ZESZYTY NAUKOWE WSEI
SERIA:
TRANSPORT I INFORMATYKA

5(1/2015)

Zeszyty naukowe
Wyższej Szkoły ekonomii i Innowacji w Lublinie
SERIA: TRANSPORT I INFORMATYKA

Printed in Poland
Innovatio Press Wydawnictwo Naukowe
Wyższej Szkoły Ekonomii i Innowacji
20-209 Lublin, ul. Projektowa 4
tel.: + 48 81 749 17 77, fax: + 48 81 749 32 13
www.wsei.lublin.pl

ISSN: 2084-8005

Spis treści

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 5–9

Jan Wrona

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

Bezpieczeństwo uczestników ruchu drogowego chorych na kataraktę

Safety of road traffic participants syffering cataract

Streszczenie

W artykule podniesiony jest problem widzenia przeszkód przez kierujących z kataraktą. Autobiograficzny przypadek analizuje trudności z rozpoznawaniem przeszkód, infrastruktury drogowej i innych uczestników ruchu. Zasygnalizowano istotny wpływ osób z kataraktą na bezpieczeństwo drogowe. Przytoczono też przykłady dwóch zdarzeń drogowych, które mogły zaistnieć wskutek wady wzroku u kierujących pojazdami.

Summary

The article raises the problem of how drivers suffering from cataract see the roadblocks. The autobiographic case analyzes difficulties with recognizing obstacles, road infrastructure and other participants of the traffic. A significant influence of persons suffering from cataract upon road safety was signaled. Examples were also quoted of two road accidents that might have taken place due to drivers' vision defects.

Słowa kluczowe: oko ludzkie, katarakta, bezpieczeństwo drogowe

Keywords: human eye, cataract, road safety

1. Wstęp

Do napisania artykułu skłoniły mnie własne niezbyt odległe doświadczenia, jako że będąc kierującym pojazdem samochodowym miałem rozpoznaną u siebie zaćmę. Dodaję, że jestem w wieku emerytalnym i nadal pracującym na pełnym etacie, co uzasadnia moje czynne korzystanie z pojazdu. Nadto jako interesujący się bezpieczeństwem drogowym spostrzegłem, że brak jest jakichkolwiek regulacji prawnych, czy medycznych, które uwzględniałyby fakt, że kierujący z chorobą zaćmy (katarakty), zazwyczaj nabywaną w starszym wieku, ma ograniczoną możliwość rozpoznawania wszystkiego co znajduje się na drodze. Jest to tym bardziej niebezpieczne, że na operację czeka się dość długo, a czas pomiędzy zabiegiem jednego oka a zabiegiem oka drugiego wymaga także długiego czasu. W tym czasie będąc osobą zdefektowaną z uwagi na wzrok nadal kierowałem samochodami. W powyższej sytuacji należałoby zastanowić się, czy osoby po tzw. „sześćdziesiątce", którzy najczęściej chorują na zaćmę (około 90 % chorych) nie powinni być zakwalifikowani do grupy osób o niepełnej zdolności do kierowania pojazdami. W dalszej części artykułu podzielę się swoimi sytuacjami, które wynikały z nabytej przeze mnie zaćmy. Mam cichą nadzieję, że pismem tym spowoduję większe zainteresowanie u wszystkich, którym na sercu leży bezpieczeństwo własne i innych.

2. Oko jako narząd ludzki zaatakowane przez kataraktę

Jednym ze składników narządu wzroku jest oko ludzkie, w skład którego wchodzi między innymi soczewka. Wraz ze starzeniem się organizmu postępuje zmętnienie soczewki. Wówczas pogarsza się ostrość widzenia, a chorzy opisują to zaburzenie jako wrażenie zamglenia widzenia. Dochodzi do stopniowego zacierania widzianego obrazu. Powyższe zjawisko zwane zaćmą (kataraktą) dotyka wielu osób w wieku po 60 roku życia. Stopień nasilenia choroby może być różny. W tym miejscu należy zauważyć, że zaćmie nie można zapobiec. W mojej sytuacji najpierw zmieniałem okulary na coraz „mocniejsze", a ostatnie okulary, które zakładałem tylko do czytania miały trzy dioptrie. W pewnym okresie zauważyłem, że w niektórych sytuacjach jak na przykład podczas intensywnego słońca miałem trudności z ostrością widzenia. Byłem jak gdyby dotknięty zjawiskiem olśnienia. Natomiast widzialność poprawiała mi się gdy było pochmurno lub zapadał zmierzch. Zacząłem nosić ciemne okulary, które jednak z biegiem czasu niewiele mi poprawiały widoczność. Udałem się do okulisty, który rozpoznał u mnie zaćmę i skierował mnie do kliniki okulistycznej, gdzie stanąłem w kolejce do zabiegu z perspektywą oddaloną o około 18 miesięcy. Wynikało to dużej liczby chętnych i ograniczonych możliwości finansowych NFZ. W tym czasie musiałem radzić sobie w życiu oraz jako kierujący samochodem. Podczas oczekiwania na

operację oka ostrość mojego wzroku malała. Bywały sytuacje drogowe na przykład jazdy w nocy i padającego deszczu, gdy zasięg widoczności drogi zmniejszał się przed moim samochodem do kilku metrów, zwłaszcza podczas wymijania się pojazdów. Bywały także sytuacje, że miałem trudności z rozpoznawaniem świecenia się i jakich świateł sterujących ruchem ulicznym. Bardzo często stosowałem technikę jazdy "za zderzakiem poprzedzającego mnie samochodu. Zdawałem sobie sprawę, że dalsze oczekiwanie na zabieg mojego oka może skończyć się bardzo źle. Utrudniony miałem także wjazd do wąskiego i nie najlepiej oświetlonego wielomiejscowego garażu. Prosiłem więc o przyspieszenie operacji mojego oka w tym także Ministra Zdrowia RP. Ostatecznie zostałem z niewielkim przyspieszeniem, wcześniej uzgodnionego terminu zoperowany na lewe oko, które było w znacznie gorszym stanie niż prawe. Rekonwalescencja po zabiegu trwała krótko, ale ponad miesiąc musiałem zapuszczać do oka kropelki. Następnie ponownie zgłosiłem się w celu zoperowania mojego prawego oka. Długi termin oczekiwania na zabieg spowodował, że musiałem sobie radzić za kierownicą jako jednooki uczestnik ruchu. Należy dodać, że dość dobrze widziałem okiem zoperowanym, bez potrzeby posługiwania się okularami. Niemniej musiałem nauczyć się rozpoznawania ludzi i uczestników ruchu jednym okiem, do czego wcześniej nie byłem przyzwyczajony. W tym miejscu pragnę zauważyć, że wstawiona syntetyczna soczewka mojego oka, nie ma takich zdolności jak soczewka naturalna. Uchybienie to czyniło, że nie rozpoznawałem właściwie odstępów bocznych od mojego pojazdu, co powodowało, że parkowałem w sposób nieprawidłowy, bez zachowania bezpiecznego odstępu bocznego od innych pojazdów. Powyższe uchybienie mojego wzroku odnosiło się także do zachowania właściwych odstępów podczas omijania przeszkód oraz podczas wyprzedzania czy wymijania się z innymi samochodami. Jako przykład mogę przytoczyć, że wjeżdżając do garażu zdarzało mi się zaczepić lusterkiem o bramę lub nie zachować odległości od ściany garażu. Powyższe niedostatki wzroku naraziły mnie na uszkodzenia własnego samochodu. Implantowana soczewka o stałej ogniskowej powoduje, że jest różna ostrość widzenia obrazów w zależności od odległości, co uzasadnia korzystanie z odpowiednich okularów. Zarówno zdolność widzenia jak i rozpoznawanie innych uczestników ruchu jak i przeszkód ustało z chwilą zoperowania drugiego oka. Mając na uwadze powyższe należałoby zastanowić się czy ja posiadając zaćmę, oczekując na zabiegi oraz będąc w trakcie poszczególnych zabiegów mogłem kierować samochodami. Z formalnego punktu widzenia aktualny stan prawny i medyczny nie stoi na przeszkodzie w kierowaniu pojazdami [1,2,4]. Jest jednak problem innego zagadnienia wynikający z mojej praktyki jako biegłego do spraw ruchu drogowego. Opracowując opinię o przyczynie wypadku analizuję zachowanie się kierujących między innymi w wieku po „sześćdziesiątce". Nikt z dotychczasowych zleceniodawców nie uwzględniał możliwości oceny zachowania się kierującego z zaćmą lub będącego po zabiegu zaćmy. Nadto organ dochodzeniowy nie zleca badania

wzroku kierującego, który był uczestnikiem zdarzenia ze skutkiem śmiertelnym. Z mojego doświadczenia, aczkolwiek nie miałem własnych zdarzeń drogowych, zasadne byłoby branie pod uwagę, wskazanego przeze mnie problemu widzenia przeszkód przez kierujących z chorobą zaćmy. Jestem świadomy problemów jakie niosłoby uwzględnienie mojej sugestii, ale mając na uwadze, że w Polsce na jeden milion mieszkańców ginie rocznie na drogach 85 osób, a za nami w tym rankingu są tylko nieliczne kraje, należałoby i ten problem wziąć pod uwagę w trosce o poprawę bezpieczeństwa drogowego. W uzupełnieniu dodam, że zasygnalizowany problem dotyczy także pieszych i rowerzystów, osób starszych uczestniczących w wypadkach drogowych, którzy stanowią znaczący odsetek osób śmiertelnych na drogach.

Aby wesprzeć podniesione przeze mnie kwestie pragnę przytoczyć dwa przykłady z mojej praktyki jako biegłego.

Pierwszy dotyczy wypadku jaki zaistniał w Lublinie w słoneczny dzień, który polegał na czołowym uderzeniu motocyklisty, w prawy bok skręcającego przed nim w lewo samochodu osobowego marki VW. Zdarzenie to ilustruje fotografia zamieszczona niżej.

Rys. 1. Powypadkowe usytuowanie pojazdów

Skutkiem zderzenia były nie przeżyciowe obrażenia motocyklisty. Kierującym samochodem był blisko 78 letni mężczyzna, który jak wyjaśnił nie widział nadjeżdżającego motocyklisty, w chwili rozpoczęcia skrętu w lewo. Klasyczne z punktu widzenia prawa drogowego, nie ustąpienie pierwszeństwa było przyczyną analizowanego zdarzenia. Należałoby jednak zastanowić się czy realne było, aby kierujący samochodem, ze sprawnym wzrokiem nie zauważył zbliżającego się motocyklisty z włączonym światłem z odległości około 50 m. Nikt jednak nie podniósł

kwestii ewentualnej wady wzroku kierującego samochodem osobowym, zarówno w procesie dochodzeniowym jak i podczas rozpraw sądowych.

Drugi przykład zaistniał w Janowie Lubelskim, także w słoneczny dzień i polegał na zaczepieniu prawej rękojeści kierownicy roweru i upadku na jezdnię kierującego rowerem, którym poruszał się 73 letni rowerzysta, o otwarte drzwi zaparkowanego samochodu osobowego stojącego przy prawej krawędzi jezdni. Bezkolizyjne ominięcie zaparkowanego samochodu, obiektywnie rozpoznawalnego było w analizowanej sytuacji możliwe, gdyby rowerzysta zachował bezpieczny odstęp boczny od stojącego za jezdni samochodu z otwartymi drzwiami. Jak wynikało z badań skrzydło drzwi wystawało poza obrys zewnętrzny samochodu o około 0,75 m. Nie wiadomo dlaczego rowerzysta (zmarł po przewiezieniu do szpitala), omijał stojący pojazd nie zachowując bezpiecznego odstępu bocznego. Nikt w procesie dochodzeniowym i sądowym nie rozpatrywał kwestii wzroku poszkodowanego, ani jego zdolności widzenia w blasku majowego słońca.

Powyżej przytoczone zdarzenia powinny sugerować potrzebę oceny wzroku i ostrości widzenia, w celu wyjaśnienia przyczyny zaistniałych wypadków.

Bibliografia

[1] Wicher J.: Bezpieczeństwo pojazdów i ruchu drogowego. Wydawnictwo WKiŁ, Warszawa 2004 r.

[2] Problematyka prawna i techniczna wypadków drogowych Materiały szkoleniowe. Wydawnictwo IES, Kraków 1998 r.

[3] Wrona J. Wrona R.: Wyprzedzanie jako szczególnie niebezpieczny manewr drogowy. Autobusy nr 7–8/2012, s. 182–186.

[4] Prawo o ruchu drogowym Dz.U. z 2012 r z późniejszymi zmianami.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 11–18

Sofiia Zakharchuk

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

Tworzenie e-booków za pomocą języka HTML

E-book design and production using HTML language

Streszczenie

Omówiono krótko niektóre standardy i formaty plików e-booków, koncentrując się na formacie MOBI. Przygotowania pliku w tym formacie można dokonać za pomocą języka HTML. Zaprezentowano aplikację Adobe Dreamweaver przeznaczoną w zasadzie do projektowania stron i portali internetowych. Pokazano kolejne kroki tworzenia pliku e-booka za pomocą tej aplikacji oraz wybrane zaawansowane zagadnienia. Przykład e-booka dla smartfonu zamyka referat.

Słowa kluczowe: e-book, aplikacje mobilne, język HTML

Summary

Selected standard and formats of e-book files have been presented. Special attention has been devoted to MOBI format. Such a file can be designed and produced using HTML language. Adobe Dreamweaver application, used usually for Internet pages and portals design, has been presented. Subsequent stages of e-book design using this application have been shown, along with more advanced topics. An example of e-book for smartphon platform closes the paper.

Keywords: e-book, mobile applications design, HTML Language

1. Wstęp

Co to jest e-book ?

E-book jest to publikacja elektroniczną w formacie EPUB lub MOBI/KF8Sofiia ZAKHARCHUK

Tworzenie e-booków za pomocą języka HTML przeznaczoną do odczytu na urządzeniach mobilnych, takich jak czytniki e-booków, tablety i smartfony.

Co to jest EPUB?

Format EPUB jest to otwarty standard publikacji elektronicznej obsługiwany przez większość urządzeń mobilnych z wyjątkiem Kindle. Charakteryzuje się tym, że nie posiada podziału na strony a parametry tekstu mogą być zmieniane przez czytelnika.

Tworzenie e-booków za pomocą języka HTML EPUB jest również podstawą przy tworzeniu formatu MOBI/KF 8 dla czytnika Kindle.

Od strony technicznej, plik EPUB to po prostu archiwum zip, o rozszerzeniu epub, Sofiia ZAKHARCHUK.

Tworzenie e-booków za pomocą języka HTML zawierające pliki HTML, NCX oraz OPF. Więcej na ten temat we wpisie na temat struktury pliku EPUB.

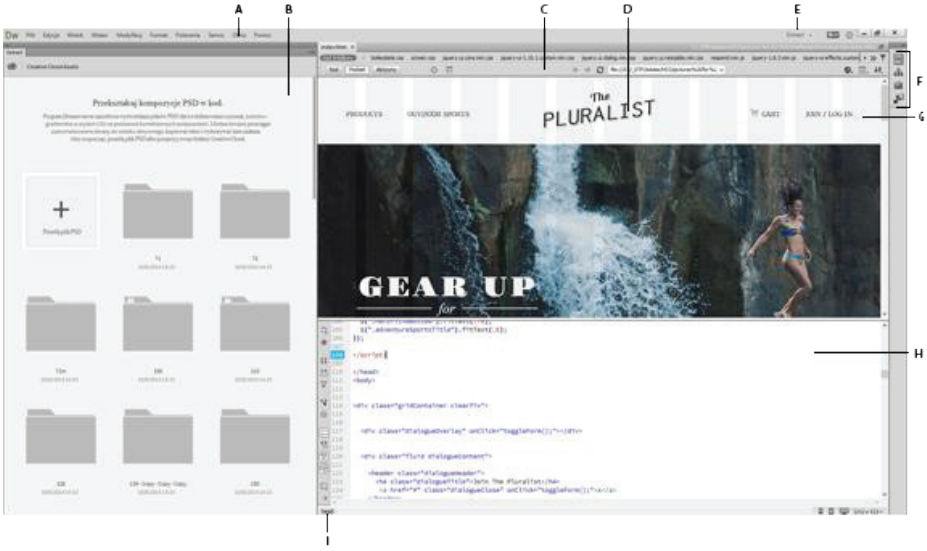
Od września 2007 roku jest to oficjalny standard IDPF – instytucji zajmującej się tworzeniem standardów dla branży publikacji elektronicznych. W jej skład wchodzą między innymi takie firmy jak Adobe, Smashwords, E-Ink, Google czy Barnes&Noble.

Czym jest format MOBI i KF8?

MOBI (znany również pod nazwą MOBI 7) jest to format publikacji elektronicznej rozwinięty przez francuską firmę Mobipocket kupioną przez Amazon w 2005 roku. Ten format wykorzystywany jest przez niektóre starsze czytniki Amazonu (Kindle 2 i wcześniejsze). Posiada on bardzo ograniczone możliwości formatowania tekstu.

W 2012 roku Amazon wprowadził format KF8 oparty na HTML5, który jest następcą MOBI i wprowadza wiele usprawnień dotyczących formatowania tekstu oraz umożliwia korzystanie z własnych czcionek.

Dokładniejsze porównanie zostanie wkrótce opublikowane w formie dłuższego wpisu.



Rys. 1. Oznaczenia: A.Pasek aplikacji B. Panel Extract C. Pasek narzędziowy Dokument D. Okno Dokument E. Przełącznik przestrzeni roboczej F. Grupy paneli G. Widok aktywny H. Widok Kod I. Selektor znaczników

2. Program Adobe Dreamweaver

Adobe Dreamweaver to program komputerowy firmy Adobe (dawniej Macromedia) przeznaczony dla projektowania stron internetowych i materiałów dla urządzeń przenośnych.

Program posiada rozbudowane opcje tworzenia serwisów internetowych; umożliwia edycję dokumentów w trybie WYSIWYG oraz trybie widoku źródła z opcją kolorowania składni, autouzupełnieniami kodu. Edytor posiada dużą liczbę gotowych akcji ułatwiających tworzenie stron takich jak generowanie przycisków w formacie „Adobe Flash" oraz możliwość obsługi zewnętrznych pluginów.

Rys. 2. Okno startowe programu Adobe Dreamweaver CS6

3. Układ przestrzeni roboczej

Przestrzeń robocza programu Dreamweaver umożliwia wyświetlanie właściwości dokumentów i obiektów. W przestrzeni roboczej znajdują są także wiele najczęściej używane opcje, umieszczone na paskach narzędziowych, co umożliwia szybkie stosowanie zmian w dokumentach.

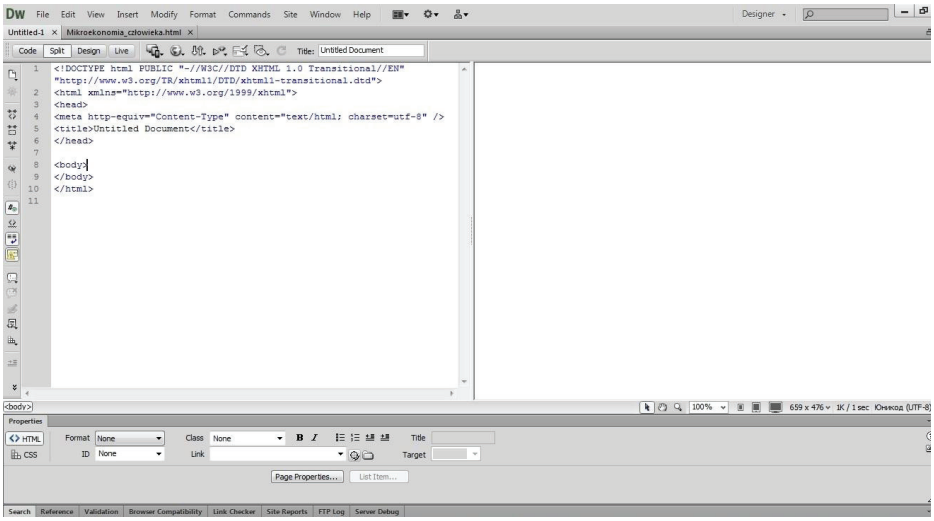
4. Elementy przestrzeni roboczej

Przestrzeń robocza składa się z następujących elementów:

Ekran Witamy, Pasek aplikacji, Pasek narzędziowy Dokument, Pasek narzędziowy Standardowy, Pasek narzędziowy Kodowanie, Okno Dokument, Inspektor właściwości, Selektor znaczników, Panel Extract, Panel Wstaw, Panel Pliki.

