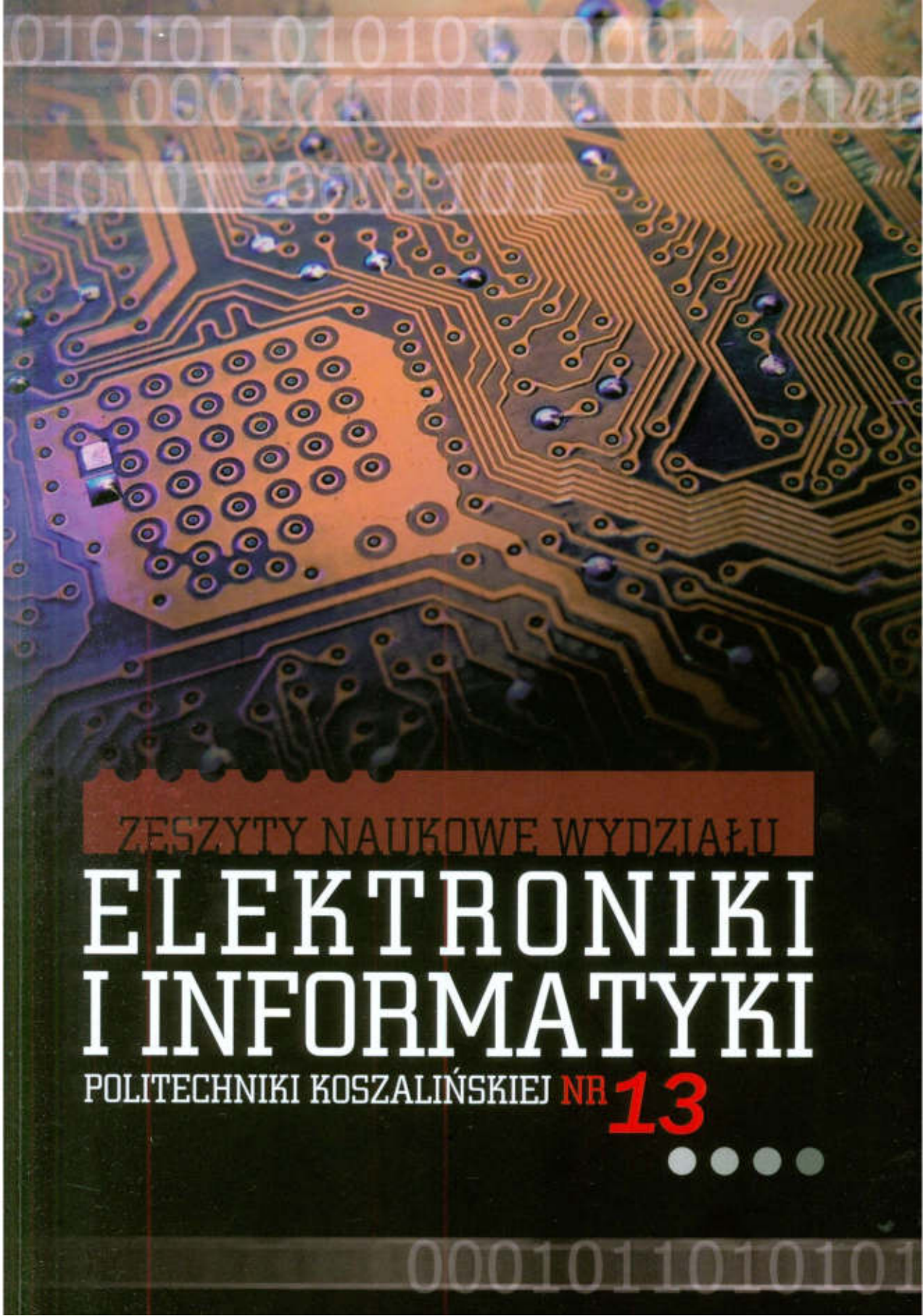


ZESZYTY NAUKOWE WYDZIAŁU

ELEKTRONIKI I INFORMATYKI

POLITECHNIKI KOSZALIŃSKIEJ NR 13

POLITECHNIKA KOSZALIŃSKA

Zeszyty Naukowe
Wydziału Elektroniki i Informatyki

Nr 13

KOSZALIN 2018

Zeszyty Naukowe Wydziału Elektroniki i Informatyki Nr 13

ISSN 1897-7421
ISBN 978-83-7365-501-0

Strona internetowa
https://weii.tu.koszalin.pl/nauka/zeszyty-naukowe

Projekt okładki
Tadeusz Walczak

Skład, łamanie
Maciej Bączek

Wydawnictwo Uczelniane Politechniki Koszalińskiej
75-620 Koszalin, ul. Racławicka 15-17

Koszalin 2018, wyd. I, ark. wyd. 5,72, format B-5, nakład 100 egz.
Druk: INTRO-DRUK, Koszalin

Spis treści

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 5-25

Damian Giebas
Rafał Wojszczyk
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
ul. J. J. Śniadeckich 2
75-453 Koszalin

Zastosowanie wybranych reprezentacji graficznych do analizy aplikacji wielowątkowych

Słowa kluczowe: reprezentacje graficzne, Sieci Petriego, Control Flow Graph, Systemy Współbieżnych Procesów, aplikacje wielowątkowe

Wstęp

Aplikacje pisane na współczesne komputery są bardzo różnorodne i znajdują zastosowanie w prawie każdej dziedzinie życia. Wiele z tych aplikacji to jednowątkowe programy, które wykonują zadania jedno po drugim. Wraz z rozwojem sprzętu komputerowego i wprowadzeniem na rynek procesorów umożliwiających współbieżne wykonywanie zadań zaczęły pojawiać się aplikacje wielowątkowe. Niektóre z języków programowania jak C i C++ nie były tworzone z myślą o wielowątkowości. Aby uzupełnić te braki dla języka C powstała biblioteka pthreads zgodna z wciąż rozwijanym standardem („ISO/IEC 9945-1:2003 - Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions" b.d.). Język C++ otrzymał wsparcie dla wielowątkowości w postaci rozszerzenia biblioteki standardowej wraz z wprowadzeniem standardu C++ 11 (Hinnant i in. 2007).

Języki te wybrano do przedstawiania przykładowych programów wielowątkowych ponieważ istnieje na rynku mnogość oprogramowania stworzonego w tych językach i wciąż wiele takiego oprogramowania powstaje.

Programowanie wielowątkowe w porównaniu do wcześniej stosowanego programowania jednowątkowego posiada szereg zalet jak i szereg wad (Torp 2002).

Najważniejsze z nich zostały przedstawione poniżej:

Zalety

- Responsywność – w przypadku długich zadań w programach z graficznym interfejsem użytkownika programy jednowątkowe ulegają tzw. zamrożeniu (ang. freezing). Problem ten nie występuje w przypadku aplikacji wielowątkowych, gdyż zadania takie mogą zostać oddelegowane do osobnych wątków.
- Współdzielenie zasobów – wątki uruchamiane w ramach jednego procesu współdzielą zasoby komputera. Wszystko dzieje się w ramach jednej przestrzeni adresowej. W przypadku programów jednowątkowych zadania należało delegować do osobnych procesów a komunikacja odbywała się poprzez kopiowanie wartości z jednej przestrzeni adresowej do drugiej.
- Oszczędność – programy wielowątkowe posiadają mniejsze zużycie pamięci niż rozwiązania wykorzystujące kilka jednowątkowych aplikacji.
- Skalowalność – aplikacje wielowątkowe znacznie lepiej wykorzystują możliwości sprzętowe procesorów wspierających wielowątkowość niż zbiór aplikacji jednowątkowych wykonujących wspólnie to samo zadanie. Jednocześnie maszyny stanów aplikacji wielowątkowych są znacznie mniej skomplikowane niż maszyny stanów analogicznego rozwiązania złożonego z aplikacji jednowątkowych.

