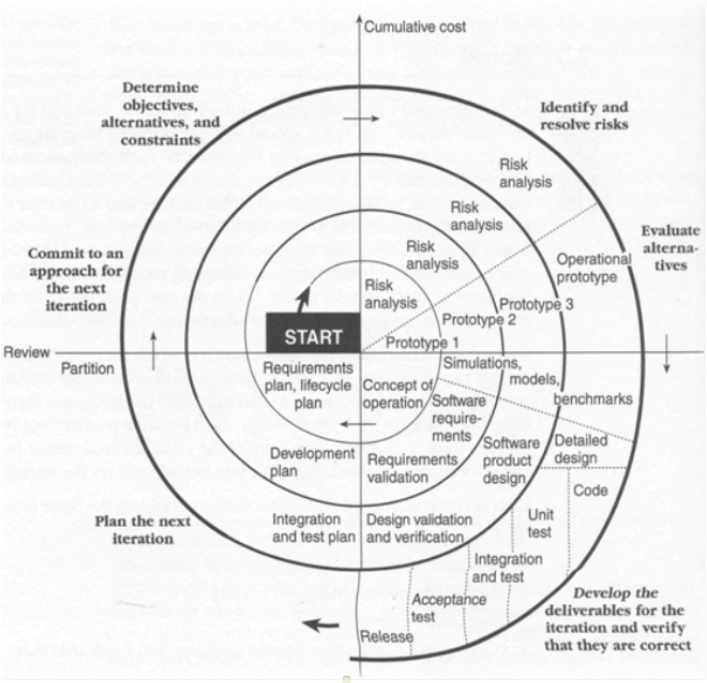
- Model kaskadowy,
- Prototypowanie,
- Model Spiralny,
- Model przyrostowy,
- Model przyrostowo-iteracyjny,
- Programowanie ekstremalne,
- Scrum (młyn) [7].
-

Model kaskadowy – model charakteryzuje sześć etapów realizacji. Pierwszym etapem jest proces planowania, określa się specyfikację wymagań projektu. Kolejny etap to analiza pod względem wykonalności projektu oraz analiza oferty. Trzecim etapem jest projektowanie działań oraz elementów projektu. Następnie następuje implementacja. Po zaimplementowaniu następuje etap testowania projektu, poszczególnych modułów oraz integracji. Końcowym

etapem jest wdrożenie i konserwacja. Wadami tego modelu może być brak elastyczności.

Prototypowanie – ten model głównie składa się z trzech etapów. Pierwszym etapem jest stworzenie prototypu. Następnie następuje szereg konsultacji z klientem, po czym ostatnim etapem jest realizacja projektu. Zaletą tego modelu jest mniejszy koszt wprowadzania jakichkolwiek zmian w prototypie. Natomiast główna wada to ogromny koszt realizacji systemu.

Model spiralny – model spiralny łączy w sobie cechy modelu kaskadowego oraz prototypowego. Kolejne prototypy tworzone są podobnie jak w modelu kaskadowym. Natomiast końcowym etapem jest stworzenie ostatecznej wersji produktu.

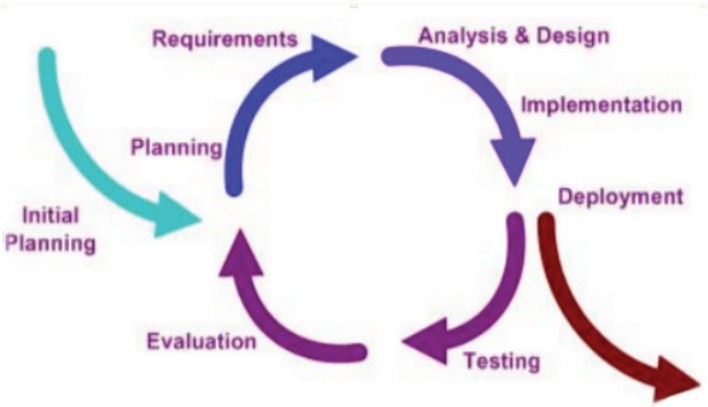


Ryc. 1. Model spiralny.

Model przyrostowy – model ten bazuje głównie na modelu kaskadowym. Proces wytwarzania oprogramowania jest realizowany w ten sposób, iż wraz z implementacją kolejnego etapu dodawana jest nowa funkcjonalność. Główne zalety tego modelu to łatwość dostosowywania modelu do zmian w specyfikacji oraz fakt, iż po zakończeniu każdego etapu mamy część zrealizowanego produktu.