Uwaga:

Program Dreamweaver zawiera też wiele innych paneli, inspektorów i okien. Panele, inspektory i okna otwiera się za pomocą menu Okno.



Rys. 3. Przestrzeń robocza

5. Proces tworzenia e-booka

Krok 1. Stworzyć podfolder w folderze programu w którym będą znajdowały się wszystkie pliki dotyczące tworzonego e-booku.

Krok 2. Otworzyć program.
File -> New -> HTML -> Create

Krok 3. Zadajemy rozmiar, styl tekstu i nagłówków.

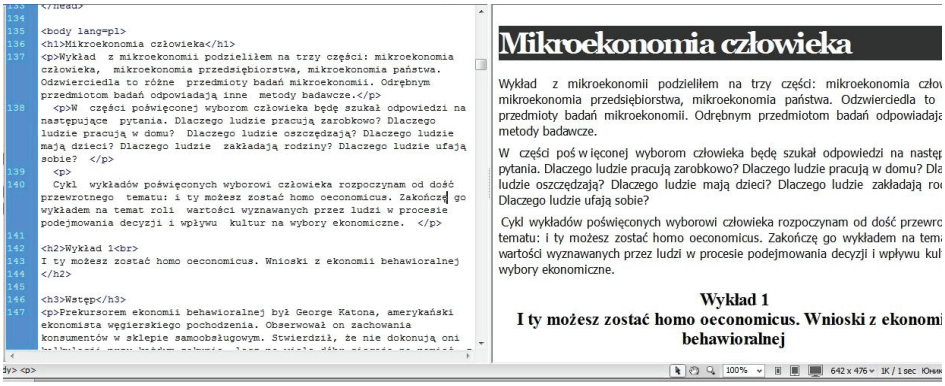
```
h1
{
        font-size: 2.5em; – rozmiar tekstu
        color: #FFF; – kolor
        width:auto; – szerokość
        height:auto; – wysokość
        background-color:#333; – kolor tła
        text-align:left; – wyrównanie tekstu
        font-family: Georgia, serif; – rodzaj czcionki
```

```
        margin-top:0.5em; – marginesy
        margin-bottom:0.5em;}
p
  {font-size:1em;
   color: #000;
   font-family: Tahoma, Geneva, sans-serif;
   text-align:justify;
   line-height:1.3em;
        margin-top:0.5em;
        margin-bottom:0.5em;
        }
.bio {
        font-size:1em;
        margin-top:3em;
        margin-bottom:1em;}
```

Krok 4. Zaczynamy edycję tekstu.

Kopiujemy tekst do lewej części ekranu w sekcję <body>.

W zależności od stylu zadajemy tekstu znacznik <h1> (nagłówek największy), <h2>.. ,<p> (paragraf).



Rys. 4. Edycja tekstu z użyciem tagów <h1>, <h2> ,<p>

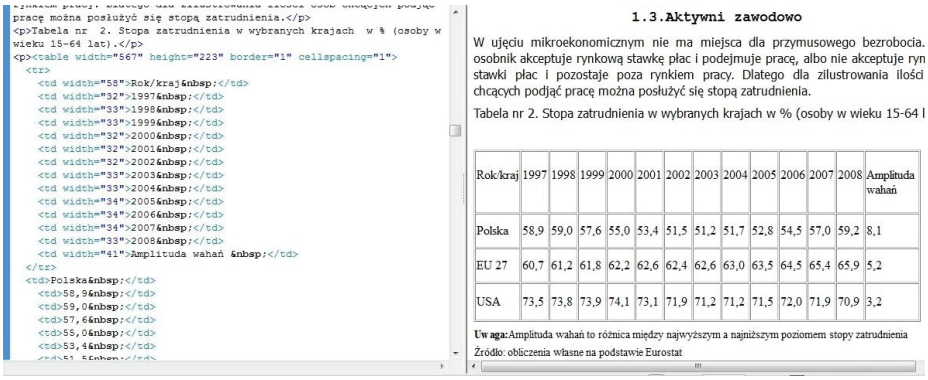
Tabele można wstawić

1) używając „menu”

Insert -> Table

Pojawi się okno w którym wybierzemy ilość kolumn i wierszy.

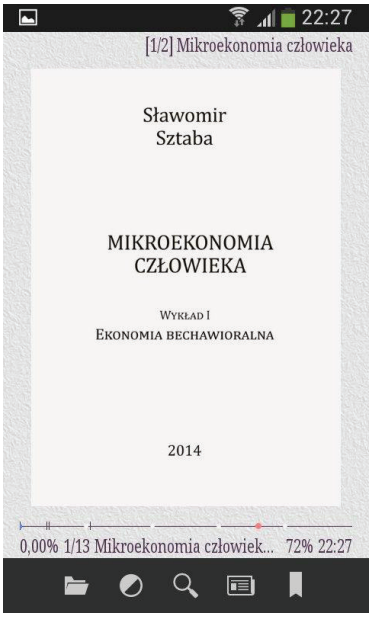
2) Pisząc za pomocą kodu, ale ta metoda wymaga dużo więcej czasu.



Rys. 5. Tabela w Adobe Dreamweaver

Krok 6. Cały sformatowany tekst zapisujemy w formacie html.
File -> Save as..

Krok 7. Pobieramy program Calibre E-book management w którym zapisujemy plik .html w formatach epub,mobi.



Rys. 6. Zrzuty ekranu smartfona

6. Tworzenie e-booków z książek skanowanych

Taka opcja jest dostępna w przypadku używania programów, które czytają tekst z obrazów skanowanych, na przykład Abbyy FineReader.

Proces jest taki sam, tylko pierwszym krokiem będzie rozpoznawanie tekstu i zapisanie go w dokumencie tekstowym.

7. Podsumowanie

Świat staje się coraz bardziej mobilny. Telefony stały się smartfonami lub tabletami i działają jak miniaturowe wersje domowych komputerów. Coraz więcej codziennych zajęć – to widać głównie w Ameryce – wykonuje się na urządzeniach mobilnych. Czytanie książek jest jedną z takich czynności i coraz więcej wydawców tworzy lub konwertuje swoje publikacje do formatów zrozumiałych przez urządzenia mobilne.

E-book to jest nowoczesny sposób na czytanie, nabycie wiedzy, gdziekolwiek jesteś i kiedykolwiek chcesz, bo zawsze masz pod ręką urządzenie przenośne które umożliwia czytanie e-booka. Wiedza dotycząca tworzenia e-booków pozwala Ci mieć ulubione książki zawsze ze sobą nawet jak nie ma jej w internecie.

Bibliografia

[1] http://blog.psboy.pl/2012/10/tworzenie-ebookow-epub-w-indesignie--cs5cs5-2.

[2] http://blog.khron.net/2012/05/11/ebook_spis_tresci_rozdzialy_kindle/

[3] http://magazynt3.pl/jak-zrobic-ebooka/

[4] http://moonopol.pl/zrob-sobie-ksiazke/

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 19–27

Michał Maj

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

Porównanie metod obliczeń równoległych OpenMP i CUDA

Comparison of OpenMP and CUDA parallel computations

Streszczenie

Programowanie równoległe oznacza tworzenie programów w taki sposób, by można je było wykonywać równocześnie na wielu procesorach. Na potrzeby niniejszego artykułu napisane zostały dwa programy zrównoleglone – jeden w CUDA C oraz jeden w OpenMP, przeznaczony dla CPU – oraz jeden sekwencyjny (niewspółbieżny). Najszybszym sposobem zrównoleglania okazał się program napisany w CUDA, w którym wykorzystuje się pamięć niekopiowaną. Wadą CUDA jest to, że działa tylko ze sprzętem firmy NVIDIA.

Słowa kluczowe: obliczenia równoległe, OpenMP, CUDA

Summary

Parallel programming means development of programs, which can be executed truly concurrently on multiprocessor platforms. For current test purposes two parallel programs have been developed – one in CUDA C language, second using OpenMP library. Also equivalent sequential (non-parallel) program has been developed. Most efficient parallelization have been achieved in CUDA program with page-locked memory. CUDA is handicapped by limitation to NVIDIA hardware.

Keywords: parallel computations, OpenMP, Compute Unified Device Architecture (CUDA)

1. Programowanie równoległe

Współczesne procesory posiadają wiele rdzeni, dzięki temu mogą wykonywać osobne wątki w jednym cyklu zegara.

Programowanie równoległe oznacza tworzenie programów w taki sposób, by można je było wykonywać równocześnie na wielu procesorach. Jest to możliwe dzięki modyfikacji kodu, który był przeznaczony na jeden procesor – następuje zrównoleglenie kodu.

Zrównoleglenie polega na wyszukiwaniu fragmentów kodu, które mogą wykonywać działania niezależnie od siebie.

2. Programowanie na procesor graficzny

Początkowo grafikę tworzono przy użyciu bibliotek OpenGL i DirectX. Później postanowiono wykorzystać moc obliczeniową GPU do innych celów, niezwiązanych z tworzeniem grafiki. Jednakże, przy ówczesnej technologii, można było komunikować się z GPU tylko przy użyciu bibliotek graficznych. W związku z tym nie był ważny rodzaj wykonywanych obliczeń – konieczne okazało się postępowanie zgodne z zasadami programowania grafiki. Specjaliści musieli programować obliczenia matematyczne przy użyciu tych samych technik, które wykorzystywali przy tworzeniu grafiki. Było to związane z architekturą GPU, w którym Shader Pixele obliczają wartość koloru na podstawie położenia piksela na ekranie. Zawierają też w sobie inne informacje, przykładowo kolor, współrzędne x, y, współrzędne tekturowe. Programiści zamieniali jedną z pobocznych wartości Shader Pixela na własne dane, potrzebne do innych niż graficzne wyliczeń.

Niedogodności z programowaniem na Graphics Processing Unit (GPU) trwały do 2006 roku, kiedy NVIDIA wydała kartę graficzną GeForce 8800gtx z obsługą DirectX 10. Procesor graficzny był oparty na technologii Compute Unified Device Architecture (CUDA). Stosuje się ją do zwiększenia wydajności obliczeń dzięki wykorzystaniu mocy układów GPU. W porównaniu do wcześniejszych kart graficznych, GPU posiadał wspólny potok przetwarzania, a nie dzielił się, jak dotychczas, na shadery wierzchołków i shadery pikseli. W serii kart G80 jednostki arytmetyczno-logiczne zostały zaprojektowane zgodnie z normą Institute of Electrical and Electronics Engineers (IEEE) dla arytmetyki liczb zmiennoprzecinkowych oraz zaimplementowano w nich instrukcje pozwalające im na przetwarzanie nie tylko grafiki, ale również do dokonywania obliczeń ogólnych. GPU otrzymał dostęp do pamięci, co umożliwiło swobodny odczyt i zapis danych. Oprócz tego jednostki wykonawcze miały dostęp do pamięci podręcznej (pamięć wspólna). CUDA pozwalała karcie graficznej na realizowanie obliczeń matematycznych związanych nie tylko z grafiką.

Ze względu na fakt, że do zaprogramowania GPU wciąż potrzebny był OpenGL lub DirectX oraz należało używać specjalistycznych języków do przetwarzania grafiki – odpowiednio GLSL oraz HLSL – NVIDIA wydała rozszerzenie do języka C pozwalające na korzystanie z architektury CUDA. Język ten nosi nazwę CUDA C. Był to pierwszy język ogólnych obliczeń dla karty graficznej.

3. Porównanie programu sekwencyjnego, OpenMP oraz CUDA

CUDA ma opinię najwydajniejszej architektury służącej do równoległego przetwarzania danych. Aby to sprawdzić, na potrzeby niniejszego artykułu napisane zostały dwa programy zrównoleglone – jeden w CUDA C oraz jeden w OpenMP, przeznaczony dla CPU – oraz jeden sekwencyjny (niewspółbieżny). Powstały do mierzenia czasu potrzebnego do wykonania obliczeń przez obie technologie. Służyły wykazaniu różnic czasowych między nimi, a także programem sekwencyjnym. Platformą testową był komputer osobisty (PC), posiadający specyfikację przedstawioną w tabeli 1. Natomiast w tabeli 2 przedstawiona została specyfikacja karty graficznej.

Tab. 1. Specyfikacja komputera osobistego (PC)

CPU	
Name	Intel® Core ™ i5-4690K CPU @ 3,5 GHz
Architecture	X64
Frequency	3 499 MHz
Number of Cores	4
Page Size	4 096
Total Physical Memory	8 111,00 MB
Available Physical Memory	3 334,00 MB
Operating System	
Version Name	Windows 8.1 Pro
Version Number	6.2.9200
Tools	
Nsight Version	4.5.0.15036
Visual Studio Version	11.0

Tab. 2. Specyfikacja karty graficznej

Frame Buffer Physical Size	4096 MB
Frame Buffer Brandwidth	224,32 GB/s
Frame Buffer Bus Width	256 bits
Frame Buffer Location	Dedicated
Graphics Clock	626 MHz
Memory Clock	3505 MHz
Processor Clock	1253 MHz
RAM Type	GDDR5
Attached Monitors	1

Stworzone przeze mnie programy sumują wektory. Przykład zawiera dwie tablice, sumujemy ich elementy i przekazujemy wynik do trzeciej tablicy, a następnie je wyświetlamy.

Na początku zdefiniowana została jednakowa dla wszystkich programów ilość danych w tablicach:

```
const int numElements = 80000000;
```

Następnie określone zostało identyczne zapełnianie tablic:

```
for (int i = 0; i < numElements; i++)
        {
                a[i] = 1.456789999997*i;
                b[i] = 2.352167999997*i;
        }
```

Najważniejszą częścią programu była funkcja sumująca wektory. Dla programu sekwencyjnego wyglądała w ten sposób:

```
for (int i=0; i<numElements; i++)
        {
                c[i] = a[i] + b[i];
        }
```

W wersji dla OpenMP wystarczyło w nagłówku dopisać:

```
#include <omp.h>
```

A następnie, przed samą funkcją, dodać: **`#pragma omp parallel for`**. W związku z tym po zmianach funkcja wyglądała następująco:

```
#pragma omp parallel for
        for (int i=0; i<numElements; i++)
        {
                c[i] = a[i] + b[i];
        }
```

W obu przypadkach wystarczyło wyświetlić dane:

```
        for (int i=0; i<numElements; i++)
                printf(„%f + %f = %f\n”, a[i], b[i], c[i]);
```

Program dla CUDA był bardziej skomplikowany, ale nie wymagał znajomości programowania grafiki, nadal był oparty na języku C – jedynie doszło kilka słów kluczowych. Funkcja sumująca wektory zawierała **`__global__`** :

```
__global__ void sumVector( double *a, double *b, double *c)
{
        int idx = threadIdx.x + blockIdx.x * blockDim.x;
        while (idx<numElements){
                c[idx]=a[idx]+b[idx];
                idx+= blockDim.x * gridDim.x;
        }
}
```

Do iteracji można wykorzystać pętlę **while ()** i zwiększyć liczbę wątków, z których każdy jest logiczną jednostką, działającą równolegle. W tym miejscu określa się początkowy indeks kolejnego wątku: **int idx = threadIdx.x + blockIdx.x * blockDim.x**. Przy **idx+= blockDim.x * gridDim.x** implementujemy krok inkrementacji. Oznacza to, że działamy na blokach **blockIdx.x**, które zawierają wątki **threadIdx.x.** Natomiast **blockDim.x** daje liczbę wątków w bloku. Funkcja działa dopóki nie zsumuje tablic. W ten sposób **blockDim.x * gridDim.x** zwiększa liczbę wątków w grid.

W przypadku OpenMP wystarczyło zsumować i wyświetlić wyniki. Natomiast w CUDA należy alokować tablice na GPU. Wykonuje się to przy pomocy **cudaMalloc**. Następnie kopiuje się tablice a i b do GPU (używając funkcji **cudaMemcpy** z parametrem **cudaMemcpyHostToDevice**):

```
cudaMemcpy( dev_a, a, numElements * sizeof(double),
      cudaMemcpyHostToDevice);
cudaMemcpy( dev_b, b, numElements * sizeof(double),
      cudaMemcpyHostToDevice);
```

Funkcja **sumVector ()**, dzięki zastosowaniu słowa kluczowego **__global__**, zostanie wykonana na urządzeniu (karcie graficznej) **sumVector<<<128,128>>>(dev_a, dev_b, dev_c)**; – w nawiasach trójkątnych znajduje się liczba bloków jakie mają zostać utworzone oraz liczba wątków.

Kolejnym krokiem, jaki trzeba zrobić, jest skopiowanie tablicy c z GPU do CPU (tutaj także używa się funkcji `cudaMemcpy`, tylko tym razem wykonuje się ją z parametrem **cudaMemcpyDeviceToHost**).

```
cudaMemcpy( c, dev_c, numElements * sizeof(double),
cudaMemcpyDeviceToHost);
```

Pozostaje już tylko wyświetlenie wyników oraz zwolnienie pamięci na GPU funkcją **cudaFree ()**.

3. Szczegóły implementacyjne i wyniki eksperymentów

Do zmierzenia czasu w OpenMP oraz w programie sekwencyjnym zastosowano funkcję **`clock ()`** z biblioteki ctime. Dla CUDA użyte zostało zdarzenie służące do pomiaru czasu **`cudaEvent_t`**. W przypadku OpenMP uzyskano wynik 240 ms, sekwencyjnie program wykonywał obliczenia w 890 ms, CUDA – 14,3 ms.

Ten wynik pokazuje jednakże tylko i wyłącznie szybkość sumowania samej funkcji na GPU. Przy wykonanym pomiarze musi się również uwzględnić czas kopiowania do i z karty graficznej. W tym wypadku osiągnięto czas 190 ms dla CUDA, szybkość operacji została ograniczona przez interface PCI-express lub magistralę FSB. Aby przyspieszyć aplikację zdecydowano się skorzystać z mechanizmu pozwalającego na alokowanie pamięci na hoście **`cudaHostAlloc ()`**. Funkcja ta różni się od funkcji **`malloc ()`** tym, że ma wyłączone stronicowanie. Blokada pamięci przez **`cudaHostAlloc ()`** oznacza rezygnację z używania pamięci wirtualnej – przez to komputer musi mieć wystarczającą ilość pamięci fizycznej, aby zmieścić wszystkie zablokowane bufory. Jednakże, dzięki niej będzie można skorzystać ze strumieni służących do kolejkowania operacji, które mają być wykonywane do GPU (np. wywołanie jądra, zdarzenia początkowe i końcowe oraz kopiowanie pamięci). Spowodują tak zwane przeplatanie – uzyska się wówczas szybsze wykonanie kopiowania. W związku z tym należy zmodyfikować program.

Po pierwsze, trzeba sprawdzić czy urządzenie obsługuje przeplatanie:

```
int whichDevice;
cudaGetDevice (&whichDevice);
cudaGetDeviceProperties (&prop, whichDevice);
```

Po drugie, należy wskazać elementy zablokowanej pamięci używając `cudaHostAlloc ()`.