Wady

- Złożoność kodu aplikacji – każde uruchomienie aplikacji może wyglądać inaczej i jest zależne od aktualnego stanu pozostałych elementów systemu. Programista nigdy nie wie, ile czasu procesora planista przydzieli danemu wątkowi a także nie zna kolejności ich pracy. Taki stan rzeczy posiada więc wpływ na:
  - Debugowanie – debugowanie takich aplikacji jest znacznie utrudnione gdyż sam proces debugowania może wpływać na sposób zachowania aplikacji.
  - Testowanie – testowanie aplikacji jest bardzo trudne gdyż bardzo ciężko jest przewidzieć wszystkie możliwe stany w jakich znajdzie się aplikacja.
- Deadlock – zjawisko nazywany także zakleszczeniem lub blokadą. Sytuacja, w której proces lub wątek w przypadku aplikacji wielowątkowych zamawia dostęp do zasobów i przechodzi w stan oczekiwania. Istnieje możliwość, że oczekujący proces lub wątek nigdy nie zmieni swojego stanu, ponieważ zamawiane przez niego zasoby są

przetrzymywane przez inne czekające procesy (Silberschatz, Galvin, i Gagne 2005).

- Race condition – zjawisko nazywane także szkodliwą rywalizacją. Sytuacja, w której kilka procesów (lub wątków w przypadku aplikacji wielowątkowych) współbieżnie sięga po te same dane i wykonuje na nich działania, wskutek czego wynik tych działań zależy od porządku w jakim następował dostęp do danych (Silberschatz, Galvin, i Gagne 2005).

Zjawiska deadlock i race condition znane były wcześniej gdyż występują one nie tylko w aplikacjach wielowątkowych ale także w rozwiązaniach, w których aplikacje jednowątkowe wykorzystują wspólne zasoby.

Inne znane zjawiska występujące w aplikacjach wielowątkowych to opisany w rozdziale nr 4 atomicity violation i order violation (Lu i in. 2016) nieporuszany w tej pracy. Niniejsza praca skupia się na reprezentacjach graficznych aplikacji wielowątkowych, które pozwolą przede wszystkim na wyeksponowanie miejsc, w których występuje zjawisko typu race condition. Najbardziej znaną reprezentacją graficzną, która pozwoliła na opracowanie metod i zbudowanie narzędzi do detekcji błędów aplikacji wielowątkowych to Control Flow Graph omawiana w rozdziale nr 1. Inną popularną reprezentacją graficzną są Sieci Petriego omawiane w rozdziale nr 2. Wykorzystywane dziś reprezentacje graficzne posiadają pewien szereg ograniczeń, które rzutują na opracowane metody i wykorzystujące je narzędzia. Wśród tych narzędzi znajdują się:

- Helgrind – narzędzie z pakietu Valgrind's Tool Suite[1] do nieinwazyjnego debugowania programów wielowątkowych, pozwalające na wykrycie wszelkiego rodzaju problemów związanych z równoległym dostępem do zasobów. Na stronie twórców znajduje się informacja o tym, że nie gwarantują oni poprawnego działania aplikacji. Mimo wszystkich zalet Helgrind nie posiada możliwości zdalnego debugowania, która to jest niezbędna do pracy w bardzo dużej ilości środowisk gdzie wykorzystywane są języki C i C++ np. w systemach wbudowanych.
- ThreadSanitizer[2] – narzędzie firmy Google bazujące na Helgrind i posiadające także jego ograniczenia. Oba narzędzia wykorzystują algorytm opisany w dokumentacji Helgrind'a. ThreadSanitizer jest narzędziem znajdującym się w pakiecie kompilatorów LLVM/Clang i GCC dla platformy x86. Narzędzie to, podobnie jak Helgrind, jest w fazie beta i jego autorzy nie gwarantują poprawnego działania.

[1] http://valgrind.org/info/tools.html#others
[2] https://clang.llvm.org/docs/ThreadSanitizer.html

- RacerX – narzędzie wykrywające zjawiska race condition i deadlock opisane w pracy (Engler i Ashcraft 2003) wykorzystujące statyczną analizę kodu (tj. pełna analiza kodu źródłowego aplikacji). Wykrywanie odbywa się poprzez stworzenie Control Flow Graf dla analizowanej aplikacji i wzbogacenie go o spisy wywołań funkcji, użytych zmiennych globalnych, wskaźników do zmiennych przekazanych jako parametr i opcjonalnie o spis wszystkich lokalnych zmiennych. Narzędzie to obecnie nie jest publicznie dostępne dla nikogo[3].
- Relay – narzędzie stworzone na Uniwersytecie Kalifornijskim w San Diego do statycznej analizy kodu celem wykrywania zjawiska race condition. Narzędzie to działało na podobnej zasadzie do RacerX, co zostało opisane w pracy (Voung, Jhala, i Lerner 2007). Narzędzie to posłużyło do analizy kodu jądra Linuksa w wersji 2.6.15. Analiza została wykonana na liczbie 4.5 mln. wierszy kodu i wykazała obecność 53 miejsc, w których występowało zjawisko race condition. Jest to jedyny tak szczegółowy raport dotyczący analizy kodu jądra Linuksa, wykonywany poprzez statyczną analizę kodu pod kątem występowania tego zjawiska. Narzędzie choć dostępne publicznie nie jest rozwijane od 2010 roku.

Dwa pierwsze opisane narzędzia do detekcji błędu wykorzystują techniki dynamiczne tj. pracują ze skompilowanym kodem aplikacji, tymczasem dwa kolejne narzędzia do pracy wymagają kodu źródłowego aplikacji, gdyż wykorzystują techniki statyczne poprzez analizę kodu źródłowego. Wykorzystanie Systemów Współbieżnych Procesów (SWP) do detekcji zjawisk race condition i deadlock jest przykładem podejścia statycznego. Główną zaletą metod dokonujących analizy kodu źródłowego jest fakt, że są one niezależne od platformy na jaką pisany jest kod aplikacji, jednak nie są one w stanie uwzględnić opisywanych zjawisk wywołanych agresywną optymalizacją kompilatora (skutkującą np. zamianą pobrania zawartości zmiennej przez stałą liczbową). Zjawiska wywołane agresywną optymalizacją są wykrywane dzięki technikom dynamicznym, jednak narzędzia wykorzystujące techniki dynamiczne są silnie związane z platformą i tak na przykład wszystkie aspekty Helgrinda można wykorzystać tylko na platformach x86 i AMD64.

Języki C i C++ doczekały się rozszerzeń, które pozwalały na równoległą pracę już wcześniej. Rozszerzenie Cilk („A Brief History of Cilk”, b.d.) dla C i C++ zostało stworzone w 1990 roku na MIT i skomercjalizowane jako Cilk++ a następnie sprzedane firmie Intel, która rozwija je pod nazwą CilkPlus. Rozszerzenie to nie zdobyło większej popularności i będzie utrzymywane tylko do

[3] https://goo.gl/DgYzt5

2018 roku. Firma Intel proponuje migrację z CilkPlus do frameworka OpenMP bądź Intel Threading Building Block (Intel TBB).

Wspomniany framework OpenMP (Bull, Reid, i McDonnell 2012) stworzony został dla języków Fortran i C a potem rozszerzony dla C++98 i jest wspierany przez największe firmy w sektorze IT. Zrównoleglanie pracy programu z OpenMP odbywa się poprzez używanie odpowiednich dyrektyw preprocesora, które powodują wzrost złożoności kodu a także nie współpracują z najnowszymi wersjami języka C++.

Konkurencyjne rozwiązanie dla OpenMP czyli biblioteka Intel TBB („Intel Threading Building Blocks Documentation", b.d.) dla języka C++ jest znacznie lepiej przystosowana do współpracy z najnowszymi wersjami tego języka. Niestety w przypadku zastosowania Intel TBB spora część kodu musi zostać napisana na nowo z wykorzystaniem jej elementów.