Model przyrostowo-iteracyjny – model ten wykonywany jest w ramach kolejnych etapów modelu przyrostowego, jest to jego rozszerzenie. Model przyrostowo-iteracyjny jest jednym z najczęściej wykorzystywanych modeli, które służą do wytwarzania oprogramowania. Jeżeli chodzi o najpopularniejsze implementacje tego modelu:

- Rational Unified Process,
- Agile Software Development.



Ryc. 2. Model przyrostowo-iteracyjny.

Programowanie ekstremalne – model ten charakteryzuje głównie iteracyjność, co oznacza aktualizowanie produktu o nowe wersje co kilka tygodni. Występuję tutaj również brak dokumentacji oraz dopuszczalne są zmiany architektury projektu. Istotny w tym modelu jest stały kontakt z klientem. Testy jednostkowe tworzone są przed oprogramowaniem.

Scrum (młyn) – model scrum charakteryzuje się dobraniem samoorganizującego się oraz interdyscyplinarnego zespołu osób. Cykle, czyli sprinty trwają około 2-6 tygodni, efektem każdego jest działający produkt. Występują tutaj codzienne "poranne" kilkuminutowe spotkania teamu, w celu omówienia wczorajszych zadań, postępów i aktualnych celów [7].

Przykładowe środowiska, narzędzia wytwarzania oprogramowania

Jeżeli chodzi o narzędzia i środowiska programistyczne to możemy tu wyróżnić zintegrowane środowiska programistyczne(IDE). Są one wyposażone w szereg zaawansowanych menadżerów, które pokazują różne informacje o strukturze aplikacji oraz posiadają systemy wykrywania i usuwania błędów. W dzisiejszych czasach mamy do dyspozycji ich szeroki wachlarz. Zależnie od oczekiwanych usług i możliwości mamy do dyspozycji środowiska różnego przeznaczenia oraz służące do różnych celów. Czasami narzędzia programistyczne są tak wspierane oraz rozwijane, że z czasem stają się zintegrowanymi środowiskami programistycznymi. Do przykładowych środowisk programistycznych możemy zaliczyć m.in.:

- **Microsoft Visual Studio** – jest to zintegrowane środowisko programistyczne firmy Microsoft. Jest przeznaczone głównie do tworzenia oprogramowania konsolowego ale i również do tworzenia graficznych interfejsów użytkownika, w tym aplikacje WPF, Windows Forms, Web Aplications i innych. Produkty mogą być tworzone na platformy: Microsoft Windows, .NET Framework, XBOX, Microsoft Silverlight. Główne narzędzia tego środowiska to: Microsoft Visual C#, Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual J#, Mi-

crosoft Visual Web Developer ASP.NET oraz Microsoft Visual F# [8].

- **Eclipse** – jest to zintegrowane środowisko programistyczne stworzone do realizacji projektów typu rich-client. Głównym przeznaczeniem tego środowiska jest tworzenie aplikacji w języku programowania Java. Platforma została stworzona przez IBM w 2004r. Eclipse posiada szereg różnego rodzaju wtyczek, które umożliwiają głównie tworzenie produktów w takich językach jak: Java, C/C++, PHP a także tworzenie GUI, modelowanie aplikacji za pomocą UML i współpracę z bazami danych czy serwerami aplikacji [9].
- **.NET** – platforma programistyczna stworzona przez firmę Microsoft. To zintegrowane środowisko programistyczne nie jest związane z żadną konkretnie technologią, natomiast daje możliwość tworzenia produktów w wielu językach, np. C++/CLI, J#, C#, F#, Visual Basic czy Delphi. To środowisko daje możliwość tworzenia aplikacji działających po stronie serwera internetowego oraz aplikacji, które działają na różnych systemach. Główne narzędzia platformy .NET to przede wszystkim ASP.NET, które ułatwia tworzenie dynamicznych stron www oraz ADO.NET, które ułatwia dostęp oraz współpracę z bazami danych [10].

Natomiast typowe narzędzia, które wspierają pod względem upłynnienia i automatyzacji proces wytwarzania oprogramowania, mamy tutaj głównie na myśli system kontroli wersji. Wiele komercyjnych środowisk programistycznych jest zintegrowanych z **systemem kontroli wersji**. System kontroli wersji głównie służy do nadzorowania, zarządzania i integrowania wspólnego kodu. Przykłady takich narzędzi to m.in.:

- CVS
- Subversion
- GIT
- Bazaar itp [11].

Bardzo ważną grupą są narzędzia dające możliwość na usuwanie błędów z gotowej już aplikacji, produktu, czyli tzw. proces **debugowanie**. Możemy wyróżnić dwa różne sposoby debugowania:

- Statyczna analiza kodu,
- Dynamiczna analiza kodu.