Później asynchronicznie przekazuje się dane na GPU poprzez funkcję cudaMemcpyAsync () korzystając ze strumieni:

```
cudaMemcpyAsync (dev_a0, host_a+1, numElements * sizeof(double),
cudaMemcpyHostToDevice, stream0);
cudaMemcpyAsync (dev_a1, host_a+1+numElements, numElements *
sizeof(double), cudaMemcpyHostToDevice, stream1);
cudaMemcpyAsync (dev_b0, host_b+1, numElements * sizeof(double),
cudaMemcpyHostToDevice, stream0);
cudaMemcpyAsync (dev_b1, host_b+1+numElements, numElements *
sizeof(double), cudaMemcpyHostToDevice, stream1);
```

W ten sam sposób kopiuje się dane z GPU do zablokowanej pamięci:

```
cudaMemcpyAsync (host_c+i, dev_c0, numElements * sizeof(double),
cudaMemcpyDeviceToHost, stream0);
cudaMemcpyAsync (host_c+i+numElements, dev_c1, numElements *
sizeof(double),  cudaMemcpyDeviceToHost, stream1);
```

Pozostało już jedynie zsynchronizowanie dwóch strumieni funkcją **`cudaStreamSynchronize ()`**, wyświetlenie wyników oraz uwolnienie strumieni i pamięci. Po dokonaniu powyższych zmian udało się otrzymać czas obliczeń wynoszący 89 ms.

Aby w pełni wykorzystać możliwości techniczne podjęto decyzję o napisaniu trzeciego programu, w którym wykorzystana będzie pamięć hosta, która nie wymaga kopiowania. Podobnie, jak w poprzedniej wersji programu, pamięć alokowana ulega zablokowaniu. Tym razem można uzyskać do niej dostęp z funkcji jądra. Oznacza to, że nie trzeba jej kopiować.

```
cudaHostAlloc( (void**)&a, numElements*sizeof(double),
cudaHostAllocWriteCombined | cudaHostAllocMapped);
cudaHostAlloc( (void**)&b, numElements*sizeof(double),
cudaHostAllocWriteCombined | cudaHostAllocMapped);
cudaHostAlloc( (void**)&partial_c,
numElements*sizeof(double), cudaHostAllocMapped);
```

Poprzednim razem użyto parametru **`cudaHostAllocDefault`**. Obecnie zastosowano **`cudaHostAllocMapped`**. Tym samym przekazane zostają GPU informacje, że następnie dojdzie do odwołania się bezpośrednio do bufora z poziomu karty graficznej. Dzięki temu przyspiesza się aplikację do 50 ms.

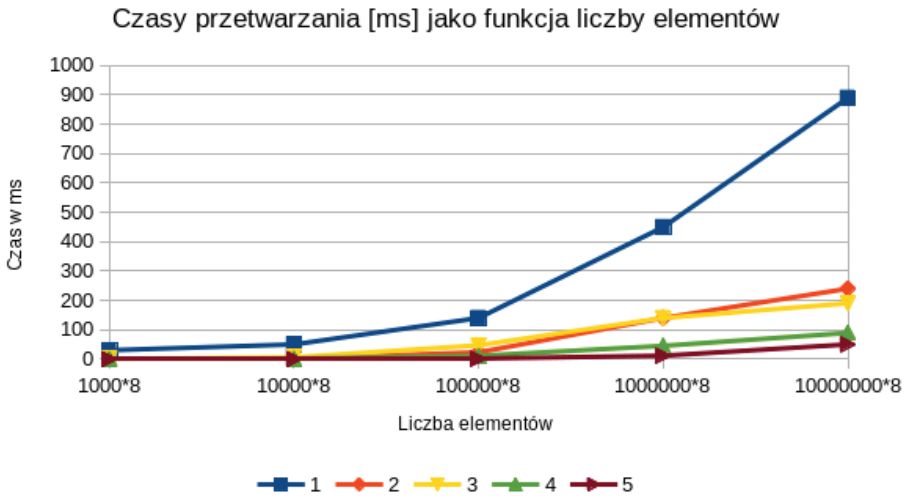
Tab. 3. Czasy przetwarzania [ms] – metoda vs. Liczba elementów

	1000*8	10000*8	100000*8	1000000*8	10000000*8
Sekwencyjnie:	30	50	140	450	890
OpenMP:	1	2	23	140	240
CUDA kopiowanie pamięci:	1	6	47	140	190
CUDA strumienie	0,6	1,2	11	45	89
CUDA pamięć niekopiowana	0	0,1	1,1	11,5	50

Podsumowując, dla 80 milionów elementów zadeklarowanych w tablicy najszybszym sposobem zrównoleglenia jest program napisany w CUDA, w którym wykorzystuje się pamięć niekopiowaną. Program wykonał się w czasie 50 ms. Dla strumieniowego rozwiązania wynik ten wyniósł 89 ms. Program, w którym skopiowano dane z RAMu do pamięci karty graficznej osiągnął czas 190 ms. W przy-

padku OpenMP osiągnięty wynik to 240 ms. W programie niezrównoleglonym (sekwencyjnym), którego użyto do porównania z programami zrównoleglonymi, czas wykonania programu wyniósł 890 ms.

W tab. 3 i na rys. 1 zaprezentowane zostały wyniki dla różnej ilości elementów, aby wykazać dla jakiej ilości elementów najlepiej zastosować zrównoleglenie w CUDA. Wyniki podane są w milisekundach (ms).



Rys. 1. Czasy przetwarzania: 1 – sekwencyjnie, 2- OpenMP, 3- CUDA kopiowanie pamięci, 4 – CUDA - strumienie, 5 – CUDA – pamięć niekopiowana

4. Wnioski

Z przeprowadzonych badań wynika, że najszybszą metodą zrównoleglenia programu jest CUDA przy użyciu pamięci nie kopiowanej.

Wiąże się to ze wszystkimi wadami i zaletami wykorzystywania zablokowanej pamięci, podobnie jak z korzystaniem ze strumieni w CUDA. Jeżeli program będzie wymagał pamięci nie zablokowanej, jedynym sposobem jest skopiowanie pamięci RAM do pamięci karty graficznej. Zaznaczyć tu należy, ze wyniki szybkości działania programu będą duże słabsze, niż w przypadku pamięci zablokowanej. Wiąże się to z prędkością magistrali FSB oraz interface PCI-Express.

Wadą CUDA jest to, że działa tylko z urządzeniami firmy NVIDIA. Jednakże, innowacyjność opisywanej technologii sprawia, że ma zastosowanie w wielu dziedzinach, między innymi w medycynie, inżynierii oraz ochronie środowiska. Dzięki CUDA C oraz interface utworzonym przez NVIDIĘ, tworząc równoległe

programy na GPU obsługujące technologię CUDA, nie jest wymagana znajomość tworzenia grafiki komputerowej oraz OpenGL i DirectX.

Warto zauważyć, że dla niektórych wartości program napisany w OpenMP wykonuje się szybciej niż wykonany w CUDA (patrz tab. 3 i rys. 1).

Bibliografia

[1] Cook S., CUDA Programming. A Developer's Guide to Paralel Computing with GPUs, Elsevier Inc., Waltham 2013.

[2] Kirk D., Hwu W., Programming massively Parallel Processors. A Hands-on Approach, Elsevier Inc., Burlington 2010.

[3] Sanders J., Kandrot E., CUDA w przykładach. Wprowadzenie do ogólnego programowania procesorów GPU, przekł. Ł. Piwko, Wydawnictwo Helion, Gliwice 2012.

[4] Wikipedia, en.wikipedia.org.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 29–40

Aleksander Wójcik

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

Projekt Lombok jako sposób na uelastycznienie kodu Javy

Project Lombok as a tool to make Java code more elastic

Streszczenie

Pierwsza część referatu jest wprowadzeniem do podstawowych mechanizmów języka Java. Omówione zostaną założenia języka, proces kompilacji, rola maszyny wirtualnej oraz zostanie wyjaśnione czym są i po co zostały wprowadzone adnotacje.

W części drugiej zostanie poruszone zagadnienie Abstrakcyjnego Drzewa Składniowego na przykładzie oraz zostaną opisane motywacje, sposób działania i możliwości projektu Lombok.

Słowa kluczowe: język Java, projekt Lombok, procesor adnotacji, Abstract Syntax Tree

Summary

The intention of the first part of this essay is to introduce the basics of Java programming language. The assumptions, compilation process, the role of virtual machine and annotation mechanism will be explained.

The second part is focused on Abstract Syntax Tree and project Lombok – its motivation, way of working and capabilities.

Keywords: Java Language, Project Lombok, annotations processor, Abstract Syntax Tree

1. Wprowadzenie

1.1. OGÓLNE INFORMACJE O JĘZYKU JAVA [1]

Podstawowe koncepcje języka Java zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią), natomiast składnia i wiele słów kluczowych z języka obiektowo-proceduralnego C++.

Język Java jest silnie zorientowany obiektowo. W kontekście własności języka o obiekcie można myśleć jako o składowej części programu, która może przyjmować określone stany i ma określone metody, które mogą zmieniać te stany bądź przesyłać dane do innych obiektów. Wyjątkiem od obiektowości są typy prymitywne: **`byte`**, **`short`**, **`int`**, **`long`**, **`double`**, **`float`** oraz specjalny typ **`void`**.

W Javie wszystkie obiekty są pochodną obiektu nadrzędnego (klasy java.lang. Object), z które-go dziedziczą podstawowe metody. Dzięki temu wszystkie klasy mają wspólny podzbiór podstawowych możliwości. W języku Java klasy mogą być wyprowadzone z innych klasy, w ten sposób dziedziną pola i metody tych klas. Z wyjątkiem klasy Object, która nie ma nadklasy, każda klasa ma jedną i tylko jedną bezpośrednią nadklasę (jednokrotne dziedziczenie). W przypadku braku wyraźnie określonej nadklasy, każda klasa jest bezwzględnie podklasą klasy Object. Podklasa dziedziczy wszystkie składowe (pola, metody oraz zagnieżdżone klasy) z jej nadklasy. Konstruktory nie są składowymi, więc nie są dziedziczone przez podklasy, ale konstruktor może być wywołany z podklasy poprzez użycie słowa kluczowego super(). Klasa Object, zdefiniowania w pakiecie java.lang, definiuje i implementuje zachowania wspólne dla wszystkich klas.

Każdy kod źródłowy programu składa się z zestawu klas pogrupowanych w pakiety.

Po napisaniu kodu źródłowego program kompilowany jest do bytecodu. Nie jest to jeszcze kod zrozumiały dla procesora w sposób bezpośredni, który pozwalałby nam na jego uruchomienie. Jest to jednak kod zapisany w określonym formacie, który może zostać poprawnie zinterpretowany przez Maszynę Wirtualną Java JVM (z ang. Java Virtual Machine), przetłumaczony na kod wykonywalny i uruchomiony.

Ważne jest to, że w Javie nie możemy bezpośrednio skompilować program do kodu maszynowego. Program zawsze jest kompilowany do kodu pośredniego, który jest następnie interpretowany przez JVM. JVM rozpoznaje system operacyjny oraz sprzęt, na jakim jest program odpalany i dostosowuje do niego wynikowy kod maszynowy. W ten sposób realizowana jest koncepcja przenośności programów napisanych w Javie. JVM zarządza również pamięcią.

1.2. Kompilacja i uruchomienie programu korzystając z JVM

Pliki źródłowe, które mogą być kompilowane mają rozszerzenie *.java. Kompilacja polega na wydaniu polecenia javac z argumentem pliku źródłowego. Powstaje wówczas plik *.class. W procesie kompilacji kompilator Java (program javac) tłumaczy kod programu na Byte Code JVM – pliki z rozszerzeniem .class. W czasie działania aplikacji Byte Bode jest tłumaczony, przez JVM, na kod maszynowy i wykonywany bezpośrednio na procesorze. Kod maszynowy wytworzony przez JVM nie jest nigdzie zapisywany, jest on generowany "w locie".

Przykładowo, aby uruchomić plik MyProgram.java z metodą główną wykonujemy kolejno:

javac MyProgram.java
java MyProgram

Rys. 1. Proces kompilacji programu MyProgram.java do kodu pośredniego (pliku MyProgram.class) za pomocą kompilatora javac oraz interpretacja kodu pośredniego do kodu maszynowego (zera i jedynki) przez JVM

1.3. Adnotacje

Nowością w stosunku do starszych języków programowania jest mechanizm adnotacji zapożyczony z języka C#. Adnotacje to dokładnie – metadane. Mogą być stosowane wobec: klasy, metody, interfejsu, pola klasy, argumentu metody, konstruktora, pakietu, zmiennej lokalnej, innej adnotacji.

Adnotacje są podstawowym narzędziem programowania aspektowego, zarządzania bean-ami w kontenerach Dependency Injection, konfiguracji systemów opartych na frameworkach klasy enterprise (Spring, EJB). Natomiast w projekcie Lombok stanowią pewnego rodzaju „polecenia generacji kodu", o czym poniżej. Warto zwrócić uwagę, że w kontekście projektu Lombok adnotacje przestają być już tylko metadanymi, ale zyskują nowe znaczenie – stanowią pewnego rodzaju instrukcje dla procesora jaki kod należy wygenerować przed ostateczną kompilacją programu do kodu pośredniego.

Przykładowymi adnotacjami w JDK są:

1. **`@Override`**
   Adnotacją tą oznaczane są metody, których przeznaczeniem jest nadpisanie metody z klasy nadrzędnej. Jeżeli kompilator nie znajdzie w klasie nadrzędnej metody o tej samej (z dokładnością do kowariantnych typów danych) sygnaturze – wówczas program się nie skompiluje
2. **`@SuppressWarnings(„unchecked”)`**
   W skrócie powoduje, że ostrzeżenia kompilatora typu „unchecked” nie będą wyświetlane na konsoli.

1.4. Działanie procesora adnotacji [4]

Procesor adnotacji działa w trakcie kompilacji programu i jest możliwe zaprogramowanie odpowiednich procedur w kontekście użytych adnotacji. Deklarowany przez nas procesor adnotacji dziedziczy bezpośrednio z klasy **`javax.annotation.processing.AbstractProcessor.`** Musimy zadbać o nadpisanie jednej metody abstrakcyjnej:

```
public abstract boolean process(Set<? extends TypeElement>
annotations,RoundEnvironment roundEnv);
```

Metoda zwraca typ logiczny – mówiący o tym, czy dane adnotacje zostały przez ten konkretny procesor obsłużone. Jeśli metoda zwróci **`true`**, wtedy kolejne procesory nie będą proszone o obsłużenie danej adnotacji. W kontekście referatu ważniejsze są argumenty metody niż to, co ona zwróci.

Pierwszym argumentem jest zbiór typów, które są żądane do obsłużenia. W szczególności procesor musi umieć zareagować na sytuację, gdy zbiór jest pusty.

Drugim argumentem jest „środowisko” zawierające w sobie informacje o kontekście użycia adnotacji. Jest to dość rozbudowana klasa.

Najważniejszą metodą środowiska jest :

```
Set<? extends Element> getElementsAnnotatedWith(Class<? extends Annotation> a)
```

która zwraca wszystkie elementy oznaczone daną adnotacją, dzięki czemu możliwe jest iterowanie po nich i wymuszenie spójnego działanie adnotacji. To właśnie w tej „iteracji” zachodzi właściwe działanie instrukcji, wymuszonych użyciem adnotacji.

Warto na tym etapie zaznaczyć, że procesor adnotacji nie umożliwia edytowania już istniejących i załadowanych klas. W przypadku adnotacji @Override procesor zgłasza wyjątek i przerywa kompilacje, gdy warunek, że metoda oznaczana adnotacją nadpisuje metodę nadklasy, nie jest spełniony. Adnotacja @SuppressWarnings blokuje pisanie niektórych komunikatów na wyjście. W obu przypadkach nie ma modyfikacji działania danej klasy.

Warto zaznaczyć, że jest możliwe definiowanie nowych klas w ramach działania procesora adnotacji.

2. Projekt Lombok

2.1. Uelastycznienie kodu Java jako cel powstania projektu

Zwyczajową nazwą publicznej metody, której jedynym celem jest zwrócenie wartości jej pola jest nazwa „getter". Zwyczajową nazwą publicznej metody, której jedynym celem jest zmienienie wartości jej pola jest nazwa „setter".

Jedną z poważniejszych wad języka Java jest generowanie dużej ilości metod zbędnych – np. getterów i setterów – mających za zadanie kontrolę nad dostępem do pól klasy zgodnie z podstawowym założeniem obiektowego paradygmatu programowania – hermetyzacji. Inne języki radzą sobie z tym lepiej. Poniżej porównanie tej samej funkcjonalności – klasy Square z polem side – w językach C# oraz Java:

```
   //Java
public class Square{
   private int size;

   public void setSide (int size){
   this.size = size
   }
   public int getSide (){
   return this.size;
   }
   }

   //C#
   public class Square {
   private int Side{get;set;};
   }
```

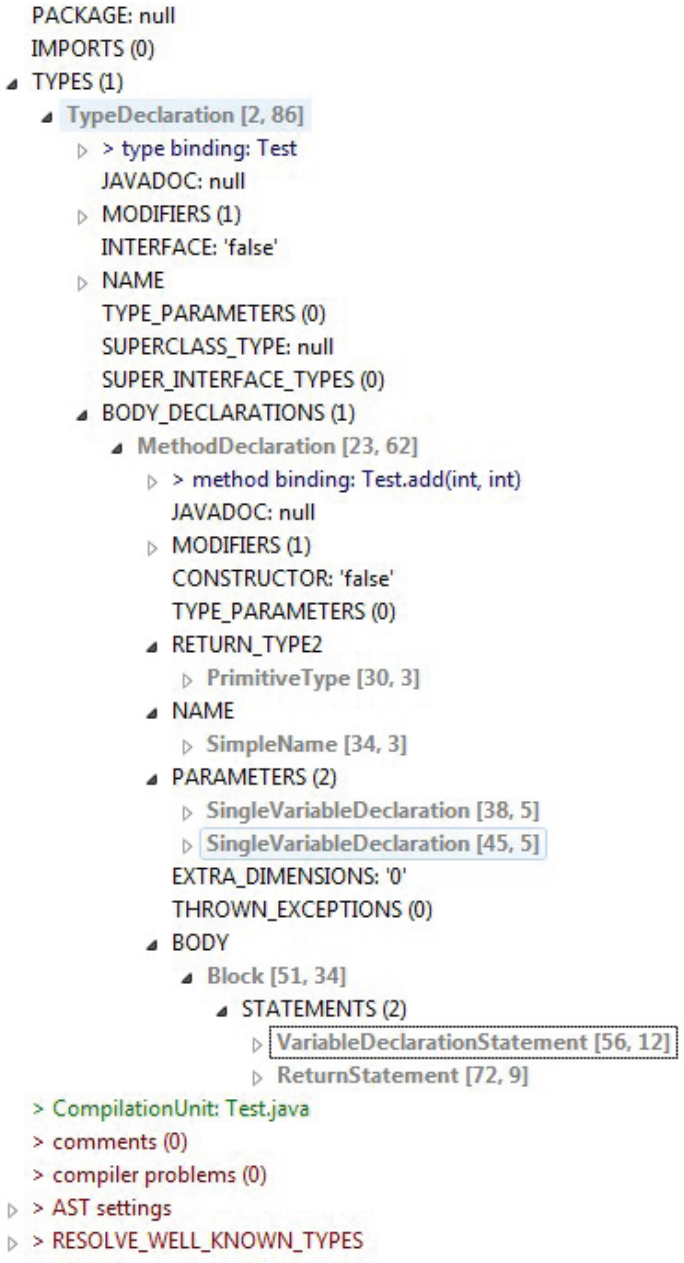
Zazwyczaj programista nie pisze getterów i setterów bez użycia IDE. W każdym podstawowym IDE (Intellij IDEA, Eclipse, Netbeans) jest możliwość automatycznej generacji w/w metod. Jednak za każdym razem, gdy programista zmienia dane pole powinien usunąć stare gettery i settery oraz wygenerować nowe, co jest zadaniem niepotrzebnie uciążliwym. Inne metody, które są często generowane przez IDE: equals(), toString(), hashCode().