Charm++ („Introduction to Charm++ Concepts", b.d.) jest dedykowanym frameworkiem dla języka C++ do tworzenia aplikacji z przetwarzaniem równoległym. Wprowadza on nowy paradygmat tj. zorientowane obiektowo asynchroniczne przesyłanie informacji (ang. object-oriented asynchronous message passing parallel programming paradigm), który dekomponuje program do kontenerów (ang. chares), które komunikują się za pomocą obiektów nazywanych komunikatami. Wady tego rozwiązania zostały przedstawione w prezentacji (Aiken b.d.). Największa z wad to łatwa do pominięcia synchronizacja pracy kontenerów, która to jest wymagana aby uniknąć zjawiska race condition. Inną dużą wadą Charm++ jest przeniesienie na programistę obowiązku zarządzania pamięcią komunikatów. Złe zarządzanie może doprowadzić do bardzo groźnych wycieków pamięci w przypadku, gdy następuje alokacja zasobów a nie są one zwalniane.

Powyższe rozwiązania dla języków C i C++ posiadają jedną niepożądaną cechę tj. wysoki poziom skomplikowania kodu napisanego z ich wykorzystaniem. W przypadku wykorzystania biblioteki pthread czy standardu C++11 kod ten jest znacznie bardziej czytelny.

Dalsza część pracy dotyczy lokalizacji zjawiska race condition znajdującego się w kodzie programu przedstawionego na listingu nr 1, przy pomocy reprezentacji graficznych aplikacji wielowątkowych. Program został napisany w języku C z wykorzystaniem biblioteki pthreads. Celem programu z poniższego listingu jest wykonanie miliona operacji inkrementacji zmiennej *balance* przez każdy z wątków aplikacji. Wynikiem działania powinna być wartość dwóch milionów. Niestety operacje inkrementacji na współdzielonym zasobie nie są synchronizowane, wynikiem czego w programie dochodzi do zjawiska race condition. Synchronizacja powinna zostać wykonana poprzez użycie mechanizmów synchronizacji nazywanych mutexami dostarczonych wraz z biblioteką pthreads. Mutexy są to abstrakcyjne struktury, które wykorzystują mechanizm wzajemnego wykluczania się

celem synchronizacji pracy nad wybranymi zasobami. Słowo mutex pochodzi od angielskich słów mutual exclusion.

Mimo iż możliwe jest pisanie w języku C programów wielowątkowych to brak natywnego wsparcia języka powoduje, że w programach tych często występują zjawiska race condition czy deadlock. Poniższy kod aplikacji zawierający zjawisko race condition będzie przekształcany do kolejnych reprezentacji graficznych, w których zjawisko to powinno zostać wyeksponowane, gdyż reprezentacje graficzne nie posiadają ograniczeń języka C i są lepiej przystosowane do przedstawiania wysokopoziomowych idei.

```
#include <stdio.h>
#include <pthread.h>

static volatile int balance = 0;

void* deposit(void *param) {
	// Block B
	char *who = (char*)param;

	int i;
	printf("%s: begin\n", who);
	for (i = 0; i < 1000000; i++)
	// Block C
	{
		balance = balance + 1; // Place with uncontrolled access. Race condition
place.
	}
	// Endblock C
	printf("%s: done\n", who);
	return NULL;
	// Endblock B
}

int main() {
	// Block A
	pthread_t p1, p2;
	char a[] = "A";
	char b[] = "B";
	printf("main() starts depositing, balance = %d\n", balance);

	pthread_create(&p1, NULL, deposit, a);
	pthread_create(&p2, NULL, deposit, b);
	// Endblock A

	// Block D
	pthread_join(p1, NULL);
	pthread_join(p2, NULL);
	printf("main() A and B finished, balance = %d\n", balance);
	return 0;
	// Endblock D
}
```

Listing 1. Kod aplikacji wielowątkowej zawierający zjawisko race condition

Zastosowanie metod graficznych wynika z potrzeby posiadania uniwersalnego narzędzia, które pozwoli na analizę kodu i wykrycie miejsca występowania omawianych zjawisk. Jak już wcześniej było wspomniane graficzne metody są lepsze do przedstawiania wysokopoziomowych idei niż język programowania jakim jest C. Dodatkowo przekształcanie kodu źródłowego do reprezentacji graficznej jest rozwiązaniem niezależnym od platformy, na którą kod będzie skompilowany. Taka transformacja kodu źródłowego do reprezentacji graficznej jest elementem statycznej analizy kodu.

1. Control Flow Graph

Control Flow Graph (CFG) to nic innego jak graf skierowany, który jest jedną z możliwych reprezentacji graficznych aplikacji wielowątkowej. CFG przedstawiony w pracy (Allen 1970) składa się z węzłów i krawędzi, które odpowiadają kolejnym blokom kodu i determinują kolejność ich wykonywania.

CFG zakłada istnienie 3 rodzajów węzłów. Pierwszym rodzajem węzła jest węzeł wejścia (ang. entry node), który cechuje się tym, że nie posiada on przodka natomiast posiada potomków.

Drugi rodzaj węzła to węzeł wyjścia (ang. exit node), który analogicznie do węzła wejścia nie posiada potomków, ale posiada przodków.

Trzeci rodzaj węzłów to węzły posiadające zarówno przodków jak i potomków. Węzły te mogą posiadać przynajmniej jednego przodka i przynajmniej jednego potomka. Przodkowie jak i potomkowie mogą być węzłami bezpośrednimi jak i pośrednimi.

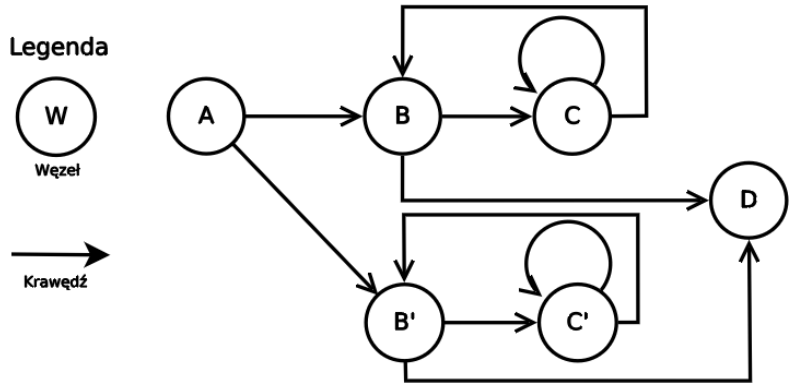
Innymi słowy CFG to graf skierowany G będący parą (B, E), gdzie B jest zbiorem węzłów $\{b_1,b_2,b_3,\ldots b_n\}$ natomiast E jest podzbiorem zbioru wszystkich możliwych krawędzi $\{(b_1,b_2),(b_1,b_3),\ldots,(b_m,b_n)\}$ występującymi między tymi węzłami.

Na rysunku nr 1 przedstawiono CFG dla aplikacji, której kod znajduje się na listingu nr 1. Kod podzielony jest na 4 logiczne bloki, które pozwalają na łatwe jego przekształcenie w CFG. Blok A jest fragmentem kodu przygotowującym aplikację do pracy na wątkach, natomiast blok D jest fragmentem kodu kończącym pracę na wątkach i kończącym pracę aplikacji. Bloki B i C są fragmentem aplikacji wykonywanym równolegle dlatego aby podkreślić ten aspekt aplikacji na CFG, dla jednego wątku zostały one oznaczone jako B i C a dla drugiego jako B' i C'. Dodatkowo blok C zawiera się w bloku B i jego praca jest powtarzana milion razy.

Z wyjątkiem funkcji main każdy logiczny blok tj. dowolna inna funkcja, ciało pętli, ciało instrukcji sterujących lub inny dowolny blok zawierający się w nawiasach klamrowych będzie posiadał swoje odzwierciedlenie w postaci węzła.