Pierwszy sposób polega na analizie kodu pod względem możliwości wystąpienia błędów, natomiast drugi sposób polega na analizie kodu podczas wykonywania pracy produktu. Służą do tego odpowiednie narzędzia, zwane **debuggerami**. Istnieją również specjalne fragment kodu, które nastawione są na weryfikowanie kodu wyszukując w nim błędów. Niektóre języki mają wbudowane w swoje struktury debuggery i kod debugowany jest automatycznie, natomiast języki niskopoziomowe jak C czy C++ nie posiadają wbudowanych narzędzi debugujących [12].

Kolejna grupa narzędzi niezbędnych przy tworzeniu oprogramowania to **kompilatory**. Kompilator jest to narzędzie odpowiedzialne za przetłumaczenie(skompilowanie) języka programowania na język maszynowy, zrozumiały dla komputera. Niektóre kompilatory najpierw tłumaczą kod programistyczny na kod assemblera a ten kolejno na język maszynowy. Zaletą kompilatorów jest to, iż programista nie musi znać języka maszynowego oraz daje możliwość sporej przenośności danych pomiędzy platformami. Jako przykłady kompilatorów możemy wymienić m.in.:

- GNU Fortran 77[FORTRAN],
- GNU C[C],
- GNU C++[C++],
- Sun Java Compiler, GNU Java Compiler[JAVA] itp [13].

Istotnymi narzędziami jakie wprowadzają ogromną innowacyjność w proces wytwarzania oprogramowania są **biblioteki** oraz **frameworki** języków programowania. Biblioteki wraz z frameworkami są tworzone bardzo dynamicznie oraz często. Sam fakt o ilości bibliotek oraz frameworków biorąc pod uwagę wszystkie języki programowania jest nieograniczenie potężna z racji tego, iż są one wytwarzane niemal cały czas. Przykładowe biblioteki czy frameworki to:

- Symfony[PHP],
- React[JS],
- Angular[JS],
- Spring[JAVA],
- Entity[C#] itp.

Natomiast najważniejszy element przy wytwarzaniu oprogramowania to sam **język programowania**. To właśnie za pomocą języka programowania wytwarzane jest oprogramowanie. Języki programowania dzieli się potocznie na języki:

- Niskiego poziomu,
- Wysokiego poziomu.

Do języków niskiego poziomu zalicza się głównie:

- Język maszynowy,
- Assembler.

Jeżeli chodzi o języki wysokiego poziomu, to są to m.in. :

- Java,
- C#,
- C/C++,
- Python,
- Pascal,
- PHP itp. [14].

Kolejne narzędzie, o którym należy wspomnieć jest **Unified Modeling Language.** UML jest tak naprawdę zunifikowanym językiem modelowania. Przede wszystkim pozwala tworzyć różnego rodzaju modele(np. informatyczne). Pozwala obrazować, tworzyć, specyfikować oraz dokumentować element tworzonego system. UML ułatwia także wykorzystanie zalet, które występują wraz z programowaniem obiektowym oraz jest przeznaczony głównie do używania w procesie analizy i projektowania systemów komputerowych. UML jest złożony z dwóch elementarnych składowych:

- Notacja,
- Metamodel [15].

Notacja	Metamodel
• element graficzne • składnia języka modelowania • istota przy szkicowaniu	• definicje pojęć języka i powiązania między nimi • istotny przy graficznym programowaniu

Tab. 1. Składowe UML.

Unified Modeling Language składa się na dwie powyższe definicje. Pierwsza czyli notacja wyznaczonych elementów, które są używane na diagramach, natomiast z drugiej strony ich semantykę, czyli metamodel. Najważniejsze jest to aby model był jednoznacznie i czytelnie opisany, tak aby inne osoby bez problem potrafiły go zrozumieć. Dlatego kluczowa jest tutaj notacja, zaś metamodel winien być zrozumiały intuicyjnie. Przy generowaniu kodu oraz przejściu do implementacji tego kodu, najważniejsze jest ścisłe zrozumienie przeznaczenia określonych elementów, aby możliwa była automatyczna konwersja modelu do innego formalizmu.
Zakres oraz przeznaczenie zastosowania UML zawiera się głównie w:

- Tworzeniu systemów informacyjnych przedsiębiorstw,
- Usługach bankowych I finansowych,
- Transporcie,
- Telekomunikacji,
- Sprzedaży detalicznej,
- Przemyśle obronnym i lotniczym,
- Nauce,
- Elektronice i medycynie,
- Rozproszonych usługach internetowych [15].

UML tworząc określony model bazuje na odpowiednich diagramach, rodzaje tych diagramów to m.in. [15]:

Diagramy strukturalne	Diagramy dynamiki
• Diagram klas • Diagram obiektów • Diagram pakietów • Diagram komponentów • Diagram wdrożenia • Diagram struktur złożonych	• Diagram przypadków użycia • Diagram stanów • Diagram przebiegu • Diagram czynności • Diagram kooperacji • Diagram przeglądu interakcji • Diagram uwarunkowań czasowych

Tab. 2. Rodzaje diagramów UML.