2.2 Abstrakcyjne drzewo składniowe (AST) [2]

Abstrakcyjne Drzewo Składniowe (ang. *Abstract Syntax Tree (AST)*) jest reprezentacją programu źródłowego napisanym w języku programowania np. Java w postaci drzewa. Powstaje ono w jednym z etapów kompilacji kodu źródłowego. Jest ono dużo mniej czytelne dla programisty niż kod źródłowy, jednak jest niezbędne dla procesu kompilacji. Poniżej porównanie tej samej funkcjonalności w postaci kodu Java oraz w postaci drzewa.

```
public class Test {
        public int add(int a, int b){
                int c = a+b;
                return c;
        }
}
```

Rys. 2. Przykład klasy Test z metodą add



Rys. 3. Klasa test z metodą add. W AST jest bardzo dużo elementów, których nie ma w kodzie np. informacje, że klasa nie jest interfejsem: INTERFACE: false

Rys. 4. Reprezentacja klasy test w postaci drzewa AST bez ciała metody – w węźle pod BODY wstawiony został trzykropek

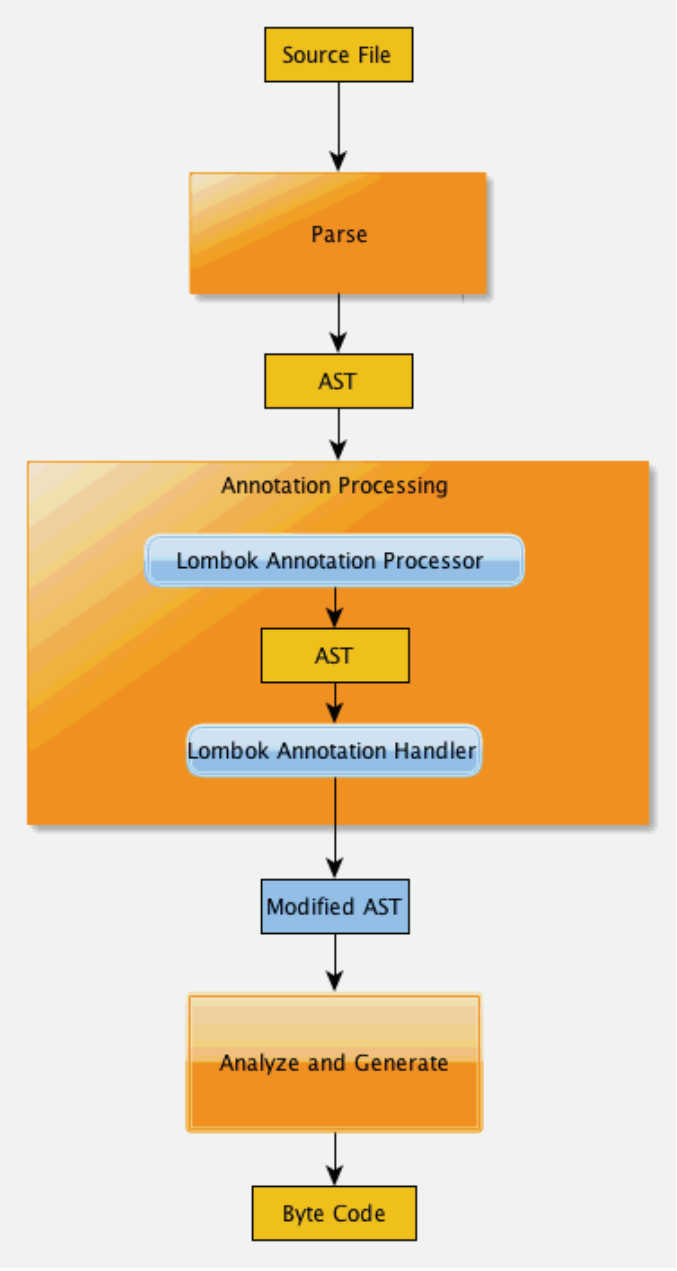
2.3 Manipulacja AST podstawowym narzędziem działania projektu Lombok

Projekt Lombok w założeniu wychodzi poza ograniczenia działania procesora adnotacji, ponieważ z założenia działa inaczej w fazie kompilacji:

Projekt Lombok korzysta z narzędzi dostarczonych przez firmę Sun (podpakietów pakietu com.sun.tools.javac) do manipulacji na AST oraz swoich metod pomocniczych tworzących zupełnie nowe API. Mimo, że całe działanie jest wykonywane w języku Java, API do manipulacji AST jest skomplikowane.

Każda adnotacja jest obsługiwana przez klasę dziedziczącą z abstrakcyjnej klasy generycznej **`lombok.javac.JavacAnnotationHandler<T extends Annotation>`** i nadpisującą jej metodę abstrakcyjną:

```
public abstract void handle(AnnotationValues<T> annotation,
JCAnnotation ast, JavacNode annotationNode);
```



Rys. 5. Proces kompilacji pliku źródłowego (Source File) do kodu pośredniego ([Java} Byte Code) z wykorzystaniem modyfikacji drzewa AST

Warto zwrócić uwagę, że w projekcie Lombok sygnatura metody jest inna niż w przypadku korzystania ze zwykłego procesora adnotacji (**`AbstractProcessor`**). W parametrze występują następujące argumenty:

1. obiekt typu **`AnnotationValues`** – właściwa adnotacja, która aktualnie jest obsługiwana
2. obiekt typu **`JCAnnotation`** – węzeł AST reprezentujący adnotację
3. obiekt typu **`JavacNode`** – specjalny obiekt projektu Lombok opakowujący węzeł AST – węzeł ten rozszerza możliwości domyślnego węzła AST z JDK Javy o możliwość przechodzenia w górę i w dół po drzewie AST. Jest to funkcjonalność zaimplementowana właśnie w projekcie Lombok. Z tego węzła drzewa AST możliwe jest też czytanie i zmienianie innych węzłów AST

Poniżej dla przykładu nagłówek odpowiedzialny za obsłużenie adnotacji @Getter [3]:

```
@Override
public void handle(AnnotationValues<Getter> annotation,
JCAnnotation ast, JavacNode annotationNode){
///…
}
```

2.4 Przykłady możliwości oferowanych przez projekt Lombok [3]

1. Generowanie getterów i setterów – poniżej – zwięzła wersja klasy Square z polem side:

```
public class Square{
        @Getter
        @Setter
        private int size;
}
```

Z kolei poniższa równoważna deklaracja klasy spodowuje wygenerowanie getterów i setterów dla wszystkich prywatnych niefinalnych pól klasy, jeśli byłoby ich więcej:

```
@Getter
@Setter
public class Square{
        private int size;
}
```

2. Generowanie getterów i setterów, metod toString(), equals(), hashCode() oraz konstruktora z parametrami analogicznymi do pól klasy. Wszystko powyższe jest zawarte w zwięzłej adnotacji @Data:

```
@Data
public class Square{
        private int size;
}
```

3. Metody oznaczane adnotacją @SneakyThrows mogą wyrzucać *checked exceptions*, (tj. wyjątki, które w normalnym działaniu programu muszą być obsłużone lub umieszczone w sygnaturze metody) bez konieczności deklarowania ich w sygnaturze metody.

4. Tworzenie niezmiennych klas bez dodatkowych zabiegów programistycznych – adnotacja @Value na klasie

3. Podsumowanie

Programy napisane w języku Java nie są bezpośrednio kompilowane do kodu maszynowego rozumianego przez procesor, ale do kodu pośredniego interpretowanego przez JVM. Sam proces kompilacji też jest wieloetapowy, co pozwala na uruchamianie odpowiednich programów (narzędzi) pomiędzy poszczególnymi etapami kompilacji – np. projektu Lombok.

Adnotacje w języku Java są metadanymi dla pól, metod, klas, argumentów, a także dla innych składowych języka Java. W projekcie Lombok stanowią natomiast instrukcje dla procesora – powodują generowanie dodatkowego kodu w trakcie kompilacji. Używanie adnotacji projektu Lombok powoduje zwiększenie czytelności kodu, zmniejszenie objętości kodu, a także daje nowe możliwości (np. adnotacja @SneakyTrows) niedostępne w inny sposób. Istnieje możliwość rozszerzenia projektu Lombok i zaprogramowania własnych adnotacji. Wykracza to poza zakres tego referatu.

Bibliografia

[1] Java Tutorial, Oracle Corporation, data odwiedzin:14.07.2015 r. http://docs.oracle.com/javase/tutorial/.

[2] Wpis na blogu programistycznym, data odwiedzin:14.07.2015 r. http://www.programcreek.com/2012/04/represent-a-java-file-as-an-astabstract-syntax-tree/.

[3] Strona główna projektu Lombok, data odwiedzin:14.07.2015 r. https://projectlombok.org/features/.

[4] Fragment dokumentacji JDK Javy, Oracle Corporation, data odwiedzin:14.07.2015 r. http://docs.oracle.com/javase/7/docs/api/javax/annotation/processing/AbstractProcessor.html.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 41–48

Aleksandra Pierepienko

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

Budowa zdalnie sterowanego robota

Construction of remote-controlled robot

Streszczenie

W niniejszym referacie przedstawię budowę zdalnie sterowanego robota, przygotowanego w projekcie stażowym. Jego główną jednostką będzie Raspberry Pi 2. Robot sterowany będzie kontrolerem do konsoli XBOX 360. Do programowania wykorzystano system Raspbian oraz język Python2, z biblioteką pygame.

Summary

Construction of remote-controlled robot will be presented in this paper. It has been prepared as practice project. Raspberry Pi2 has been used as main controlling unit. This robot is controlled with XBOX 360 console. Raspbian system and Python2, with pygame library have been used for software development.

Słowa kluczowe: Raspberry Pi, robotyka, silniki, serwomechanizmy, elektronika, programowanie mikrokontrolerów, sterowanie silnikami

Keywords: Raspberry Pi, robotics, engines, servomechanism, electronics, microcontroller programming, engine control

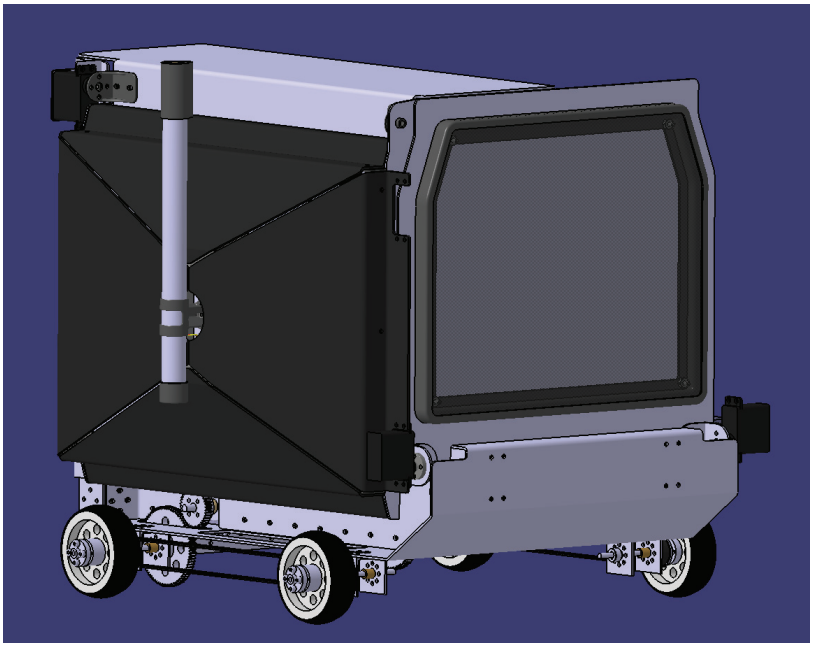
1. Wygląd i części mechaniczne

1.1. Wygląd zewnętrzny

Robot został skonstruowany tak, by stanowić przede wszystkim atrakcję dla dzieci. Będzie on wykorzystywany na urodzinach do wwożenia tortu na salę zabaw, oraz jako urozmaicenie przedstawień teatrzyku kukiełkowego w przedszkolach.

Zamysł przewidywał kojarzenie się wyglądu robota z łazikami kosmicznymi. Do jego budowy wykorzystano części z zestawu TETRIX, stelaż oraz pokrycie wykonano z aluminium wycinanego laserowo, oraz elementów plastikowych wykonanych w technologii druku 3D.

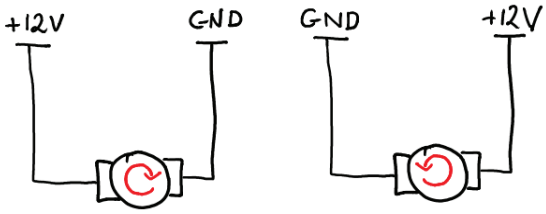
Robot ma wiele ruchomych części. Pokrycie odsłania się, a jego dno unosi. Do tego elementem uatrakcyjniającym całą konstrukcje są ruchome race symulujące silniki rakietowe.



Rys. 1. Wstępny projekt robota

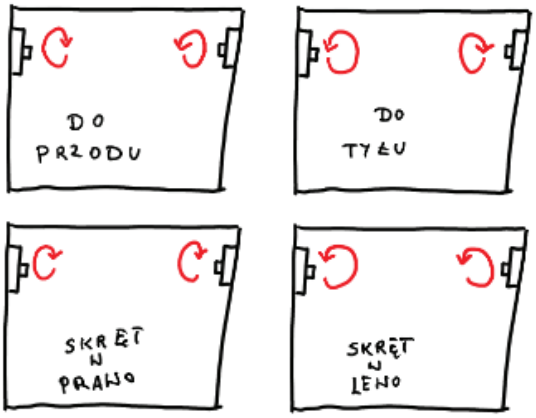
1.2. Główny napęd

Robot napędzany jest dwoma silnikami DC. Umieszczone są one z przodu robota, przy lewym i prawym kole. Tylne koła napędzane są za pomocą pasa przenoszącego obroty koła przedniego. Silniki są sterowane za pomocą mostka H.



Rys. 2. Sterowanie kierunkiem obrotów silnika DC

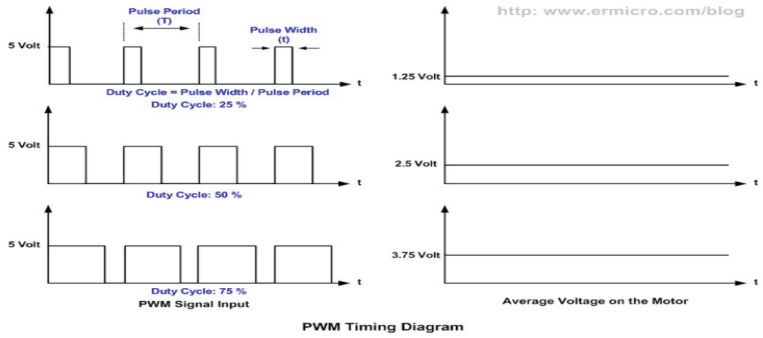
Sterowanie silnikami DC obywa się poprzez zmianę kierunku przepływu prądu. Jeśli do pinu numer 1 damy „+", a do pinu numer 2 „-", silnik będzie obracał się zgodnie z ruchem wskazówek zegara. Jeśli natomiast zamienimy podłączenie pinów, silnik będzie kręcił się w przeciwną stronę. Przedstawiono to na rys. 2. Jazda do przodu i do tyłu odbywa się poprzez wysterowanie ruchów silników w przeciwną stronę – ponieważ będą umieszczone po przeciwległych stronach. Jeśli silniki będą się poruszały w tą samą stronę nastąpi skręt w lewo lub w prawo, w zależności od kierunku obrotu.



Rys. 3. Schemat działania kierunku obrotu silników

1.3. Serwomechanizmy

W robocie wykorzystano 8 serwomechanizmów. Działają one w parach i obracają się w stosunku do siebie w przeciwną stronę.



Rys. 4. Przykładowy diagram sygnałów PWM

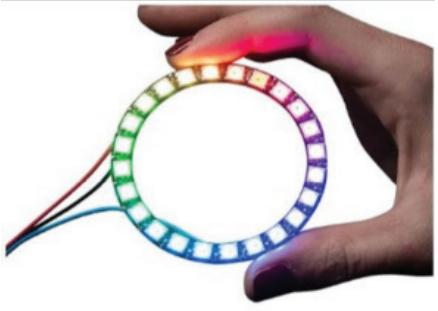
Do wysterowania serwomechanizmów wykorzystano 16 kanałowy sterownik PWM firmy Adafruit [4]. Serwomechanizmy sterowane są za pomocą sygnału PWM. PWM (ang. Pulse-Width Modulation) to metoda regulacji sygnału napięciowego o stałej amplitudzie i częstotliwości, polegająca na zmianie wypełnienia sygnału. Dzięki temu można precyzyjnie sterować kątem wychylenia ramienia serwomechanizmu.

1.4. Oświetlenie

Robot został odpowiednio podświetlony. Jego przednimi reflektorami są taśmy LED w kolorze białym, zasilane 12V. Do tego boki robota będą doświetlone diodami RGB które będą zmieniały kolor według wskazań sterownika.

Najciekawszym elementem oświetlenia jest NeoPixel. Jest to listwa złożona z 24 LEDów, zabudowana w płytkę o kształcie okręgu.

Diody wykorzystane w tym układzie mają interfejs szeregowy. Oznacza to, że każda dioda ma 4 piny – dwa zasilające, jeden pin „IN" do przyjmowania sygnału oraz jeden pin „OUT" do przesyłania sygnału do kolejnych elementów. Każda dioda ma przypisany adres, dzięki czemu można programować zaawansowane animacje.



Rys. 5. Adafruit NeoPixel

Podstawową technologią służącą do tworzenia dokumentów WWW jest język HTML, który jest odpowiedzialny za poinformowanie przeglądarki w jaki sposób ma być prezentowany tekst oraz wszelkie inne elementy umieszczone na stronie. W celu separacji treści stron internetowych zapisanych w języku HTML od sposobu ich prezentacji stosowany jest mechanizm CSS (*Cascade Style Sheet*).

2. Oprogramowanie

2.1. System operacyjny i środowisko programistyczne

Pierwotnie robotem sterować miała kostka LEGO EX3, jednakże jednostką centralną stanie się minikomputer Raspberry Pi 2. Został on wybrany ze względu na cenę, dobre parametry oraz możliwość łatwej obsługi pinów GPIO.

Raspberry Pi działa pod kontrolą systemu Raspbian. Do programowania użyto języka Python 2, ponieważ jest on głównym językiem do obsługi pinów GPIO w Raspberry. Dostępne jest szerokie wsparcie społeczności, a większość użytych bibliotek była dostępna właśnie w tym języku.

2.2 Biblioteka adafruit NeoPixel

Do obsługi modułu NeoPixel wykorzystaliśmy bibliotekę dystrybutora dostępną na jego stronie internetowej [7].