Funkcja main w językach C i C++ jest miejscem początku i końca pracy aplikacji dlatego rozbita została na wyżej omówione bloki A i D.

Rysunek nr 1 przedstawia CFG aplikacji, której kod znajduje się na listingu nr 1. Diagram zaczyna się od węzła A znajdującego się w funkcji main. Poprzedza on utworzenie dwóch wątków aplikacji, które przedstawione są jako węzły B i B'. Węzły C i C' są węzłami odpowiadającymi ciału pętli for, a więc dopóki warunek pętli będzie spełniony wciąż będzie wykonywać się wskazany blok, na co wskazuje obecność krawędzi (*C*,*C*) i krawędzi (*C'*,*C'*). Po zakończeniu pracy pętli kontrola wraca do głównego ciała funkcji, a więc do bloków B i B'. Program posiada już tylko blok D odpowiadający za zakończenie pracy a odpowiadający mu węzeł D kończy graf.



Rys. 1. Control Flow Graph aplikacji z listingu nr 1

Utworzony CFG odzwierciedla dokładnie kolejność wykonywanych bloków kodu, jednakże nie można wyczytać z niego informacji dotyczących operacji na współdzielonych zasobach. Operacje te, dziejące się w bloku C aplikacji, nie posiadają swojej reprezentacji graficznej i CFG będzie identyczny zarówno dla prawidłowo działającej aplikacji jak i tej, w której znajduje się zjawisko race condition. Wykorzystanie wyłącznie CFG nie jest wystarczająco dobrą notacją pozwalającą na wykrycie zjawisk race condition i deadlock.

Inną niedogodnością jest fakt, że CFG nie pozwala pokazywać zagnieżdżeń bloków. Bez dokładnego opisu można odnieść wrażenie, że po wyjściu z bloku C i powrotu do bloku B można ponownie wrócić do bloku C, co nie jest możliwe do wykonania w aplikacji.

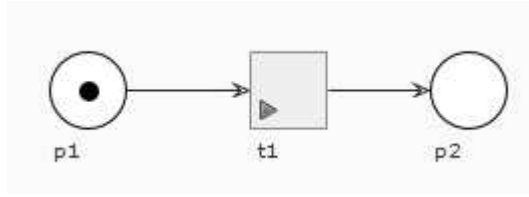
Fakt, że dany blok jest wykonywany w dwóch niezależnych wątkach pozwala domniemać, że jest to miejsce, w którym może wystąpić zjawisko race condition. W przypadku systemów, gdzie wątki nie współdzielą między sobą żadnych zasobów należałoby również sprawdzić wszystkie takie miejsca. W sytuacji gdy zasób jest

współdzielony przez dwa wątki, które nie posiadają wspólnych bloków, mechanizm ten jest niewystarczający do zlokalizowania zjawiska race condition.

Control Flow Graf jest wykorzystywane w narzędziach do detekcji omawianych zjawisk np. w narzędziu RacerX, a każdy z węzłów CFG tworzonych przez to narzędzie dodatkowo wzbogacany jest o spisy wywołań funkcji, użytych zmiennych globalnych, wskaźników do zmiennych przekazanych jako parametr i opcjonalnie o spis wszystkich lokalnych zmiennych. Dopiero w sytuacji, gdy istnieje komplet tych wszystkich informacji możliwe jest wykrycie zjawiska race condition.

2. Sieć Petriego

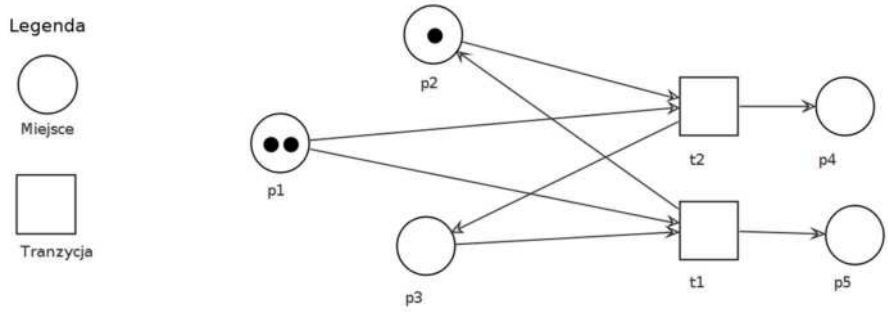
Sieć Petriego (SP) to formalny model przepływu informacji zaprojektowany do opisywania systemów asynchronicznych, w których zadania wykonywane są równolegle.



Rys. 2. Przykład Sieci Petriego

Sieci Petriego składają się z miejsc i tranzycji połączonych krawędziami skierowanymi (Peterson 1977). Przepływ informacji jest wykazywany poprzez przesuwanie żetonów między miejscami poprzez przejście po krawędziach. Na krawędziach znajdują się tranzycje, które odpowiadają za pozwolenie dokonania przejścia, co następuje gdy na wszystkich miejscach wejściowych tranzycji znajdują się żetony. Najprostszy przykład SP przedstawia rysunek nr 2. Znajdują się na niej dwa miejsca p1 i p2 i jedna tranzycja t1. Żeton znajdujący się w miejscu p1 zostanie przeniesiony po krawędziach do miejsca p2 ponieważ jest spełniony warunek przejścia tj. miejsce p1 jest jedynym miejscem wejściowym tranzycji t1 i posiada żeton.

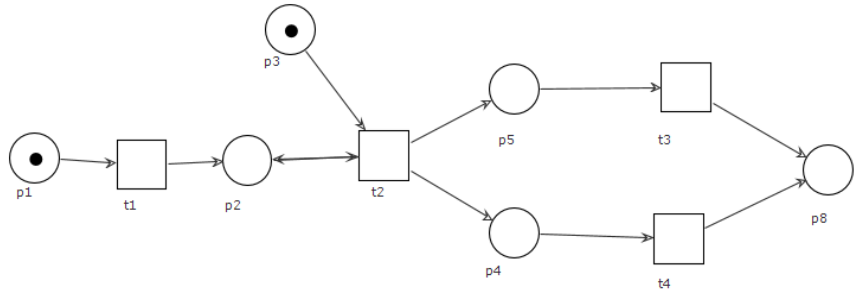
W przeciwieństwie do CFG, SP nie buduje się tylko na logicznych blokach kodu. Przy budowie należy uwzględnić takie rzeczy jak stan początkowy niektórych elementów tj. miejsce odzwierciedlające licznik pętli powinno posiadać tyle żetonów ile iteracji wykona pętla, czy też informacje o kolejności wykonywania poszczególnych zadań w przypadku gdy mogą one zostać zrobione równolegle. Przykładem może być sieć z rysunku nr 3 przedstawiająca zastosowanie mechanizmu wzajemnego wykluczania, który narzuca kolejność przesunięć żetonów z miejsca p1 przez tranzycję t1 i t2.



Rys. 3. Przykładowa Sieć Petriego z mechanizmem wzajemnego wykluczania

Możliwość wykorzystania mechanizmów wzajemnego wykluczania (w powyższym przykładzie składa się on z tranzycji t1, t2 i miejsc p2, p3) pozwala na kontrolę przesunięć żetonów w sieci i symulować wielowątkowość aplikacji. Jednakże nie jest to realne odzwierciedlenie. W przypadku aplikacji wielowątkowych, których głównym celem jest szybkość wykonania operacji, programista nie narzuca ich kolejności wykonywania. To planista decyduje, który wątek w danym momencie pracuje i sytuacja naprzemiennej pracy wątków (jak zostało to pokazane na rysunku powyżej) jest mało prawdopodobna.