Narzędzia **CASE** są również istotne w procesie tworzenia oprogramowania. Computer Aided Software Engineering jest to oprogramowanie używane do komputerowego wspomagania projektowania oprogramowania. Główne funkcje CASE to: analiza, projektowanie oraz programowanie. Narzędzia CASE automatyzują metody dokumentacji, projektowania oraz tworzenia struktur kodu program w określonym języku programowania obiektowego. Głównymi narzędziami CASE są:

- Narzędzia do modelowania w UML oraz podobnych,
- Narzędzia do zarządzania systemem kontroli wersji,
- Narzędzia do refactoringu, czyli do wprowadzania zmian w danym projekcie [16].

Wnioski

Już dzisiaj widzimy w jak szybkim tempie postępuje rozwój technologiczny oraz jakie ogromne możliwości daje nam rozwój komputerów i internetu. Postęp ten spowodował, iż granice jakimi były czas oraz odległość wraz z wpływem globalizacji i nowych technologii zamazały się. Bez rozwoju stopnia zaawansowania szeroko pojętego oprogramowania na dzień dzisiejszy rozwój na tak wysokim poziomie nie był by możliwy. Natomiast tempo rozwoju technologicznego ciągle wzrasta, co za tym idzie dostępność oraz ilość narzędzi i środowisk wysoko zaawansowanych także rośnie, co pozytywnie wpływa na automatyzacje i innowacyjność procesu wytwarzania oprogramowania ale nie tylko. Wzrost poziomu zaawansowania w procesie wytwarzania oprogramowania znacząco wpływa na wzrost komfortu życia społeczeństwa oraz jego funkcjonowania. Możemy śmiało pokusić się o tezę, iż urządzenia elektroniczne, co za tym idzie oprogramowanie jest na dzień dzisiejszy wszechobecne i odgrywa ogromne znaczenie na funkcjonowanie w życiu codziennym. Jeżeli chodzi o kierunek rozwoju tworzenia oprogramowań, to rozwój zmierza ku stworzeniu sztucznej inteligencji, co w najbliższej przyszłości będzie możliwe. Już możemy zauważyć prototypy autonomicznych maszyn, robotów oraz urządzeń wspomagających w życiu codziennym społeczeństwo.

Literatura

[1] http://web.archive.org/web/20010530092843/http://home.olemiss.edu/~misbook/sfsysfm.htm, dostęp: 15.04.2018r.
[2] http://edu.pjwstk.edu.pl/wyklady/pri/scb/index05.html, dostęp: 15.04.2018r.
[3] Kernighan, Brian W.; Plauger, P. J. (1976), Software Tools, Addison-Wesley, p. 352.
[4] https://pl.wikibooks.org/wiki/C/Zintegrowane_%C5%9Brodowisko_programistyczne, dostęp: 15.04.2018r.
[5] Mordechai Ben-Ari: Understanding Programming Languages. Chichester: John Wiley & Sons, 1996r.
[6] C. Boldyreff i R. Dewar, Environments to Support Collaborative Software Engineering., 2003, 2nd Workshop on Cooperative Supports for Distributed Software Engineering Processes, Benevento.
[7] Górski J. „Inżynieria Oprogramowania w projekcie informatycznym", Mikom, s. 74-84.
[8] https://docs.microsoft.com/pl-pl/visualstudio/releasenotes/vs2017-relnotes, dostęp: 15.04.2018r.

[9] http://help.eclipse.org/oxygen/index.jsp, dostęp: 15.04.2018r.
[10] Microsoft makes .Net open-source, finally embraces iOS, Android, and Linux - ExtremeTech, www.extremetech.com, dostęp: 15.04.2018r.
[11] https://docs.microsoft.com/pl-pl/vsts/tfvc/overview?view=vsts, dostęp: 15.04.2018r.
[12] Andreas Zeller: Why Programs Fail: A Guide to Systematic Debugging. Morgan Kaufmann, 2005.
[13] A.V. Aho, R. Sethi, J.D. Ullman, Kompilatory. Reguły, metody i narzędzia, WNT, Warszawa 2002.
[14] Bruce J. MacLennan: Principles of Programming Languages. Oxford University Press, 1987,s. 132.
[15] Tom Pender: UML Bible. John Wiley & Sons, 2003.
[16] A. Jaszkiewicz, Inżynieria oprogramowania, Helion, Gliwice, 1997, s. 112.
[17] P.Kardasz, Programowanie w języku Python, Wrocław 2017