Podstawowe parametry wyglądają następująco:

```
# Konfiguracja łańcucha LED:
LED_COUNT   = 24      # Liczba diód LED w łańcuchu
LED_PIN     = 18      # Pin GPIO do kontroli – z obsługą PWM
LED_FREQ_HZ = 800000  # częstotliwość sygnału LED (domyślne – 800khz)
LED_DMA     = 5       # Kanał DMA użyty do generowania sygnału
LED_INVERT  = False   # Domyślne
```

Po zdefiniowaniu prostych funkcji sterowanie listwą odbywa się w prosty sposób. Wybiera się numer diody, do tego kolor z jakim ma świecić z palety RGB (Red, Green, Blue) i przypisane kolorowi numery wpisuje się według wzoru:

```
Nazwa_funkcji(numer_diody, Color(Red_numer, Green_numer, Blue_numer))
```

Np:

```
theaterChase(3, Color(127, 127, 127))
```

2.3 Biblioteka Pygame i obsługa kontrolera od XBOX 360

Podstawą obsługi naszego robota jest biblioteka Pygame [6]. Jest to biblioteka do tworzenia gier komputerowych oraz aplikacji multimedialnych w języku Python. Dzięki tej bibliotece można obsłużyć kontroler od konsoli XBOX 360. Biblioteka stanowi wolne oprogramowanie.

Szukając przykładów użycia kontrolera znaleziono podobny projekt – samochodzik RC sterowany padem. Wykorzystano udostępniony program który przechwytywał i wyświetlał wydane komendy [5].

Aby wykorzystać ten program należy zainstalować sterownik do kontrolera. Znajduje się on w repozytorium, wystarczy więc wpisać komendę

```
sudo apt-get install xboxdrv
```

a następnie uruchomić ją w tle poleceniem:

```
sudo xboxdrv –silent
```

Fragment programu przechwytującego wygląda następująco:

```
def myCallBack(controlId, value):
        print “Control id = {}, Value = {}”.format(controlId,
↳value)

xboxCont = XboxController.XboxController(
        controllerCallBack = myCallBack)
```

Program drukuje numer wciśniętego klawisza, oraz jego stan. W przypadku klawiszy cyfrowych jest to 0 i 1, a w przypadku analogowych dźwigni stany od -100 do 100, na przykład:

```
LX = 90 # analogowy drążek w pozycji 90 na osi X
LY = 50 # analogowy drążek w pozycji 50 na osi Y
Control id = 13, Value = 1 # Przycisk nr 13 został wciśnięty
Control id = 13, Value = 0 # Przycisk nr 13 został zwolniony
```

Działając na podstawie biblioteki Pygame i powyższego programu udało się napisać proste sterowanie.

2.3 Sterowanie serwomechanizmami

Do sterowania serwomechanizmami oraz prędkością silników wykorzystano sterownik PWM. Jest on kontrolowany po magistrali I2C. Prosty program, który porusza serwem od poziomu minimalnego do maksymalnego:

```
while (True):
pwm.setPWM(0, 0, 150) # wybieramy serwomechanizm na ↳pozycji 0,
długość sygnału PWM to 150
        time.sleep(1)
        pwm.setPWM(0, 0, 600)
        time.sleep(1)
```

2.3 Sterowanie robotem

Ponieważ robot nie ma żadnych ruchów autonomicznych i jest sterowany przez użytkownika cały program opiera się na prostych instrukcjach warunkowych. Poniżej fragment programu który kieruje silniki robota w ruchu zgodnym ze wskazówkami zegara z użyciem mostka H, za pomocą analogowej dźwigni:

```
if(ControlId == 1 and value > 50)
        GPIO.output(silnikprawy_1, HIGH)
        GPIO.output(silnikprawy_2, LOW)
        GPIO.output(silniklewy_1, HIGH)
        GPIO.output(silniklewy_2, LOW)
```

Zmienne „silnik" oznaczają który silnik jest sterowany oraz który pin podłączony jest do mostka H. W przypadku instrukcji dotyczącej „silniklewy_2" na drugi pin lewego silnika dajemy stan niski. W pozostałych – analogicznie.

3. Podsumowanie

Budowa robota była ciekawym wyzwaniem. W najbliższym czasie być może poszerzą się jego możliwości – może o komplet czujników, by stał się bardziej autonomiczny lub o mechaniczne ramię które umożliwi mu chwytanie różnych przedmiotów?

Gorąco zachęcam do konstrukcji własnych robotów na platformach Raspberry Pi, Arduino lub Lego Mindstorms.

Bibliografia

[1] Borkowski P., Przygoda z elektroniką, wyd. Helion, Gliwice, 2013.

[2] Cook D., Budowa robota dla początkujących, wyd. Helion, Gliwice, 2012.

[3] Cook D., Budowa robota dla średniozaawansowanych, wyd. Helion, Gliwice, 2013.

[4] https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi/ odwiedzone w dniu 01.07.2015.

[5] http://www.stuffaboutcode.com/2014/10/raspberry-pi-xbox-360-controller-python.html, odwiedzone w dniu 01.07.2015.

[6] http://www.pygame.com, odwiedzona w dniu 01.07.2015.

[7] https://learn.adafruit.com/adafruit-neopixel-uberguide/overview, odwiedzone w dniu 01.07.2015.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 49-56

Damian Radowiecki

Wyższa Szkoła Ekonomii i Innowacji w Lublinie

C++11 – najnowszy standard języka C++

C++11 – the newest C++ language standard

Streszczenie

C++11 jest najnowszym standardem języka C++. Celem niniejszego referatu jest przybliżenie najistotniejszych według autora innowacji w języku takich jak np.: słowo kluczowe auto, pętla zakresowa czy nowy sposób alokowania pamięci dynamicznej. Praca nie obejmuje całego zakresu nowego standardu języka, lecz porusza jedynie wybrane tematy. Każdy rozdział dodatkowo opatrzony jest krótkim kodem źródłowym lub też wycinkiem z kodu.

Summary

C++11 is the newest C++ language standard. The purpose of this essay is to show the most relevant, according to the author, innovations in the language, such as: auto keyword, range-for loop or the new way of dynamic memory allocation. The essay does not include the full scope of the new language standard, but only chosen topics. Each unit additionally has a short source code or a shortened code.

Słowa kluczowe: C++11, najnowszy standard C++

Keywords: C++11, the newest standard C++

1. Uwaga wstępna

Warto przed przejściem do omawiania nowości języka, zaznaczyć, że niektóre kompilatory nie mają włączonej automatycznie obsługi standardu C++11. Przykładowo dla kompilatora g++ przy kompilacji dodajemy parametr:

```
    -std=c++0x
lub:
    -std=c++11
```

1. Słowo kluczowe auto, czyli dedukcja typu

Pierwotnie, w języku C i C++ słowo kluczowe auto posiadało swoją praktycznie niewykorzystywaną funkcję, mianowicie oznaczało, że dana zmienna jest automatyczna (istnieje tylko w określonym bloku, czyli jest zmienną lokalną automatyczną), co w praktyce nie wymagało oznaczenia słowem auto [2].

W standardzie C++11, słowo to otrzymało nową funkcjonalność – umiejętność dedukcji typu. Jest to cenna zdolność zwłaszcza, jeśli mamy do czynienia z „trudnymi" typami takimi ja np. iterator kolekcji.

Użycie słowa kluczowego auto jest bardzo proste, wystarczy umieścić po nim nazwę zmiennej i zainicjalizować ją wartością:

```
int main(){
    auto i=20;// typ int
    vector<int> a;
    auto j=a.begin();//iterator typu vector
    }
```

2. Nowe literały napisowe–czyli jak uniknąć znaków sterujących

Aby uniknąć w łańcuchach używania znaków sterujących, należy każdorazowo poprzedzać je znakiem backslash, co często bywa bardzo niewygodne. W celu uniknięcia takich problemów powstały nowe literały napisowe [1]. Użycie ich odbywa się według następującej składni:

```
R"ogranicznik( dowolny tekst )ogranicznik";
```

gdzie, jako ogranicznik stosuje się sekwencję znaków składającą się, z co najwyżej 16 znaków prostych (bez ukośników, spacji czy nawiasów). Należy pamiętać, aby ogranicznik wraz z poprzedzającym go nawiasem i następującym po nim cudzysłowie, nie powtórzył się we wprowadzonym tekście np.:

```
#include<iostream>
```

```
using namespace std;
int main(){
cout<<R"literal(¥n¥n¥lit¥n¥)literal"<<endl;
//wyisze:¥n¥n¥lit¥n¥
//cout<<R"c(dowolny tekst)c"dalszy ciąg tekstu ) ↳c";
// błąd!!!

}
```

3. Drobna zmiana w wyrażeniach szablonowych

Zmiana ta dotyczy wyrażeń szablonowych [1], a dokładnie odstępów pomiędzy nawiasami ostrymi, które w poprzedniej wersji były konieczne:

```
vector<list<double> >;
```

Teraz możliwe jest używanie następującej składni:

```
vector<list<double>>;
```

4. Puste wskaźniki – nullptr

Słowo kluczowe **`nullptr`** nie wprowadza wielkich rewolucji, pozwala na zastąpienie symbolu 0 i symbolu **`NULL`** przy definiowaniu pustych wskaźników nowym słowem kluczowym – **`nullptr`** [1].

```
#include<iostream>

using namespace std;
int main(){
        int *wsk;
//wsk=0; //stara składnia
        //wsk=NULL; //stara składnia
        wsk=nullptr;
}
```

5. Pętla zakresowa – rozszerzenie funkcjonalności pętli for

Dzięki nowemu standardowi można za pomocą pętli for iterować po całej tablicy lub kolekcji bez wyznaczania początku i zakończenia iteracji. Jest to duże ułatwienie znane już z innych języków programowania, a teraz przeniesione na pole

języka C++. Zasada działania jest niezwykle prosta, wystarczy umieścić pomiędzy nawiasami zmienną tego samego typu, co elementy tablicy lub kolekcji i po znaku dwukropka umieścić zakres, który będzie podlegał iteracji (tablica/kolekcja)[3]:

```
        for(typ x: kolekcja){
//instrukcje
 }
```

6. Listy inicjalizacyjne. Nowa składnia inicjalizacji

Dzięki nowej składni inicjalizacji można w łatwy sposób przypisać wartości zerowe lub jeśli chodzi o wskaźniki, wartości nullptr do zmiennych. Aby to uczynić wystarczy po nazwie zmiennej umieścić dwa nawiasy klamrowe [2]:

```
   int main(){
        int j{};//inicjalizacja zerem
        double *k{};//inicjalizacja wartością nullptr
        }
```

Składnia nie pozwala jednak na inicjalizację z zawężeniem wartości (wymagającą rzutowania), przykładowe błędy:

```
int main(){
        //int a{4.6};// błąd!!!
        //char b{123}; błąd!!!
        }
```

Jak widać w drugim przykładzie, zawężenie wartości dotyczy także tego, czy można daną wartość reprezentować bezstratnie.

Aby można było korzystać z list inicjalizacyjnych w dowolnym miejscu programu (np. klasie, funkcji) powstała tzw. lista inicjalizacyjna **`std::initializer_list`**<>. Przykładowym jej zastosowaniem może być konstruktor klasy [2]:

```
#include<iostream>

using namespace std;

class P {
public:
        P(std::initializer_list<int>nazwa){
        }
        ~P(){}
```

```
        };
int main(){
        P a{2,3,4,5,6};
}
```

Przeglądanie listy odbywa się za pomocą iteratora, którego początkową wartość możemy ustawić za pomocą funkcji **`nazwa.begin()`** lub **`nazwa.end()`**.

7. Nowy sposób dynamicznej alokacji pamięci–wskaźniki `shared_ptr`, `unique_ptr`

Po funkcjach **`malloc`** i **`free`** języka C, operatorze new i delete w C++ przyszedł czas na kolejny sposób dynamicznej alokacji pamięci. Standard C++11 proponuje programistom wskaźniki **`shared_ptr`** i **`unique_ptr`** [2]. Zarówno pierwszy, jak i drugi służą do dynamicznej alokacji pamięci. Różni je jedynie to, że pierwszy z nich pozwala na wskazywanie przez inne wskaźniki tego samego adresu, a drugi na to nie pozwala. Składnia wygląda następująca:

```
#include<memory> //biblioteka obsługująca wskaźniki
//shared_ptr, unique_ptr
        ...
        shared_ptr<typ> nazwa<typ>
        unique_ptr<typ> nazwa<typ>
```

Przy dynamicznej alokacji tablic:

```
        shared_ptr<typ[]> nazwa<typ[rozmiar]>
        unique_ptr<typ[]> nazwa<typ[rozmiar]>
```

Korzystnym faktem używania nowego sposobu dynamicznej alokacji pamięci jest to, że programista nie musi pamiętać o jawnym zwolnieniu pamięci, odbywa się to niejawnie. Istnieje oczywiście możliwość jawnego zwolnienia pamięci, służy do tego funkcja składowa **`reset()`**:

```
        wskaznik.reset();
```

Godnym uwagi jest fakt, że nowy sposób alokacji pozwala na nadmiarowe zwalnianie pamięci bez żadnych konsekwencji, na co trzeba było uważać przy używaniu operatora **`delete`**.

Warto też wspomnieć o dostępie do danych umieszczonych w pamięci za pomocą wskaźników. Odbywa się to na prawie identycznej zasadzie, jak przy normalnych wskaźnikach. Jedyną operacją, której nie możemy wykonywać jest przesuwanie wskaźnika do tablicy w sposób arytmetyczny np.:

```
*(wskaznik_do_tablicy_obiektow_int+3)=23;// błąd!!!
```

Wszystkie inne operacje na wskaźnikach są dozwolone.

8. Lambdy – nowy sposób definiowania funkcji

Lambdy są nowym sposobem definiowania funkcji w języku C++. Są to funkcje bezimienne, które można wykorzystać przy różnego rodzaju algorytmach, potrzebujących funkcji orzekającej [3]. Składnia:

```
[](argumenty wywołania)->typ_zwracany{ ciało funkcji}
```

Przykładowo, jeśli jakiś algorytm potrzebuje funkcji sprawdzającej czy określona wartość jest podzielna przez 3, to zamiast definiować funkcję w innym miejscu programu, a następnie wstawiać jej wywołanie do algorytmu możemy wstawić bezpośrednio funkcję typu lambda [2]:

```
#include<iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main(){
vector<int> A={1, 2, 4, 6, 7};
vector<int> ::iterator pozycja;
pozycja = find_if(A.begin(), A.end(), [](const int& a)->int{
return (a%3==0); });
…
}
```

Możliwe jest także ustalenie dostępu funkcji lambda do zmiennych spoza funkcji. I tak np. aby funkcja miała dostęp do wszystkich zmiennych (dostęp tylko do odczytu), należy wstawić pomiędzy nawiasy kwadratowe znak równości:

```
[=]()->typ{ ... }
```

Aby pracować na referencjach do zmiennych spoza zakresu funkcji stawiamy pomiędzy nawiasy kwadratowe znak ampersand:

```
[&]()->typ{ ... }
```

Istnieje także możliwość wyboru, które zmienne będą odebrane jako referencje, a które jako kopie:

```
[$zmienna_1, zmienna_2]()->typ{ . . . }
//zmienna_1 odebrana przez referencję, zmienna_2 odebrana //jako kopia wartości
```

9. Mocniejsze typy wyliczeniowe – enumclass

Standard C++11 wprowadza mocniejsze typy wyliczeniowe[2], które to eliminują problemy, z którymi spotykają się programiści C++. Pierwszym z nich jest niejawna konwersja składowych typu **enum** na typ **int**:

```
#include <iostream>
using namespace std;

int main(){
        enum typ_1{samochod,rower,autobus};
        int x=typeOne::autobus;// możliwe przy zwykłym
        //typie wyliczeniowym
        cout<<x<<endl;
        enum class typ_2{czarny,czerwony,biały};
        //int y=typ_2::czerwony;// błąd!!!
        int y=(int)typ_2::czerwony;
        cout<<y<<endl;
}
```

Używając „starego" typu wyliczeniowego należy uważać na niejawne konwersje. Nowy typ nie pozwala nam na niejawne konwersje, trzeba zaznaczyć kompilatorowi chęć dokonania takiej konwersji, w innym wypadku kończy się to błędem kompilacji.

Drugim problemem rozwiązanym za pomocą składni **enumclass** jest kolidowanie nazw składników:

```
#include <iostream>
using namespace std;

int main(){
        // enum typ_1{czerwony,niebieski}; błąd!!!
        // enum typ_2{czerwony,żółty}; błąd!!!
        enum class KlasaEnum_1{zero,jeden,dwa};
        enum class KlasaEnum_2{zero,jeden,dwa};
        }
```

W zwykłym typie **enum**, w powyższym przypadku gdyby usunąć komentarze składniki czerwony typów wyliczeniowych kolidowałyby ze sobą powodując błąd

kompilacji. Dzięki typowi `enumclass` można uniknąć wzajemnych zależności pomiędzy typami wyliczeniowymi.

10. Podsumowanie

Przyglądając się nowościom wprowadzonym w języku C++ można dostrzec zmiany zarówno, jeśli chodzi o tak proste rzeczy jak drobna zmiana w wyrażeniach szablonowych czy też bardziej zaawansowane jak wprowadzenie funkcji lambda. Niektóre ułatwiają programowanie bardziej a inne mniej, mimo wszystko warto mieć ogólny obraz większości z nich, gdyż część z nich na pewno wejdzie do powszechnego użytku już niebawem.

Bibliografia

[1] Josuttis N. M., C++ Biblioteka Standardowa, Gliwice, Helion, 2014.
[2] Staszewski A., C++11. Nowy standard. Ćwiczenia, Gliwice, Helion, 2012.
[3] http://www.stroustrup.com/C++11FAQ.html.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 57–64

Ewa Falkiewicz, Beata Siemieńska*, Anna Borek**

*Uniwersytet Technologiczno-Humanistyczny w Radomiu,
**Wojskowa Akademia Techniczna w Warszawie

Podejście systemowe w kwestiach modelowania problemów na szczeblu strategicznym państwa

System approach in the modelling of the problems on the strategic level of the state

Streszczenie

Celem referatu jest pokazanie znaczenia dynamiki systemowej w modelowaniu problemów o charakterze dynamicznym na szczeblu strategicznym państwa oraz wykorzystania jej elementów w procesie symulacji komputerowej. Ponadto zostanie krótko scharakteryzowane podejście systemowe, stosowane przy rozwiązywaniu problemów złożonych, a także przedstawione będą podstawowe założenia dynamiki systemowej. Dodatkowo przedstawiona będzie aplikacja Vensim PLE, jako przykład programu symulacyjnego o charakterze edukacyjnym, a także pokazany prosty model symulacyjny, zbudowany na podstawie tego pakietu, imitujący epidemię ptasiej grypy.

Słowa kluczowe: dynamika systemowa, model matematyczny, symulacja komputerowa, modelowanie złożonych systemów, Vensim PLE.

Summary

The aim of the following paper is the presentation of the meaning of the system dynamics in modelling of the problems with dynamic features on the strategic level of the state as well as the use of its components in the process of computer simulation. Furthermore, the system approach used while solving complex problems together with basic assumptions of system dynamics are briefly described here. Additionally, Vension PLE application is presented as the example of the educational simulation program. Besides the simple simulation model, based on the mentioned program, imitating the epidemic of bird flu is provided in the papter.

Keywords: system dynamics, mathematical model, computer modeling, modeling of complex system, Vensim PLE

1. Wprowadzenie

Zarządzanie złożonymi procesami na szczeblu strategicznym państwa nie jest rzeczą prostą i wymaga ogromnego zaangażowania całego sztabu ludzi. Do tego, aby cały ten skomplikowany system sprawnie funkcjonował potrzebne są odpowiednie rozwiązania naukowe oraz techniczne.

Nieustanny rozwój nauki oraz techniki stwarza możliwości do wykorzystywania narzędzi komputerowych, wspomagających działania na różnych płaszczyznach funkcjonowania państwa polskiego. Pojawiające się nowe oprogramowania są wykorzystywane przy rozwiązywaniu różnego rodzaju problemów, niejednokrotnie o znaczeniu strategicznym. Podstawą działania takich programów są algorytmy, zapisane na bazie odpowiedniego aparatu matematycznego.