Na rysunku numer 4 przedstawiona została Sieć Petriego dla rozważanej aplikacji. Sieć ta jest zbudowana z 6 miejsc i 4 tranzycji. Miejsce p1 odpowiada blokowi A wybranej aplikacji i oznacza jej uruchomienie. Miejsce p2 jest odpowiednikiem momentu uruchomienia obu wątków aplikacji. W przypadku Sieci Petriego możemy symulować działanie pętli for, a więc blok C w tym przypadku będzie składał się z miejsc (p3, p5, p8) i tranzycji (t2, t3) dla pierwszego wątku, a także z miejsc (p3, p4, p8) i tranzycji (t2, t4) dla drugiego wątku. Miejsce p3 jest licznikiem pętli, które powinno mieć milion żetonów, gdyż tyle iteracji wykonuje każda z pętli w wątkach. Umożliwi to każdej gałęzi sieci wykonać się milion razy, tak jak w każdym wątku wykonywane jest milion operacji na wspólnym zasobie.



Rys. 4. Sieć Petriego aplikacji wielowątkowej z listingu nr 1

Miejsce p8 jest odpowiednikiem bloku D aplikacji i kończy ono całą sieć, a ilość żetonów odpowiada wartości zmiennej balance. Jeśli na miejscu p3 znajdowałoby się milion żetonów (tj. tyle ile wynosi maksymalna ilość iteracji pętli w bloku C) to po wykonaniu symulacji w miejscu p8 znajdzie się ich 2 miliony.

Konstrukcja sieci nie pozwala na wystąpienie sytuacji, w której zachodzi zjawisko race condition, tak więc wynik działania sieci będzie zgodny z oczekiwanym wynikiem działania aplikacji ale nie z jej realnym działaniem.

Dodatkową wadą takiej reprezentacji jest to, że do jednego i tego samego kodu aplikacji można zbudować wiele modeli sieci. Sytuacja ta powoduje, że gdy stworzony zostanie model sieci dla aplikacji nigdy nie ma pewności, że będzie można odczytać z niego wszystkie niezbędne informacje pozwalające zlokalizować poszukiwane informacje.

W przypadku zastosowania w sieci mechanizmów wzajemnego wykluczania należy zawsze określić, która tranzycja będzie posiadała priorytet, wynikiem czego jest z góry ustalona kolejność działania tranzycji. Sytuacja taka nie ma miejsca w aplikacjach. Programista nigdy nie ma pewności, który wątek pierwszy otrzyma dostęp do zasobu, gdyż praca wątków nastawiona jest na jak najszybsze wykonywanie zadań i są one wykonywane od razu w momencie, gdy planista przydzieli wątkowi czas procesora. Także w przeciwieństwie do sieci mechanizmy wzajemnego wykluczenia dostarczane z językiem C nie wymuszają kolejności pracy wątków.

3. Sformułowanie problemu

Przedstawiona analiza dwóch powszechnie znanych reprezentacji graficznych aplikacji wielowątkowych pozwala stwierdzić, że za ich pomocą zlokalizowanie zjawiska race condition jest bardzo złożone i w wielu przypadkach wymaga stosowania dodatkowych (nadmiarowych) mechanizmów kontrolnych.

Posiadany jest kod aplikacji wielowątkowej napisany w języku C z wykorzystaniem biblioteki pthreads.

Ograniczeniami jest składnia języka C, jego gramatyka a także fakt, że obliczenia muszą być wykonywane równolegle.

Zatem pytanie jest następujące. Czy kod aplikacji jest poprawny tj. nie występują w nim zjawiska:

- deadlock,
- race condition?

Dwie wcześniej omówione reprezentacje graficzne nie pozwoliły na stwierdzenie czy kod znajdujący się na listingu nr 1 jest wolny od tych zjawisk. W punkcie nr 4 przedstawiona zostanie reprezentacja wykorzystująca modele systemów współbieżnych procesów do tego celu. Przedstawione zostaną dwie reprezentacje,

z których na pierwszej zjawisko race condition będzie widoczne, a na drugiej zostanie przedstawione rozwiązanie eliminujące to zjawisko.

4. Model SWP dla aplikacji wielowątkowych

System procesów jest to zbiór procesów $P=\{P_i|i=1\ldots ln\}$ realizujących operacje w oparciu o zbiór wspólnie wykorzystywanych zasobów $R=\{R_k|k=1\ldots lm\}$. Współbieżne wykonanie procesów oznacza, że każda kolejna operacja jednego procesu rozpoczyna się przed zakończeniem operacji innego procesu i związane jest z ograniczonym dostępem procesów do współdzielonych zasobów (Banaszak, Majdzik, i Wójcik 2008). Specyficznym przypadkiem są systemy, w których procesy są realizowane cyklicznie (tzn. operacje procesów są powtarzane wielokrotnie w stałych odcinkach czasu). W tym ujęciu przez System Współbieżnych Procesów Cyklicznych (SWPC) rozumie się jako zbiór wykonujących się współbieżnie procesów cyklicznych, które są związane ze sobą poprzez korzystanie ze wspólnych zasobów (Bocewicz, Banaszak, i Wójcik b.d.).

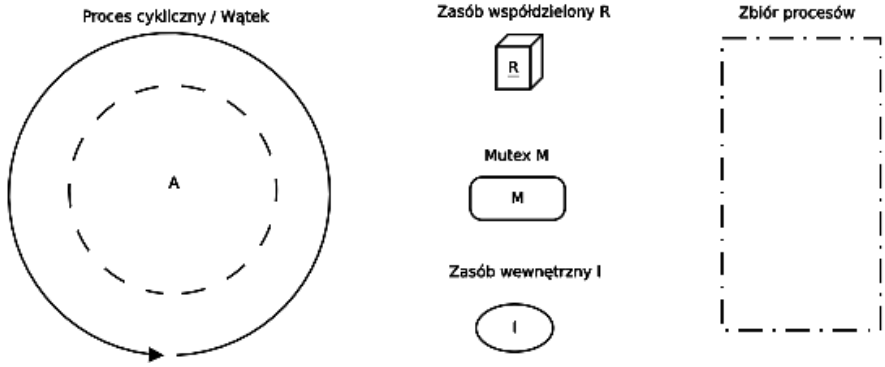
Gdy mowa o SWPC należy wspomnieć o konfliktach żądań zasobowych, które są konsekwencją wystąpienia m.in. takich zjawisk jak zagłodzenie i blokada. Podobne zjawiska można spotkać w aplikacjach wielowątkowych. Zagłodzenie (ang. starvation) występuje w momencie, gdy jeden z wątków aplikacji przez cały okres jej działania nie zwalnia określonego zasobu i tym samym uniemożliwia dostęp do niego innym wątkom. Blokada (ang. deadlock) natomiast występuje wtedy, gdy dwa wątki (lub więcej) próbują otrzymać dostęp do wzajemnie zajmowanych przez siebie zasobów i powstaje tzw. cykl żądań zasobowych. Sytuacja taka powoduje, że każdy z wątków czeka aż pozostałe zwolnią swoje zasoby, co nigdy nie następuje.

Kolejnym specyficznym zjawiskiem aplikacji wielowątkowych jest zjawisko race condition czyli sytuacja, w której stan współdzielonego zasobu (np. wartość zmiennej reprezentowanej przez ten zasób) jest zmieniany przez jeden z wątków w momencie, gdy inne wątki dokonują operacji z już nieaktualną wartością zasobu. Konsekwencją takiego zjawiska jest możliwość uzyskiwania różnych wyników aplikacji (często trudnych do przewidzenia) w zależności od kolejności dostępu wątków do współdzielonych zasobów.

Analogicznie jak omawiane w poprzednich punktach modele CFG i SP systemy współbieżnych procesów cyklicznych mogą również być wykorzystywane do reprezentacji aplikacji wielowątkowych. W tym celu wykorzystuje się zbiór elementów graficznych (Rys. 5) składający się z:

- zasobów współdzielonych reprezentujących instancję dowolnego typu, która jest współdzielona między wątkami np. poprzez wskaźnik lub jako zmienna globalna,

- zasobów wewnętrznych wątków, które podobnie jak zasoby współdzielone są instancjami dowolnego typu, a ich okres życia trwa tak długo jak okres życia wątków,
- procesów cyklicznych reprezentujących wątki aplikacji,
- mechanizmu synchronizacji (mutexu) zapewniającego wzajemne wykluczanie procesów na zasobach (w języku C mutex jest algorytmem implementowanym w postaci obiektu, na którym mogą być wykonywane operacje blokowania i zwolnienia).