Zmagając się z problemami o znaczeniu strategicznym, ważnymi elementami, które należy koniecznie wziąć pod uwagę przy programowaniu, są: czas oraz optymalne rozwiązania, uzyskane na podstawie analizy wielu kryteriów. Niezależnie od zastosowanej metody, która prowadzi do rozwiązania optymalnego danego problemu, coraz częściej stosuje się dodatkowo elementy symulacji. Są one połączeniem możliwości komputera z rzeczywistymi sytuacjami, a ich celem jest odpowiedź na zasadnicze pytanie: „jak dany problem zachowa się w przyszłości"? [5]

2. Podejście systemowe

W czasach współczesnych potrzeba myślenia i działania systemowego przy rozwiązywaniu rożnego rodzaju problemów strategicznych jest coraz bardziej pożądana. Złożoność i różnorodność dyscyplin naukowych implikują konieczność integracji na różnych płaszczyznach aktywności pracowników naukowych oraz inżynierów. Obserwując otoczenie, można zauważyć, że większość kluczowych problemów posiada wiele powiązanych ze sobą przyczyn oraz skutków. Aby skutecznie rozwiązywać problemy złożone, należy dążyć do opracowania i wdrożenia w życie nowoczesnych rozwiązań systemowych.

Celem takiej praktyki systemowej jest budowanie modeli symulacyjnych. Projektowanie i wdrażanie koncepcji systemowych z wykorzystaniem tychże modeli umożliwia poprawę analizowanych sytuacji, minimalizując przy tym niepożądane skutki uboczne. Wiadomo, że modele towarzyszą ludziom od wielu stuleci i służą im jako narzędzie do uogólniania i upraszczania kluczowych zjawisk, zachodzących w otaczającym ich świecie. Obecnie modelowanie systemowe odbywa się z wykorzystaniem narzędzi informatycznych.

W sytuacjach, kiedy dostępna jest informacja na temat relacji pomiędzy poszczególnymi elementami danego systemu, można tworzyć modele ilościowe. Stwarzają one możliwość przeprowadzenia dokładniejszych badań potencjalnych

przyszłych scenariuszy. Przykładowym podejściem tego typu jest dość znana metoda dynamiki systemowej [4].

3. Dynamika systemowa

3.1. Ogólna charakterystyka

Dynamika systemowa (DS) wykorzystywana jest do modelowania procesów zarządzania obiektami w ujęciu systemowym. Została ona opracowana przez amerykańskiego inżyniera J. W. Forrestera, a jej głównym założeniem jest symulacja interakcji pomiędzy obiektami, które są elementami systemów dynamicznych. Podstawowe założenia tej koncepcji opierają się o takie dyscypliny, jak: tradycyjna teoria zarządzania, cybernetyka oraz symulacja komputerowa. Koncepcja DS stanowi użyteczną bazę narzędziową, służącą do budowania modeli, które mogą wspomagać podejmowanie decyzji w różnych płaszczyznach działalności państwa [6].

Metoda DS opiera się na ilościowej i jakościowej analizie procesów oraz struktur systemów. Elementem jakościowym oceny dynamiki systemu jest analiza pętli dodatnich i ujemnych sprzężeń zwrotnych, natomiast elementem ilościowym jest matematyczny model, składający się z równań logicznych oraz algebraicznych [3].

W ostatnich latach można zauważyć znaczący wzrost zastosowań dynamiki systemowej w strukturach działalności państwa. Dużą częstotliwość wykorzystywania symulacji da się zaobserwować w sektorze logistycznym, produkcyjnym oraz wojskowym. Istota dynamiki systemowej w przypadku wspomnianych sektorów opiera się na modelowaniu symulacyjnym zjawisk w nich zachodzących. Metoda DS wykorzystywana jest tutaj przede wszystkim do analizy problemów, charakteryzujących się dużą ilością współzależności pomiędzy elementami składowymi. Posługiwanie się tą metodą wymaga jednak przestrzegania określonej procedury modelowania.

Elementy dynamiki systemowej z powodzeniem mogłyby być wykorzystywane np. w systemie Zarządzania Kryzysowego. Poszczególne komórki tego systemu, odpowiedzialne za monitorowanie i prognozowanie zagrożeń mogłyby być wspierane przez eksperckie oprogramowania symulacyjne. Programy tego typu, imitując przebieg niekorzystnych zjawisk, pozwoliłyby wyciągać odpowiednie wnioski i sugerowałyby podjęcie konkretnych kroków przeciwdziałania zagrożeniom.

3.2. Algorytm postępowania w procesie symulacyjnym

Proces modelowania symulacyjnego powinien być odpowiednio usystematyzowany. W pierwszym etapie takiego procesu trzeba wyjść od sformułowania problemu, a w następnej kolejności należy wytypować czynniki, które wpływają na ów

problem. Na ich podstawie powstaje konceptualizacja modelu pod postacią schematów: przyczynowo-skutkowego oraz strukturalnego. Wyodrębnione powyżej schematy przedstawiają wzajemne zależności pomiędzy wskazanymi czynnikami.

Kolejnym etapem procesu symulacyjnego jest budowa matematycznego modelu, który opisuje reguły decyzyjne i który jest następnie rozwiązywany w procesie symulacji komputerowej. Uzyskane wyniki symulacji ukazują zachowanie się badanego systemu w czasie i są porównywane z dostępną wiedzą o systemie. Tu następuje ewentualna weryfikacja modelu pod kątem nieprawidłowości do momentu, aż budowany model będzie odzwierciedlał w sposób rzetelny realne zachowanie się danego systemu. Poprawnie zweryfikowany model jest wykorzystywany do wielokrotnych symulacji zjawiska.

Praca związana z budową modelu oraz jego wykorzystaniem odbiega od prostego, liniowego procesu przechodzenia od jednego etapu do następnego. W świecie rzeczywistym można zaobserwować częste powroty do etapów poprzednich. Stąd też istotą DS jest próba zrozumienia zachowania się systemu poprzez analizę układu sprzężeń zwrotnych. Takie podejście stwarza możliwość doskonalszego poznania systemu i wgląd w przyczynowe związki pomiędzy jego elementami w procesie przewidywania [6].

4. Vensim – narzędzie symulacyjne systemów

4.1. Vensim PLE

Jednym z wielu programów przeznaczonych do symulacji modeli dynamicznych jest produkt Vensim firmy Ventana System Inc. Program ten służy do budowy i symulacji dowolnego systemu, którego elementy i relacje można opisać matematycznymi zależnościami. Może on być wykorzystywany zarówno w naukach technicznych, ekonomicznych, społecznych, jak też wielu innych dziedzinach, które posiadają cechy dynamiki z uwzględnieniem sprzężeń zwrotnych.

Jedną z pięciu wersji programu, które są wykorzystywane do modelowania i symulacji jest produkt Vensim PLE, mający zastosowanie edukacyjne i jest on udostępniany za darmo na stronie www.vensim.com. Vensim PLE jest wersją uboższą w stosunku do wersji komercyjnych, lecz jego ograniczenia nie mają znaczącego wpływu na efektywność osiągnięcia celu w warunkach edukacyjnych [5].

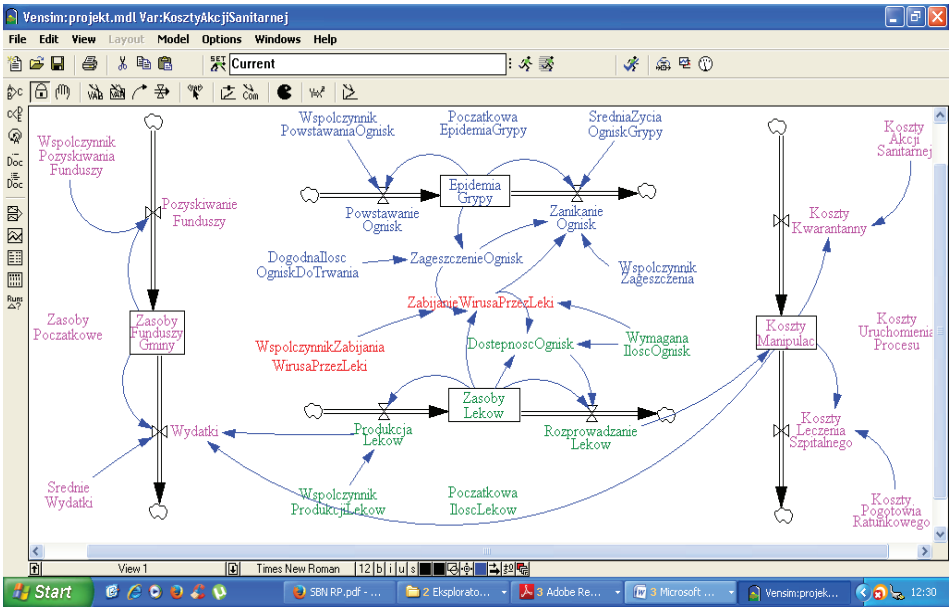
Vensim PLE jest pakietem symulacyjnym, wyposażonym w moduł graficzny. Jest on przeznaczony do wspomagania procesu modelowania w aspekcie DS. Jego zaletą jest to, że w prosty i płynny sposób pozwala budować modele systemowo-dynamiczne, począwszy od schematu przyczynowo-skutkowego, aż do diagramu strukturalnego. Jest to produkt przeznaczony zarówno do celów edukacyjnych, jak i badań własnych.

Pakiet Vensim PLE umożliwia użytkownikowi:

- budowanie nowych oraz modyfikowanie istniejących modeli systemowo-dynamicznych,
- przeprowadzanie eksperymentów symulacyjnych na istniejącym już modelu,
- analizę modelu oraz wizualizację wyników eksperymentów symulacyjnych.

4.2. Przykładowy model symulacyjny

Jak wcześniej wspomniano, profesjonalne modele symulacyjne zaimplementowane na bazie zaawansowanych aplikacji mogą być wykorzystywane np. w komórkach Zarządzania Kryzysowego, gdzie można przewidywać scenariusze potencjalnych zagrożeń. Poniżej na rysunku zamieszczono przykładowy prosty model symulacyjny na bazie narzędzia Vensim PLE, przedstawiający epidemię ptasiej grypy. Cały proces modelowania oraz symulacji tego typu zjawisk jest szczegółowo opisany w [8].

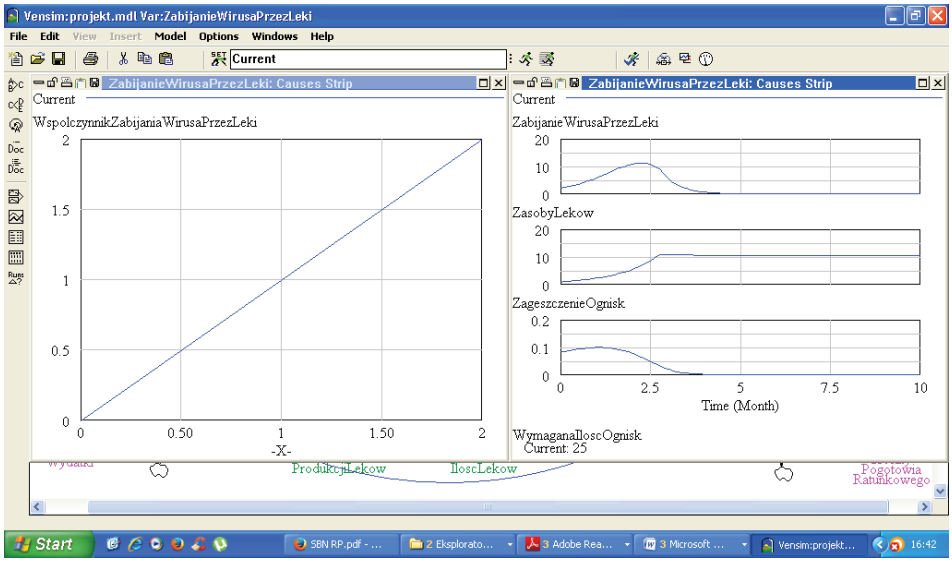


Rys. 1. Schemat strukturalny epidemii ptasiej grypy

Po uprzednim narysowaniu diagramów oraz opisaniu zmiennych równaniami matematycznymi, można przejść wreszcie do procesu symulacyjnego. Zostanie wówczas uruchomiona procedura obliczeniowa, w skutek której zaprojektowany model będzie rozwiązany dla każdej chwili czasu.

Prezentowana aplikacja Vensim PLE jest wyposażona w narzędzia wykorzystywane do analizy strukturalnej modelu, a także w narzędzia służące do analizy

rezultatów symulacji dla zmiennej aktywnej. Te drugie umożliwiają wizualizację wyników eksperymentów symulacyjnych w postaci tabel roboczych oraz wykresów. Poniżej na rysunku zamieszczono wykresy: zmiennej aktywnej „Zabijanie Wirusa Przez Leki" oraz pozostałych zmiennych, od których jest ona zależna [5].



Rys. 2. Wykresy: zmiennej aktywnej oraz zmiennych, od których jest ona uzależniona

Szczegółowe informacje na temat zastosowania metody dynamiki systemowej w modelowaniu złożonych systemów i procesów z wykorzystaniem narzędzia Vensim są przedstawione w [1], a także w [2]. Ponadto inną pozycją godną uwagi, która przedstawia problem od strony teoretycznej jest [7].

5. Inne pakiety wspomagające modelowanie systemowo-dynamiczne

Oprócz znanego już nam pakietu Vensim w celach modelowania systemowo-dynamicznego wykorzystuje się także inny język programowania o nazwie DYNAMO. Początkowo był to język wyższego rzędu niezbyt przyjazny dla użytkownika nieobytego z informatyką. Aktualnie jest on dostosowany do systemu Windows oraz, tak jak inne języki również wyposażony w moduł graficzny, który przeznaczony jest do tworzenia schematów strukturalnych modelowanego problemu.

Na rynku dostępne są także jeszcze inne programy, które wspomagają konstruowanie i rozwiązywanie modeli systemowo-dynamicznych, a które nie są oparte o standard języka DYNAMO. Można do nich zaliczyć dość znane aplikację pod nazwami: IThink oraz Powersim.

Pakiet IThink służy do budowy oraz symulacji procesów z zakresu zarządzania oraz ekonomii, a także może być narzędziem, służącym do rozwiązywania problemów w biznesie. Znajduje on szerokie zastosowanie w elementach zarządzania przedsiębiorstwem, takich jak:

- prognozowanie,
- rozwój organizacyjny,
- planowanie gospodarki zasobami ludzkimi,
- planowanie produkcji,
- reengineering procesów organizacyjnych,
- planowanie strategiczne,
- analizy finansowe.

Natomiast aplikacja Powersim pod względem działania jest podobna do dwóch wcześniej omówionych programów. Różnice natomiast występują w stosowanej tu symbolice oraz w sposobie generowania wykresów i tworzeniu oraz wyglądzie tabel.

Przedstawione powyżej pakiety symulacyjne nie są jedynymi programami, które wspomagają modelowanie systemowo-dynamiczne. Niektóre ośrodki badawcze na świecie, zajmujące się dynamiką systemową stworzyły własne programy symulacyjne. Pełna lista takich ośrodków znajduje się na stronie Towarzystwa Systemowo Dynamicznego pod adresem http://www.albany.edu/cpr/sds [5].

6. Podsumowanie

W niniejszym referacie skoncentrowano się głównie na idei znaczenia dynamiki systemowej, której komponenty mogą być wykorzystywane w procesie modelowania i symulacji skomplikowanych zależności, występujących pomiędzy elementami i podmiotami na szczeblu strategicznym państwa.

Jak pokazano, skuteczność w rozwiązywaniu problemów złożonych tego typu ma swoje podłoże w podejściu systemowym. Wykazano, że niezbędnym środkiem do realizacji rozwiązań systemowych są narzędzia informatyczne. Szczególną rolę przypisano tutaj modelom symulacyjnym, tworzonym na podstawie aplikacji, które opierają się o zasady dynamiki systemowej.

Dla przykładu zaprezentowano modelowanie zjawiska o charakterze dynamicznym, jakim jest np. epidemia ptasiej grypy, wykorzystując do tego edukacyjny pakiet Vensim PLE. Następnie dokonano wstępnej i krótkiej analizy niektórych parametrów oraz zmiennych tego modelu, wykorzystując dostępne narzędzia, którymi dysponuje wspomniana aplikacja.

W niniejszym rozdziale omówiono również inne programy symulacyjne, takie jak: DYNAMO, IThink, czy Powersim, podając ich krótką charakterystykę oraz sposoby i dziedziny, w których mogą być one zastosowane. Dodatkowo wspomniano też o ośrodkach badawczych, które są twórcami własnych środowisk symulacyjnych.

Bibliografia

[1] Hoffmann R. Protasowicki T. Metoda dynamiki systemowej w modelowaniu złożonych systemów i procesów, http://www.isi.wat.edu.pl/sites/default/files/Biuletyn/Nr_12_2013/19_28_met_dyn_rhoffmann_tprotasowicki_12_2013.pdf.

[2] Hoffmann R. Protasowicki T. Modelowanie pola walki z zastosowaniem koncepcji dynamiki systemowej, http://www.isi.wat.edu.pl/sites/default/files/Biuletyn/Nr_12_2013/29_34_mod_pola_walki_rhoffmann_tprotasowicki_12_2013.pdf.

[3] Kasperska E. Analiza procesów i struktur obiektów ekonomicznych w oparciu o metodę Dynamiki Systemowej oraz język symulacyjny DYNAMO .http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ekon-element-000171309233.

[4] Kronenberg J. Bergier T. Wyzwania zrównoważonego rozwoju w Polsce, https://books.google.pl/books?id=jKM7bAXezf4C&pg=PA49&lpg=PA49&dq=co+to+jest+dynamika+systemowa.

[5] Krupa K. Modelowanie symulacja i prognozowanie. Systemy ciągłe, WNT, Warszawa 2009.

[6] Łatuszyńska M. Metoda dynamiki systemowej we wspomaganiu rozwiązywania problemów opieki zdrowotnej, http://rocznikikae.sgh.waw.pl/p/roczniki_kae_z25_09.pdf.

[7] Żukowski P. Podstawy budowy modelu dynamiki systemu zarządzania oraz jego symulacja w organizacji gospodarczej na podstawie metodologii dynamiki systemów J. W. Forrestera, http://webcache.googleusercontent.com/search?q=cache:ATScmSZKdzoJ:cejsh.icm.edu.pl.

[8] http://www.pwsz.pl/testy/attachments/article/857/Pakiet%20symulacyjny%20Vensim%20-%20opis.pdf.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 65–70

Monika Maj, Anna Borek*, Beata Siemieńska**

*Wojskowa Akademia Techniczna w Warszawie
**Uniwersytet Technologiczno-Humanistyczny w Radomiu

Wymagania dla systemu wspomagania decyzji w zarządzaniu kryzysowym

The demands for information technology systems supporting the decisions in crisis management

Streszczenie
Niniejsze opracowanie przedstawia teoretyczne rozważania nad specyfikacją systemu wspomagania decyzji odpowiadającego konieczności efektywnego wykorzystania zaplecza informacyjnego utrzymywanego na potrzeby systemu zarządzania kryzysowego. Istotność poruszanej problematyki definiuje charakter sytuacji, w jakich uruchamiane są procedury zarządzani kryzysowego (element presji czasu, zaskoczenia, nieprzewidywalności zagrożeń, konieczność szybkiego podejmowania decyzji).

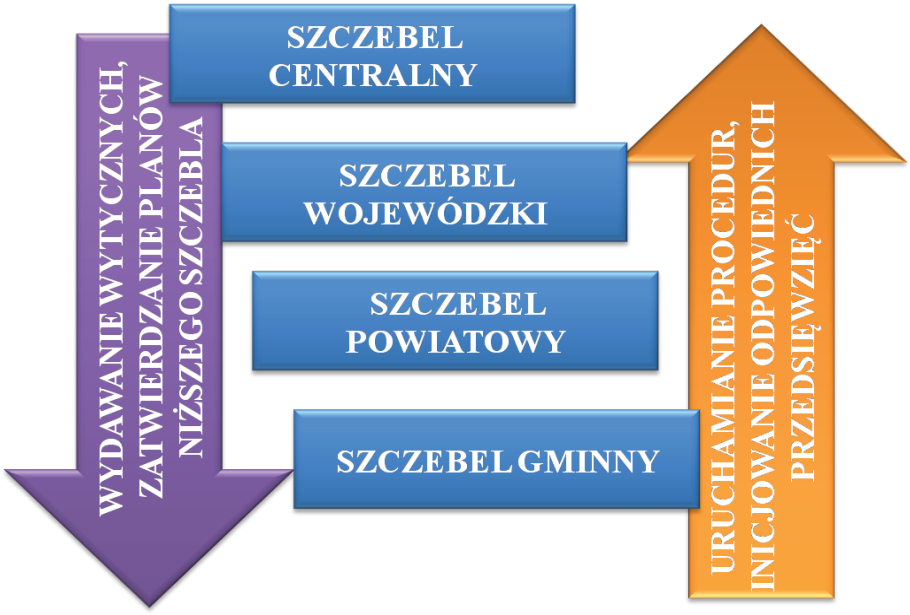
Summary
The following paper presents the theoretical considerations on the specification of the information technology system of the decision support. That corresponds with the necessity of the effective use of IT background maintained for the needs of crisis management system. The significance of the mentioned issues are defined by the nature of phenomena in which the procedures of crisis management are introduced (the element of time limit, astonishment, unpredictability of threats, the necessity of quick decision making).

Słowa kluczowe: system zarządzania kryzysowego, wspomaganie decyzji, potrzeby informacyjne decydentów systemu

Keywords: crisis management system, decision support system, system information needs of decision-makers

1. System zarządzania kryzysowego

Zarządzanie kryzysowe definiowane ustawowo odnosi się do działalności organów administracji państwowej, stanowiącej element zintegrowanego systemu kierowania bezpieczeństwem państwa. Idea opiera się na zorganizowaniu systemu pozwalającego w sposób optymalny wykorzystywać posiadane zasoby, siły i środki, celem zapobiegania sytuacjom negatywnie wpływającym na bezpieczeństwo państwa i jego obywateli, przeciwdziałania im, reagowania w momencie ich wystąpienia, a także likwidacji skutków i przywracaniu zasobów systemu do stanu sprzed ich wystąpienia [1]. Przy czym zgodnie z prawnie przyjętymi założeniami, funkcjonowanie systemu zarządzania kryzysowego opiera się na zasadzie odgórnego planowania i oddolnego inicjowania i uruchamiania niezbędnych procedur, co w sposób uproszczony ukazuje rys. 1.



Rys. 1. Uproszczony model funkcjonowania systemu zarządzania kryzysowego w Polsce

W niniejszej pracy przyjmując założenie, iż system zarządzania kryzysowego tworzą jego wszystkie elementy, poszczególne szczeble wchodzące w skład systemu, traktujemy, jako podsystemy.

Zgodnie z art. 21 ustawy o zarządzaniu kryzysowym odpowiedzialność za podjęcie niezbędnych działań spoczywa na organie, który jako pierwszy otrzymał informację o zaistniałym zdarzeniu. Mając na względzie zadania organów samorządów terytorialnych, zwyczajowo przyjąć należy, iż w przypadku zda-

rzeń o mniejszej skali występujących na obszarze jednej gminy – na gminach, a w przypadku występowania zdarzeń na obszarze więcej niż jednej gminy na organach powiatowych, spoczywać będzie ciężar kierowania działaniami z zakresu zarządzania kryzysowego.

Wyniki prac Najwyższej Izby Kontroli dotyczące przeglądu przygotowania systemu ochrony ludności przed klęskami żywiołowymi oraz sytuacjami kryzysowymi wskazały na braki i problemy związane z organizacją systemu na wszystkich szczeblach zarządzania kryzysowego, wpływające w sposób znaczący na spadek wiarygodności efektywnego działania w sytuacjach trudnych [2]. Celem zwiększenia niezawodności systemu i integralności jego elementów warto zastanowić się nad możliwością wykorzystania odpowiednio przygotowanego systemu wspomagania decyzji, opartego na optymalnie zorganizowanych bazach danych, pozwalających usprawnić proces decyzyjny decydentów systemu. Mając na względzie błędy występujące w systemie oraz presję czasu wywieraną w sytuacjach nadzwyczajnych, uruchamianie procedur bez posiadania niezbędnych informacji oraz doskonałej znajomości prawnie przyjętych rozwiązań może w sposób znaczący obniżyć skuteczność podejmowanych przedsięwzięć, to z kolei bezpośrednio wpływa na (nie)bezpieczeństwo ludności obszaru dotkniętego występowaniem określonego rodzaju ryzyka.

2. Informacyjne wymagania systemu wspomagania decyzji

System wspomagania decyzji odpowiadający potrzebom informacyjnym podmiotów odpowiedzialnych za zarządzanie kryzysowe powinien gromadzić i przetwarzać dane dotyczące:

- Charakterystyki obszaru podlegającego ochronie, w tym przede wszystkim zawierać dane demograficzne, topograficzne i ekonomiczne, a także informacje odnoszące się do otoczenia podsystemu (bazy danych gromadzone i przetwarzane w ramach funkcjonowania podsystemu);
- Charakterystyki występujących ryzyk i zagrożeń z oszacowaniem prawdopodobieństwa ich wystąpienia, skutków i czasu trwania (bazy danych gromadzone i przetwarzane w ramach funkcjonowania podsystemu, dane gromadzone i przetwarzane przez wyspecjalizowane instytucje);
- Dostępność sił i środków (bazy danych gromadzone i przetwarzane w ramach funkcjonowania podsystemu, dane gromadzone i przetwarzane przez wyspecjalizowane instytucje):
  - Ludzkich (służby, straże, inspekcje);
  - Materiałowych (stany magazynów, utrzymywane zasoby, ewidencję prowadzonych działalności na obszarze samorządu terytorialnego);
  - Finansowych.

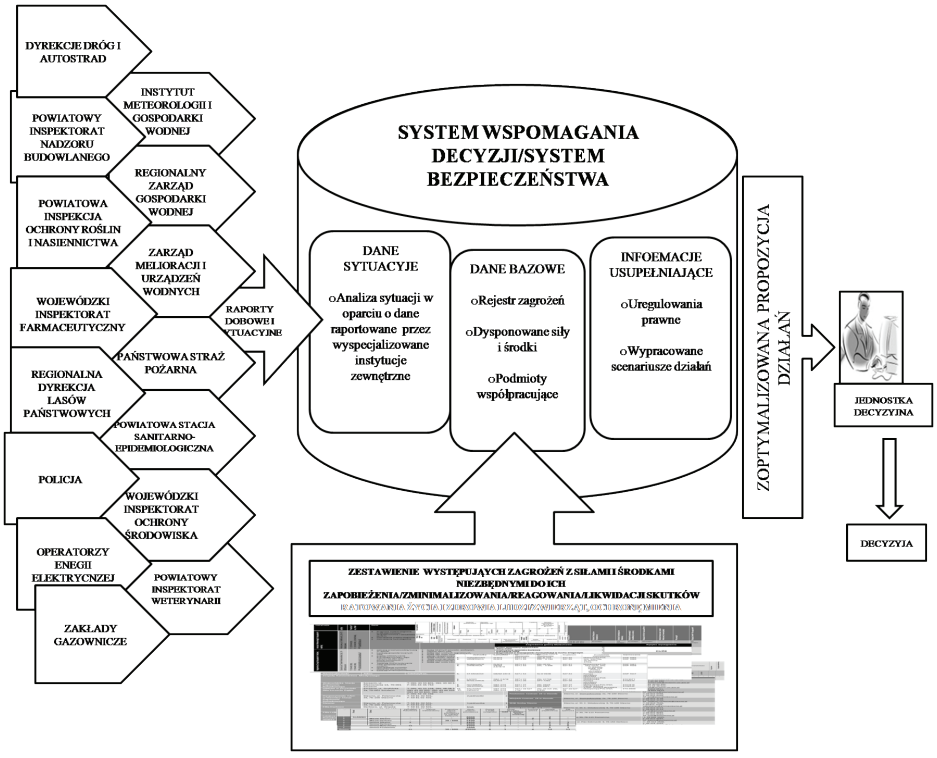
Istotą kwestię stanowią także informacje niezbędne dla osoby kierującej działaniami odnoszące się do:

- Prawnego uregulowania działań zgodnie z przyjętymi normami, standardami, wytycznymi itp.;
- Wykorzystania wypracowanych procedur i wariantów działań mieszczących się w ramach planowania z zakresu zarządzania kryzysowego i obrony cywilnej;
- Wskazania kompetencji użytkowników systemu, ich roli i podległości.

3. Model systemu wspomagania decyzji

Odpowiadając wyżej nakreślonym potrzebom informacyjnym, system informatyczny winien składać się z trzech odpowiednio zasilanych i skorelowanych baz danych (Rys.2):

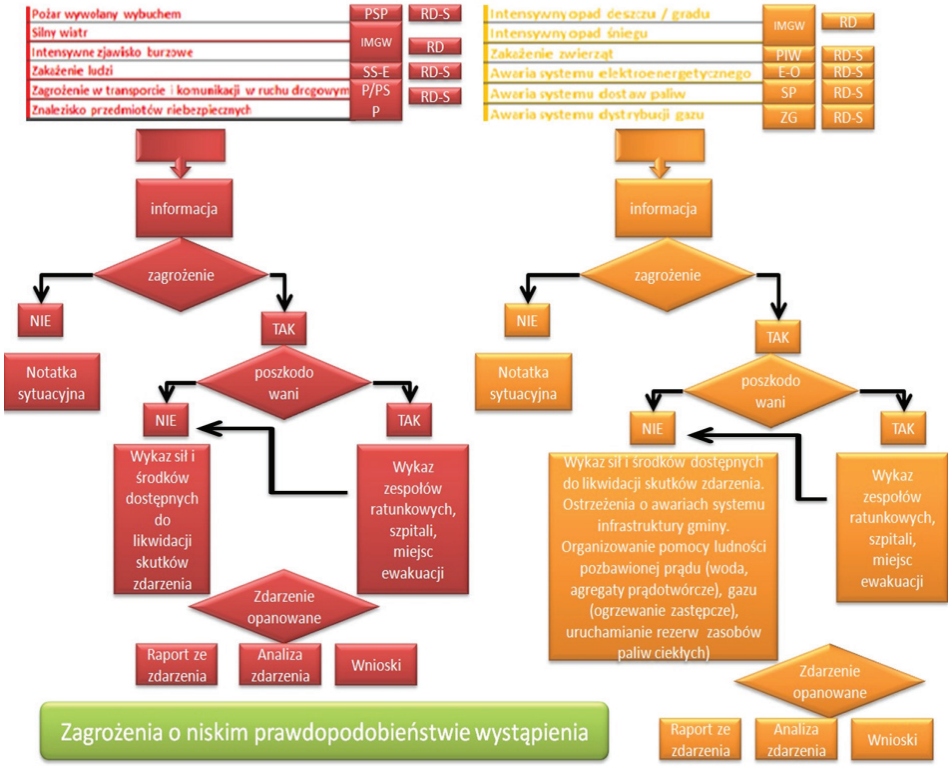
- Danych sytuacyjnych, zasilanych z zewnętrznych jednostek wyspecjalizowanych w monitorowaniu i analizowaniu określonych zjawisk;
- Danych bazowych, danych gromadzonych i aktualizowanych przez organy administracji terenowej;
- Danych uzupełniających zawierających prawne usankcjonowanie działań.



Rys. 2. Model systemu wspomagania decyzji

Istotną kwestię stanowi konieczność współpracy w ramach wymiany informacji sytuacyjnych z instytucjami wyspecjalizowanymi w monitorowaniu poszczególnych zjawisk, co wymaga wcześniejszego określenia częstotliwości i rodzaju raportowanych informacji, oraz zawężenia obszaru danych niezbędnych dla decydentów systemu. Przykładowo Instytut Meteorologii i Gospodarki Wodnej statutowo zobligowany jest do monitorowania sytuacji, zbierania, przetwarzania i analizowania danych dotyczących zjawisk meteorologicznych, atmosferycznych i hydrologicznych, a także opracowywania i udostępniania prognoz i ostrzeżeń [3], dla potrzeb systemu określimy wymóg raportowania stałego w postaci prognoz dostarczanych, co 24h/72h, oraz sytuacyjnego – raporty opadowe, prognozy specjalne.

Odgórne ustalenie sposoby zasilany danymi zniweluje możliwość wystąpienia „bałaganu informacyjnego", a zastosowanie odpowiednich rozwiązań informatycznych zmniejszy prawdopodobieństwo identyfikacji błędów związanych z ludzkim wymiarem funkcjonowania systemu.



Rys 3. Uproszczony model interfejsu użytkownika

Najsłabszym elementem każdego systemu informatycznego jest człowiek, dlatego też dla spełnienia stawianych systemowi wspomagania decyzji wymagań, panel użytkownika winien opierać się na przejrzystym i nieskomplikowanym algorytmie ułatwiającym korzystanie z systemu różnym grupom użytkowników, minimalizując w ten sposób margines błędu wynikający z użytkowania systemu.

4. Podsumowanie

Dynamika rozwoju zdarzeń w sytuacjach trudnych, wymusza na decydentach systemu podejmowania szybkich i skutecznych przedsięwzięć mających na celu zminimalizowanie skutków występujących zdarzeń. Wieloaspektowość, wielowymiarowość i szeroki zakres rodzajowy potencjalnych zagrożeń generuje wątpliwości, co do możliwości przygotowania personelu jednostek samorządów terytorialnych niższego szczebla, do efektywnego kierowania działaniami w sytuacjach kryzysowych (braki kadrowe oraz zaplecze merytoryczne). Wdrożenie systemów wspomagania decyzji decydentów systemu opartych na wymaganiach teoretycznych wynikających ze specyfiki zagadnień zarządzania kryzysowego, bezpośrednio wpłynęłoby, na wzrost jakości całego systemu.

Mając na uwadze kwestie finansowania budowy omawianego systemu wspomagania decyzji, warto przeanalizować zyski, jakie niosłoby jego wdrożenie. Z jednej strony samorządy, które często zmuszone są uruchamiać procedury zarządzania kryzysowego dzięki systemowi usprawniłyby swoje działanie, przez eliminację błędów i skrócenie czasu od wystąpienia zagrożenia do podjęcia decyzji, z drugiej strony w samorządach, w których procedury zarządzania wdrażane są bardzo rzadko bądź wcale, system zapewniłby ich sprawne uruchomienie w razie wyniknięcia takiej potrzeby.

Bibliografia

[1] Ustawa o zarządzaniu kryzysowym z dn. 26 kwietnia 2007 r. z późniejszymi zmianami.

[2] Informacja o wynikach kontroli NIK, Przygotowanie systemu ochrony ludności przed klęskami żywiołowymi i sytuacjami kryzysowymi, https://www.nik.gov.pl/plik/id,5308,vp,6885.pdf.

[3] http://www.imgw.pl/index.php?option=com_content&view=article&id=58&Itemid=18.

Zeszyty Naukowe WSEI seria: TRANSPORT I INFORMATYKA, 5(1/2015), s. 71–80

***Jan Wrona, **Rafał Wrona**

*Wyższa Szkoła Ekonomii i Innowacji w Lublinie
**Politechnika Lubelska

Straty mechaniczne silnika spalinowego w nieustalonych warunkach jego pracy

Combustion engine mechanical losess in transient operating conditions

Streszczenie

W opracowaniu podjęto próbę analizy i oceny strat mechanicznych silnika spalinowego w nieustalonych warunkach jego pracy, które stanowią na przykład w ruchu miejskim ponad 80% czasu jego pracy. Ocenie poddano wpływ na straty mechaniczne silnika spalinowego takich parametrów jak: prędkość obrotowa wału korbowego, obciążenie czy stan cieplny silnika. Wykorzystano różne metody badawcze i analityczne w celu interpretacji graficznej poziomu strat mechanicznych dla różnych silników spalinowych pracujących w zmiennych warunkach pracy, które ilustrowano wykreślnie.

Summary

In the foregoing study an attempt was made at analyzing and assessing the combustion engine mechanical losses in transient operating conditions, which, in. e. constitute more than 80% of its service life. The effect of such parameters as crankshaft rotational speed, load or thermal state of the engine upon the mechanical losses of combustion engine was assessed. Different research and analysis methods were used for graphic interpretation of the mechanical losses level for various combustion engines working in transient operating conditions, which were illustrated graphically.

Słowa kluczowe: straty mechaniczne, silnik spalinowy, nieustalone warunki pracy

Keywords: mechanical losses, combustion engine, transient operating conditions

1. Wstęp

Ciągłe dążenie do uzyskiwania coraz większych osiągów w zakresie mocy jednostkowych i minimalizacji zużycia paliwa, sprzyja rozwojowi konstrukcji i powoduje postęp w dziedzinie eksploatacji tłokowych silników spalinowych. Opracowanie nowych konstrukcji i modyfikacja istniejących, wymaga szerokiej informacji o wpływie warunków użytkowania na trwałość, ekonomię i ekologię pracy silników spalinowych. Znajomość procesów zachodzących w silniku, podczas jego pracy w warunkach rzeczywistych jest zagadnieniem istotnym z uwagi na fakt, że użytkowanie silnika samochodowego i ciągnikowego charakteryzuje się dużą zmiennością prędkości obrotowej wału korbowego i obciążeń w różnych jego stanach termicznych i zmiennym dawkowaniu paliwa. Można przyjąć, że 80-90 % całego czasu pracy silnika w ruchu miejskim przypada na stany nieustalone [1,2]. Podstawowe parametry pracy silnika spalinowego jak moment obrotowy, zużycie paliwa oraz emisja toksycznych składników spalin w nieustalonych warunkach różnią się od wyników uzyskiwanych w warunkach normatywnych (ustalonych). Zasadne staje się zatem uwzględnienie na etapie opracowań nowych konstrukcji lub wprowadzania zmian modyfikacyjnych konstrukcji już istniejących, rzeczywistych warunków użytkowania silników spalinowych.

2. Teoretyczne i rzeczywiste warunki pracy silnika

Ustalonym stanem pracy silnika określa się taki stan, w którym kolejne przebiegi pracy (cykle robocze) w każdym z cylindrów są identyczne [1] czyli:

$$p(V) = idem$$
$$p(t) = idem$$

gdzie: p- ciśnienie w cylindrze, V – chwilowa objętość cylindra roboczego, t – czas.

Pierwszy z wyżej wymienionych warunków oznacza, że praca indykowana kolejnych przebiegów pracy jest jednakowa, natomiast drugi oznacza, że przebieg zmian ciśnienia w czasie dla kolejnych przebiegów pracy jest jednakowy. W celu uzyskania stanu ustalonego powyższe warunki muszą być spełnione jednocześnie. Tymczasem podczas rzeczywistych warunków pracy silnika, z powodu istnienia cech konstrukcyjno-użytkowych, występuje zjawisko niepowtarzalności kolejnych przebiegów pracy, a także istnieje niepowtarzalność kolejnych przebiegów pracy w poszczególnych cylindrach silnika spalinowego. Różnice te zależą od cech konstrukcyjnych silnika, stopnia obciążenia, prędkości obrotowej wału korbowego, stanu technicznego, danych regulacyjnych itp. W rzeczywistych warunkach pracy silnika spotykane jest zjawisko „taktowania" aparatury wtrysko-

wej (brak wtrysku) lub „zaniku" zapłonu w poszczególnych cylindrach, co jest charakterystyce dla silników dwusuwowych. Powyższe fakty potwierdzają tezę, że ustalone warunki pracy silnika spalinowego są trudne do osiągnięcia w rzeczywistych warunkach jego pracy.