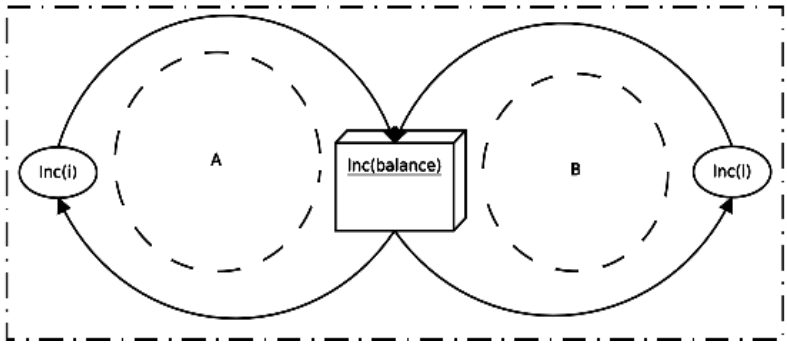


Rys. 5. Elementy SWPC zastosowane modelowania aplikacji wielowątkowych

Zarówno na zasobach jak i mutexach mogą być wykonywane operacje procesów cyklicznych (nazwy tych operacji są podawane wewnątrz zasobu). Są to między innymi:

- Inc – operacja inkrementacji zasobu,
- Lock – operacja założenia blokady na obiekcie mutex'u,
- Unlock – operacja zwolnienia blokady z obiektu mutex'u.

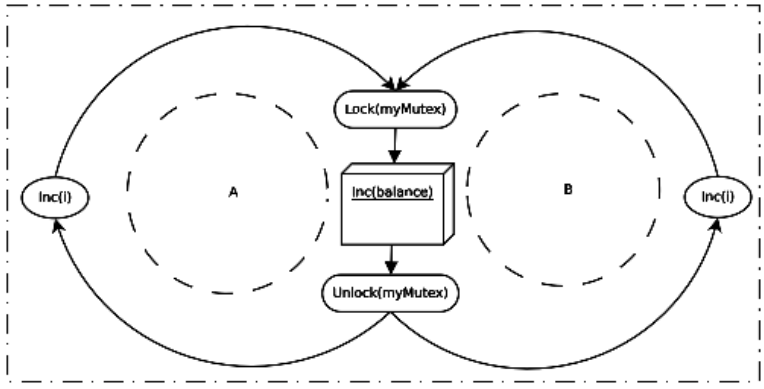
Proponowany model SWPC (wykorzystujący zaproponowany zestaw elementów), w przeciwieństwie do Sieci Petriego i CFG, ukrywa wiele szczegółów implementacyjnych. Uwydatnione na nim zostaną tylko te cechy aplikacji, które są istotne do oceny jej poprawności (pod względem występowania zjawisk prowadzących do konfliktów żądań zasobowych). Takie podejście powinno pozwolić na dokładne odtworzenie aplikacji z modelu i jednocześnie wskazać miejsca, w których może występować zjawisko race condition lub deadlock.



Rys. 6. Model SWPC aplikacji wielowątkowej z listingu nr 1

Rysunek nr 6 przedstawia SWPC dla aplikacji z listingu nr 1. Różni się on znacznie od sieci Petriego i CFG. System zawiera parę procesów (A, B) odpowiadających obu wątkom rozważanej aplikacji. Procesy A i B znajdują się w ramach jednego zbioru, tak samo jak oba wątki pracują w ramach jednej aplikacji. Oba procesy wykonują operację zwiększenia wartości współdzielonego zasobu o nazwie *balance* lub zwiększają wartość swoich wewnętrznych zasobów, analogicznie do wątków przykładowej aplikacji. Pozostałe elementy aplikacji tj. wyświetlanie informacji na standardowe wyjście, inicjalizacja zmiennych czy zakończenie pracy wątków zostają ukryte, gdyż są one zbędne w procesie wykrywania zjawiska race condition.

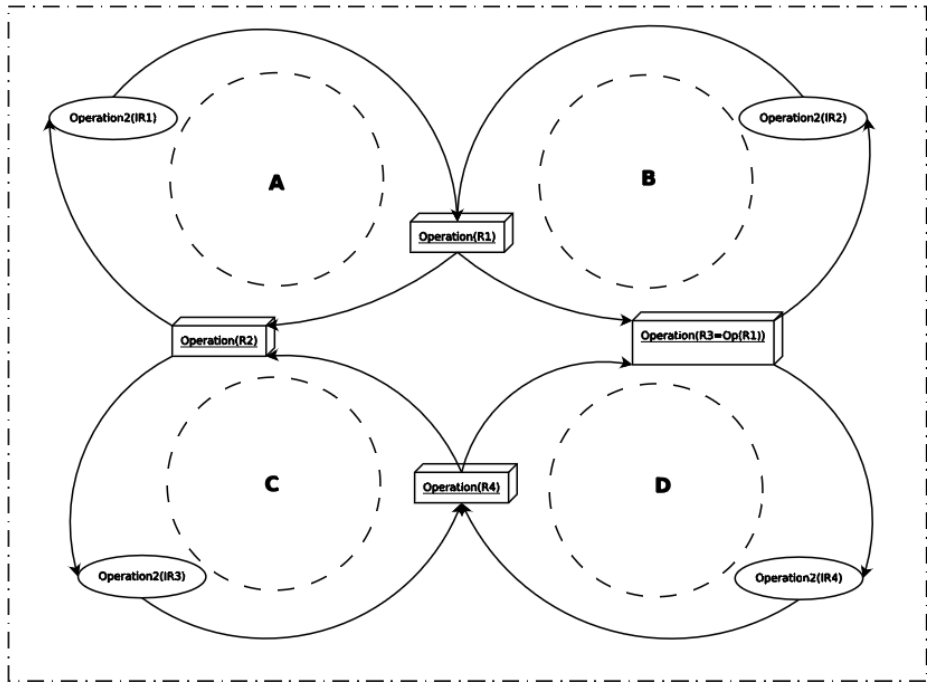
Prezentowany SWPC choć bardzo ogólny, posiada niezbędne informacje dotyczące odtworzenia rozważanej aplikacji. Programista otrzymuje komplet informacji pozwalających odtworzyć jej kod. Na rysunku łatwo dostrzec, że praca na współdzielonym zasobie nie jest synchronizowana tj. brakuje mutexu zapewniającego wzajemne wykluczanie procesów na zasobie współdzielonym. Oznacza to, że na tym zasobie może dochodzić do zjawiska race condition.



Rys. 7. Model SWPC aplikacji z listingu nr 1 bez błędu race condition

Ukrycie zbędnych detali dotyczących realizowanych w aplikacji wątków czyni model bardzo czytelnym. Pominięcie szczegółów implementacyjnych nie wpływa na ocenę poprawności aplikacji. W przeciwieństwie do SP i CFG, model SWPC uwydatnia wrażliwe elementy aplikacji, co przekłada się na lepsze przedstawienie sposobu działania aplikacji i na zlokalizowanie miejsc, gdzie mogą wystąpić potencjalne błędy.

Wyeliminowanie błędu wynikającego z wystąpienia zjawiska race condition jest możliwe w wyniku dodania elementów synchronizacji. Na rysunku nr 7 przedstawiono model SWPC z mutexami, które eliminują zjawisko race condition. W przedstawionym rozwiązaniu procesy przed zajęciem zasobu współdzielonego blokują do niego dostęp (Lock(myMutex)), a następnie go zwalniają (Unlock(myMutex)) po zakończeniu operacji na tym zasobie.

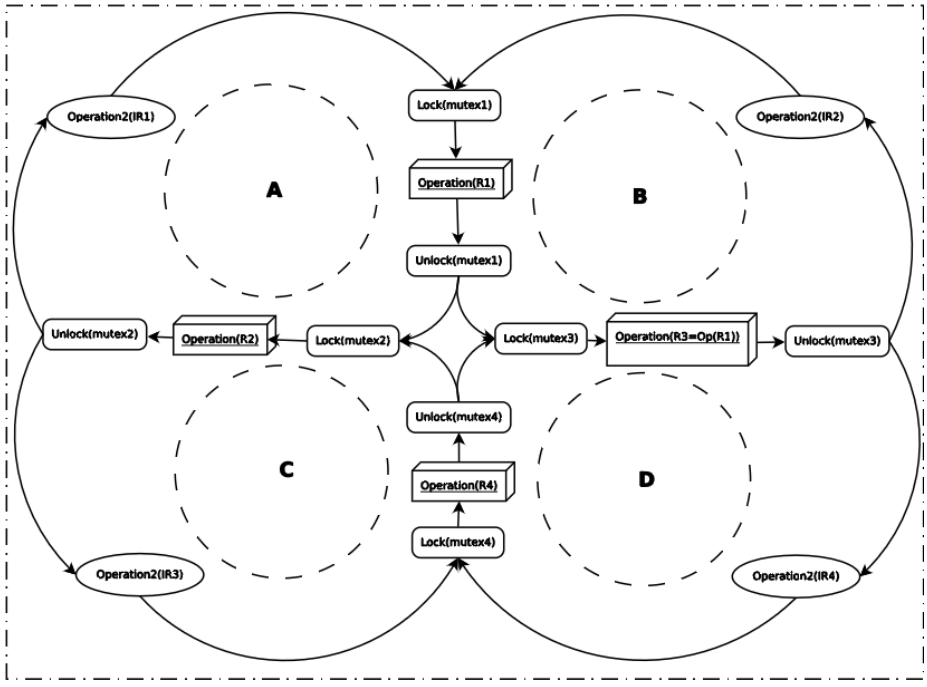


Rys. 8. Przykładowy model SWPC przykładowej aplikacji bez elementów synchronizujących

Aplikacja z listingu nr 1 jest przykładem, dla którego zbudowany SWPC nie jest skomplikowany. Na rysunku nr 8 znajduje się model SWPC dla hipotetycznej aplikacji posiadającej cztery wątki. W aplikacji znajdują się cztery współdzielone zasoby R1, R2, R3 i R4, a każdy z wątków pracuje z dwoma z nich i z własnym zasobem wewnętrznym. Dodatkowo w wątku B operacja na zasobie R3 jest zależna

od nowej wartości zasobu R1 (zależność ta jest wyrażona poprzez równanie R3=Op(R1) wpisane w element graficzny) ustawianej właśnie przez wątek B. Z rysunku łatwo można wyczytać, że operacje realizowane na zasobach współdzielonych nie są synchronizowane, a więc niewątpliwie może dochodzić do zjawiska race condition. Poza race condition w aplikacji zachodzi także zjawisko atomicity violation. Zjawisko to jest konsekwencją związku między zasobem R1 i R3. Stan zasobu R1 wpływa na stan zasobu R3. Zanim proces B wykona operację na zasobie R3, stan zasobu R1 może zostać zmieniony przez proces A.

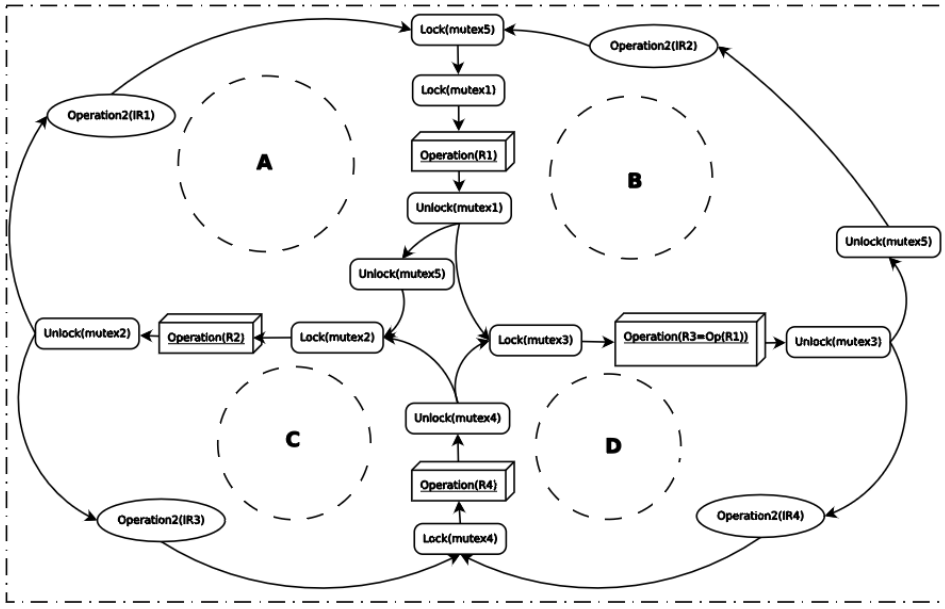
Wyeliminowanie zjawiska race condition sprowadza się do umieszczenia w aplikacji 4 mutexów mutex1, mutex2, mutex3, mutex4 celem zapewnienia wzajemnego wykluczania procesów na zasobach współdzielonych - odpowiedni SWPC jest przedstawiony na rysunku nr 9. Przed każdą operacją na współdzielonym zasobie dokonywana jest akcja blokady na odpowiednim mutexie, a po jej wykonaniu mutex ten jest zwalniany.



Rys. 9. Przykładowy model SWPC przykładowej aplikacji z atomicity violation

Niestety takie podejście nie eliminuje zjawiska atomicity violation. Zjawisko to wciąż jest obecne, gdyż wątek B po zwolnieniu mutex1 przechodzi do operacji blokowania mutex3. W tej sytuacji niezabezpieczony przez wątek B zasób R1 może zostać zmieniony przez wątek A.

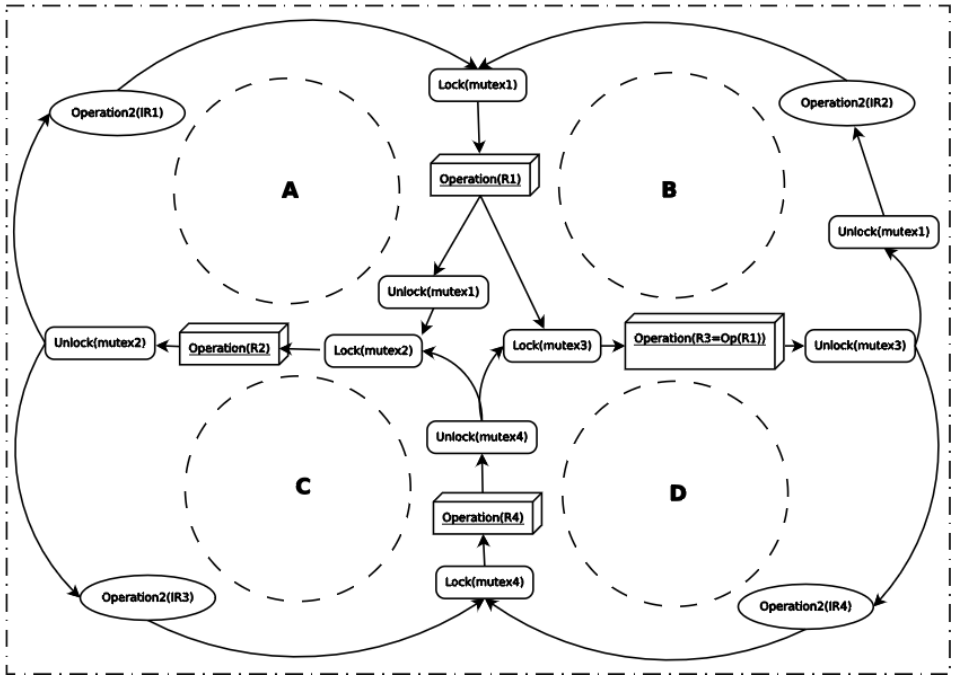
Jednym z dopuszczalnych sposobów wyeliminowania tego zjawiska jest wprowadzenie dodatkowego mutexu, który w wątku B będzie strzegł pracy na obu zasobach, a w wątku A tylko operacji na zasobie R1. Model z takim mutexem został przedstawiony na rysunku nr 10. Wyeliminowanie zjawiska poprzez dodanie kolejnego mutexu zwiększa ryzyko wystąpienia blokady jednakże, istnieje rozwiązanie lepsze tj. bez dodawania mutex5.