W literaturze technicznej [1, 2, 3] można spotkać inne kryteria warunkujące stan pracy silnika spalinowego np.:

$$\frac{dM}{dt} = 0, \qquad \frac{dn}{dt} = 0, \quad \frac{dQ}{dt} = 0 \tag{1}$$

gdzie: M – moment obrotowy wału korbowego silnika, n – prędkość obrotowa wału korbowego, Q – ciepło doprowadzone z paliwem, t – czas.

Należy zauważyć, że określony wyżej zespól warunków (1) nie może być w warunkach rzeczywistych spełniony, gdyż paliwo a więc i ciepło doprowadzone jest do przestrzeni roboczej cylindra w postaci dawek paliwa, zaś chwilowa wartość momentu obrotowego i prędkości obrotowej zawsze ulega zmianie w zakresie jednego obrotu wału korbowego. Amplituda tych zmian zależy od układu i liczby cylindrów, obciążenia i prędkości obrotowej oraz masowego momentu bezwładności koła zamachowego i związanych z nim mas wirujących. Dla potrzeb praktyki inżynierskiej kryterium stałości warunków pracy silnika może być zdefiniowana w następujący sposób:

$$M = \text{const}$$
$$n = \text{const}$$
$$q = \text{const}$$

gdzie: M, n i q – średnie wartości momentu obrotowego, prędkości obrotowej wału korbowego i dawki paliwa przypadającej na jeden przebieg pracy silnika.

Niekiedy ze względów pomiarowych warunek q = const, zastępuje się warunkiem φ = const, gdzie φ – kąt położenia dźwigni sterującej dawką paliwa. Określone wyżej kryteria stałości warunków pracy silnika należy uzupełnić jeszcze jednym warunkiem, że T = const. gdzie: T – temperatura określająca stan cieplny silnika.

W praktyce może to być temperatura cieczy chłodzącej i olejku smarującego. Zatem gdyby choć jedno z wyżej wymienionych warunków pracy silnika nie było spełnione, to silnik pracowałby w warunkach nieustalonych. Zmiany warunków pracy silnika mogą więc być określone przez zmienność:

- momentu oporów odbiornika mocy (oporów ruchu),
- dawkowania paliwa,
- stanu cieplnego silnika,
- kombinacji dwóch lub wszystkich trzech wymienionych wyżej parametrów.

W szczególnych przypadkach zmiana warunków pracy silnika może być spowodowana także innymi przyczynami jak: zmianą kąta wyprzedzenia wtrysku (zapłonu), otwarciem lub zamknięciem systemu doładowującego lub zmianą położenia zaworu sterującego recyrkulacją spalin itp. W praktyce istnieje więc możliwość tworzenia nieskończenie dużo nieustalonych warunków pracy silnika spalinowego, co stwarza możliwość ich wielokierunkowej klasyfikacji. Najczęściej stosowany jest podział stanów nieustalonych ze względu na zmianę prędkości obrotowej wału korbowego silnika, przy czym rozpatruje się:

a) zwiększenie prędkości obrotowej /Δn/ > 0

możliwe gdy dφ > o

dφ < 0

dφ = 0 jednocześnie M – Mop > 0

gdzie: φ – kąt położenia dźwigni regulującej dawkę paliwa, M – chwilowy moment obrotowy silnika, Mop – chwilowa wartość momentu oporów silnika.

b) prędkość obrotowa wału korbowego silnika nie ulega zmianie / Δn / = 0

jest możliwe gdy: dφ > 0

dφ < 0 a jednocześnie M – Mop = 0

(warunek dφ = 0 odpowiada stanowi ustalonemu)

c) prędkość obrotowa wału korbowego zmniejsza się / Δn/ < 0

realizowany jest gdy: dφ > 0

dφ = 0

dφ < 0 a jednocześnie M – Mop < 0

4. Nieustalone warunki pracy silnika spalinowego

Zmiany wskaźników indykowanych i efektywnych silnika spalinowego podczas jego pracy w warunkach nieustalonych mogą być wyrażone w funkcji prędkości obrotowej wału korbowego, obciążenia lub stanu cieplnego silnika, które ulegają zmianie w poszczególnych cyklach roboczych. Analiza pracy silnika o zapłonie samoczynnym w warunkach nieustalonych pozwala wyodrębnić trzy grupy zmian jego parametrów roboczych:

- zmiany energii kinetycznej przemieszczających się mas w ruchu postępowym i obrotowym,
- zmiany własności fizycznych i termicznych czynnika roboczego podczas napełnienia i spalania,
- zmiany oporów mechanicznych wynikających ze zmian parametrów kinematycznych, obciążenia i zjawisk trybologicznych węzłów ciernych.

Główne parametry pracy silnika spalinowego w warunkach nieustalonych oznacza się indeksem „n" na przykład: moment efektywny $M_{e\,n}$, moc indykowana

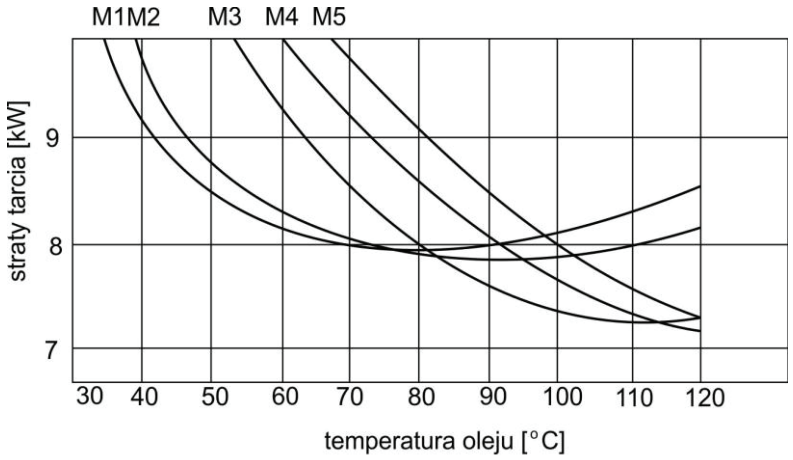
N_{in},, średnie ciśnienie indykowane p_{in} czy moc efektywna N_{en}.. Powyższe parametry można przedstawić w postaci funkcji czterech wartości tj. prędkości kątowej wału korbowego „ω", przyspieszenia kątowego $\varepsilon = \frac{d\omega}{dt}$, stopnia zasilania silnika φ i prędkości zmian stopnia zasilania silnika $\mu = \frac{d\varphi}{dt}$. Po wyznaczeniu różniczek zupełnych wyżej wymienionych funkcji i przeprowadzeniu odpowiednich przekształceń matematycznych można otrzymać ogólne równania opisujące stan pracy silnika w warunkach nieustalonych. Otrzymane w powyższy sposób równania, mogą być bazą dla określenia matematycznego ujęcia warunków pracy stanów nieustalonych dowolnego silnika spalinowego. Równanie opisujące straty mechaniczne silnika pracującego w warunkach nieustalonych ma postać (2).

$$N_{tn} = N_{tu} + (b_1 - a_1)\, n\, \frac{d\varphi}{dt} + (l_1 - c_1)\, n\, \frac{d\varphi}{dt} \qquad (2)$$

gdzie: b_1, a_1, l_1 i c_1 –parametry uwzględniające cechy konstrukcyjne i eksploatacyjne silnika.

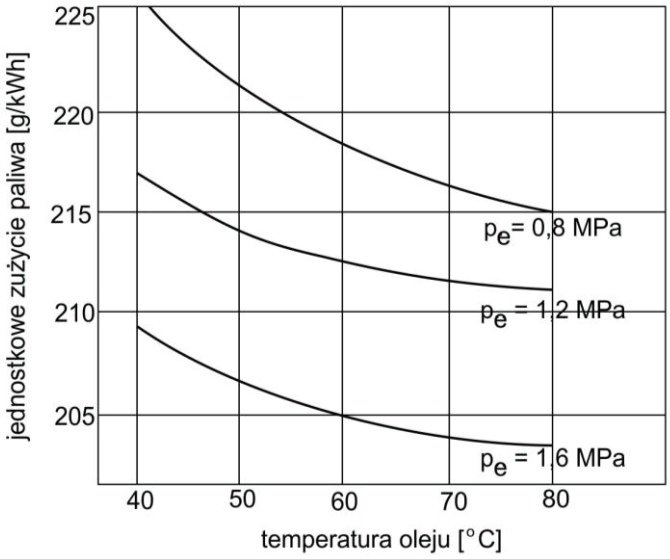
Trudności w określaniu konkretnych zależności stoją na przeszkodzie praktycznego wykorzystania równań przytoczonych wyżej. Nadto duża ilość wariantów nieustalonych warunków pracy silnika spalinowego oraz trudności w matematycznym zapisie poszczególnych stanów pracy silnika, skłoniły zainteresowanych do analizy wybranych warunków pracy silnika odpowiadających praktyce eksploatacyjnej. Powoduje to ukierunkowaną działalność badawczą pod kątem analizy wpływu nieustalonych warunków pracy silnika na jego parametry dynamiczne ekonomiczne i ekologiczne. Biorąc pod uwagę ocenę strat mechanicznych silnika podczas jego pracy w warunkach nieustalonych, najbardziej celowe jest przeprowadzenie analizy uwzględniającej: przyspieszanie prędkości obrotowej wału korbowego obciążonego silnika, wpływu stanu termicznego silnika (na przykład podczas nagrzewania się), wpływu zmian oporów obciążonego osprzętu silnikowego czy wpływu stopnia doładowania. Przytaczany w literaturze fachowej bilans cieplny silnika odnosi się do ustalonych warunków jego pracy i określa ilość ciepła doprowadzonego z czynnikiem roboczym oraz rozdział strat odprowadzanych przez układ wydechowy czynnik chłodzący i smarujący oraz straty oporów ruchu, a także ciepło zamienione na pracę efektywną. W warunkach nieustalonych udziały poszczególnych strat zarówno w ujęciu ilościowym jak i jakościowym ulegają zmianie [3,4,5]. Odnosi się to także do strat mechanicznych silnika, które w wyniku na przykład wzrostu temperatury pracy silnika ulegają zmianie. Spośród wymienianych czynników, największy wpływ na opory tarcia ma tem-

peratura i lepkość czynnika smarującego. Na rys.1 zilustrowano zależność strat tarcia dla różnych olejów w funkcji ich temperatury.



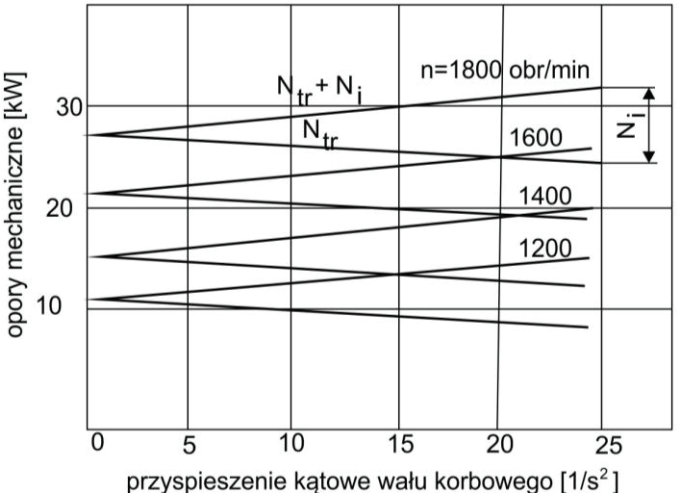
Rys. 1. Wpływ temperatury różnych olejów smarujących na straty tarcia.

Z charakteru przebiegu krzywych wynika, że każdy olej posiada określona temperaturę dla której istnieją najmniejsze straty tarcia. Zmiana temperatury oleju zarówno w dół jak i w górę powoduje istotną zmianę oporów tarcia, przy czy szczególne znaczenie ma obniżenie temperatury oleju poniżej 60^0C. Zmniejszenie strat tarcia ma także skutki ekonomiczne w pracy silnika, zwłaszcza jeśli silnik pracuje ze znacznym obciążeniem. Na rys. 2 zilustrowano wpływ temperatury oleju silnikowego na jednostkowe zużycie paliwa silnika doładowanego w różnym stopniu przy stałej temperaturze czynnika chłodzącego.



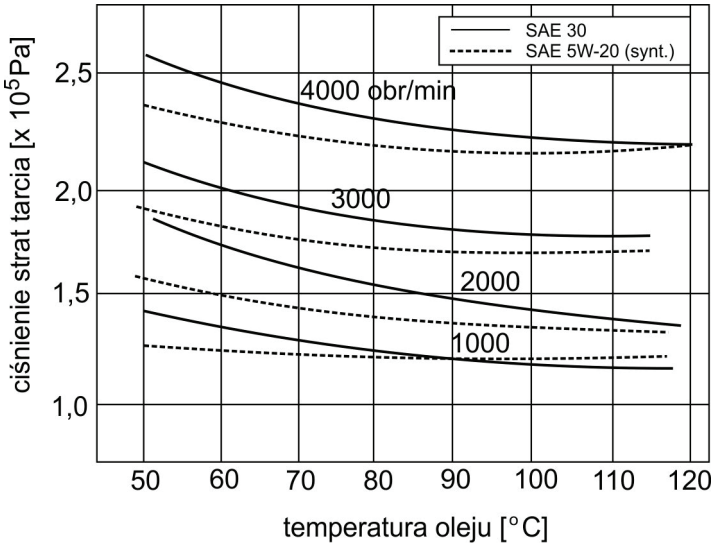
Rys. 2. Wpływ temperatury oleju silnikowego na jednostkowe zużycie paliwa dla różnych stanów wartości ciśnienia efektywnego

Wraz ze wzrostem temperatury czynnika smarującego zmniejsza się jednostkowe zużycie paliwa, przy czym powyższe zmiany w g_e znacząco maleją, jeśli wzrasta średnie ciśnienie efektywne silnika, co ma miejsce podczas doładowania silnika spalinowego. Na kolejnym rys. 3 zilustrowano wpływ przyspieszenia kątowego wału korbowego na poziom oporów mechanicznych silnika posiadającego różne prędkości obrotowe wału korbowego. W miarę wzrostu prędkości obrotowej silnika rosną straty mechaniczne, natomiast w niewielkim stopniu zmieniają się straty mechaniczne, tj. straty maleją ze wzrostem przyspieszenia kątowego wału korbowego.



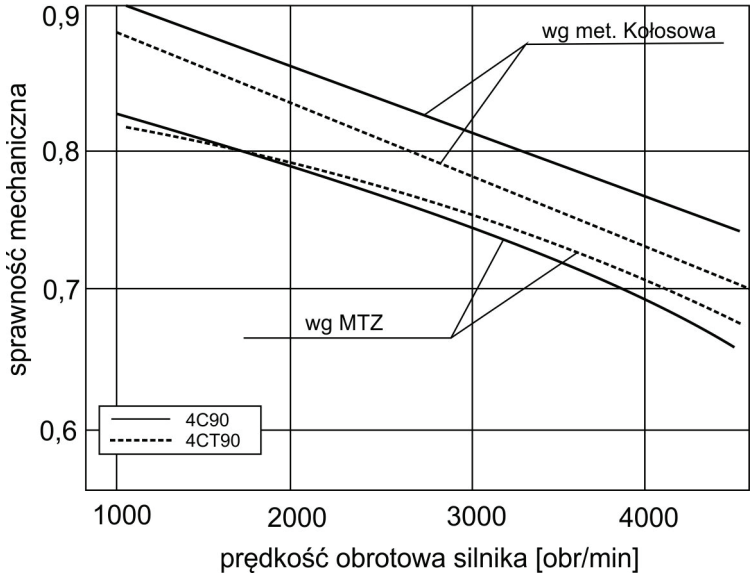
Rys. 3. Zmiany oporów mechanicznych zależnie od przyspieszenia kątowego wału korbowego silnika

Na kolejnym rys. 4 przytoczono straty mechaniczne silnika benzynowego o pojemności 1,3 dm3 w funkcji temperatury dla dwóch olejów smarujących. Rysunek wskazuje na spadek strat tarcia w miarę wzrostu temperatury oleju z jednoczesnym zwiększeniem oporów w miarę wzrostu prędkości obrotowej wału korbowego silnika.

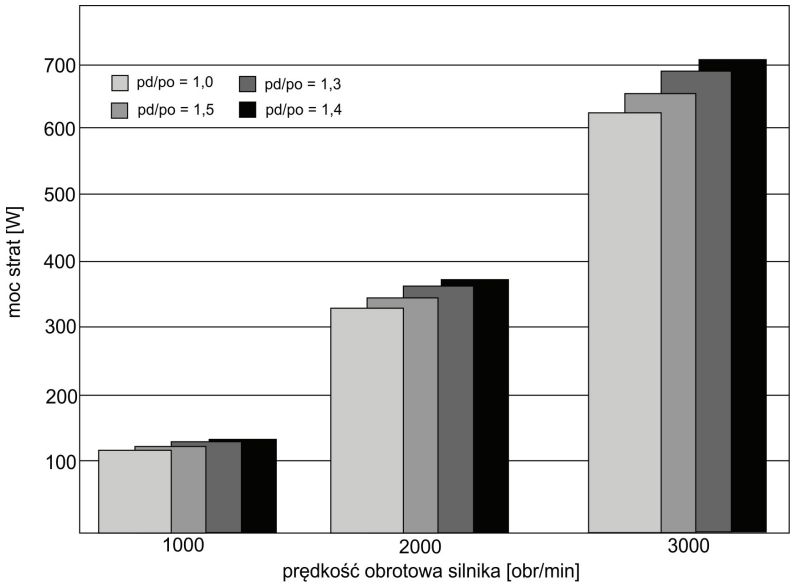


Rys. 4. Straty mechaniczne silnika o zapłonie iskrowym i pojemności 1,3 dm^3 w funkcji temperatury oleju dla dwóch olejów smarujących

Przytoczone wyżej rys. od 1 do 4 dotyczą badań strat mechanicznych w warunkach rzeczywistych z uwzględnieniem wpływu prędkości obrotowej wału korbowego na sprawność mechaniczną silników spalinowych. Natomiast na rys. 5 i 6 zilustrowano wpływ stopnia doładowania na opory tarcia i sprawność mechaniczną silnika ZS (4c 90) w wersji wolnossącej i z doładowanej (4cT 90), otrzymanych metodą analityczną oraz określoną według metody Kołasowa i według MTZ (Motor Technische Zeitschrift) rys 6.



Rys. 5. Wpływ stopnia doładowania na poziom strat tarcia według metody MTZ i J. Kołasowa.



Rys. 6. Wpływ doładowania silnika o zapłonie samoczynnym na jego sprawność mechaniczną.

Należy zauważyć, że zależności otrzymane na drodze teoretycznej korelują z wykresami jakie otrzymano podczas badań eksperymentalnych, jakie przytaczali autorzy artykułu we wcześniejszych publikacjach [4 i 5].

Bibliografia

[1] Bernhardt M., Badania trakcyjnych silników spalinowych. Wydawnictwo WKiŁ, Warszawa 1970 r.

[2] Brun R., Szybkobieżne silniki wysokoprężne. Wydawnictwo WKiŁ, Warszawa 1973 r.

[3] Włodarski J.K., Tłokowe silniki spalinowe procesy trybologiczne. Wydawnictwo WKiŁ, Warszawa 1982 r.

[4] Wrona J. Wrona R., Eksperymentalne badania sprawności mechanicznej silników spalinowych. Autobusy nr 5/2014, s. 131–133.

[5] Wrona J. Wrona R., Cybernetyczny model strat silnika spalinowego. Zeszyty Naukowe WSEI Seria Transport i Informatyka nr 3/2013, s. 59–66.