Rys. 10. Przykładowy model SWPC przykładowej aplikacji z rozwiązanym atomicity violation poprzez nadmiarowy mutex

Rozwiązanie, które pozwoli na eliminacje zjawiska atomicity violation bez dodawania mutex5 zostało przedstawione na rysunku nr 11. Rolę elementu synchronizującego pracę wątku B otrzymał mutex1, dzięki czemu można było usunąć nadmiarowy mutex. W momencie, gdy wątek B zaczyna pracę blokuje on możliwość pracy na współdzielonych zasobach wątkom A i D, aż do momentu gdy skończy pracę.

Zaprezentowane modele dla hipotetycznej aplikacji pokazują, że dzięki SWP w bardzo prosty sposób można zlokalizować nie tylko zjawisko race condition, ale także zjawisko atomicity violation. Dodatkowym atutem SWP jest jego czytelność, co pozwoliło na optymalizację polegającą na usunięciu nadmiarowego mutexu. Operacja ta wpłynie na szybkość aplikacji, ponieważ do wykonania jest mniej operacji blokowania i odblokowania, które potrafią być bardzo kosztowne.



Rys. 11. Przykładowy model SWPC przykładowej aplikacji z rozwiązaniem dla atomicity violation z minimalną ilością mutexów

Podsumowanie

Wszystkie 3 przedstawione reprezentacje posiadają swoje wady i zalety. W temacie aplikacji wielowątkowych CFG powinien być używany w momencie kiedy obiektem zainteresowania jest ilość bloków logicznych i kolejność ich wykonywania. Niestety CFG jest bardzo ogólną graficzną reprezentacją i nie nadaje się do analizy relacji między wątkami, bez dodatkowych informacji o poszczególnych blokach kodu, które są przedstawiane jako węzły.

Sieci Petriego są narzędziem dużo bardziej wyrafinowanym. Pozwalają pokazać mechanizm wzajemnego wykluczania i przepływ informacji. Jednakże poziom skomplikowania sieci będzie rósł wraz z poziomem skomplikowania aplikacji, a próba jej optymalizacji może spowodować ukrycie istotnych szczegółów. Dla każdej aplikacji wielowątkowej możliwe jest również stworzenie wielu różnych SP. Każda z sieci może działać dokładnie tak jak zakłada idea aplikacji wielowątkowej, natomiast żadna z nich nie będzie działać tak jak realna aplikacja, gdy w aplikacji występuje race condition.

Metoda wykorzystująca modele SWP wydaje się znacznie lepszym rozwiązaniem niż dwie poprzednie metody. SWP ukrywa większość szczegółów

implementacyjnych uwydatniając te miejsca, w których może wystąpić błąd race condition, atomicity violation czy deadlock, który podobnie do atomicity violation jest zjawiskiem wynikającym z niepoprawnego ustawienia mutexów. Interpretacja modelu SWP jest znacznie prostsza niż w przypadku SP czy CFG, a rozszerzenie notacji pozwoliło na zlokalizowanie miejsca wystąpienia błędu race condition w przykładowej aplikacji. Dodatkowo niewielka zmiana w modelu SWP pokazała jak należy rozwiązać problem race condition w przykładowej aplikacji czy atomicity violation w hipotetycznej aplikacji w rozdziale nr 4. Na niekorzyść metody wykorzystującej modele SWP przemawia fakt, że nie były one tworzone z myślą o aplikacja wielowątkowych, więc na potrzeby niniejszego artykułu należało rozszerzyć standardową notację, aby można było za jej pomocą wyrazić wszystkie niezbędne elementy aplikacji wielowątkowej.

Literatura

1. „A Brief History of Cilk”. b.d. https://www.cilkplus.org/cilk-history.
2. Aiken, Alex. b.d. „Charm++”. Udostępniono 28 październik 2017. https://web.stanford.edu/class/cs315b/lectures/lecture11.pdf.
3. Allen, Frances E. 1970. „Control Flow Analisys”. http://sumanj.info/secure_sw_devel/p1-allen.pdf.
4. Banaszak, Zbigniew, Paweł Majdzik, i Robert Wójcik. 2008. *Procesy współbieżne. Modele efektywności funkcjonowania*.
5. Bocewicz, Grzegorz. 2013. *Modele multimodalnych procesów cyklicznych*.
6. Bocewicz, Grzegorz, Zbigniew Banaszak, i Robert Wójcik. b.d. „Harmonogramowanie pracy wózków samojezdnych w warunkach ograniczonego dostępu do współdzielonych zasobów ESW”. Udostępniono 28 październik 2017. https://s3.amazonaws.com/academia.edu.documents/40428772/Harmonogramowane_pracy_wzkw_samojezdnych20151127-13023-1hjyvu1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1509226185&Signature=dw0FK%2FEGMwDVjXYuoovI%2FpsG%2F9Q%3D&response-content-disposition=inline%3B%20filename%3DHarmonogramowane_pracy_wozkow_samojezdny.pdf.
7. Bull, J. Mark, Fiona Reid, i Nicola McDonnell. 2012. „OpenMP Application Programming Interface Examples”. W *International Workshop on OpenMP*, 271–274. Springer. https://link.springer.com/chapter/10.1007/978-3-642-30961-8_24.

8. Engler, Dawson, i Ken Ashcraft. 2003. „RacerX: effective, static detection of race conditions and deadlocks". W *ACM SIGOPS Operating Systems Review*, 37:237–252. ACM. http://dl.acm.org/citation.cfm?id=945468.
9. Hinnant, Howard E., Beman Dawes, Lawrence Crowl, Jeff Garland, i Anthony Williams. 2007. „Multi-threading Library for Standard C++". 24 czerwiec 2007. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2320.html.
10. „Intel Threading Building Blocks Documentation". b.d. https://software.intel.com/en-us/tbb-documentation.
11. „Introduction to Charm++ Concepts". b.d. http://charmplusplus.org/tutorial/CharmConcepts.html.
12. „ISO/IEC 9945-1:2003 - Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions". b.d. Udostępniono 10 wrzesień 2017. https://www.iso.org/standard/38789.html.
13. Lu, Shan, Soyeon Park, Eunsoo Seo, i Yuanyuan Zhou. 2016. „Concurrency Testing Topics". kwiecień 6. http://www.it.uu.se/edu/course/homepage/conctest/vt16/testing-module1-lecture3.pdf.
14. Peterson, James L. 1977. „Petrie Nets". http://cs.rpi.edu/academics/courses/spring04/dci/peterson.pdf.
15. Silberschatz, Abraham, Piter B. Galvin, i Greg Gagne. 2005. *Postawy systemów operacyjnych*.
16. Torp, Kristian. 2002. „Multithreading in Java". listopad 19. http://people.cs.aau.dk/~torp/Teaching/E02/OOP/handouts/multithreading.pdf.
17. Voung, Jan Wen, Ranjit Jhala, i Sorin Lerner. 2007. „RELAY: static race detection on millions of lines of code". W *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 205–214. ACM. http://dl.acm.org/citation.cfm?id=1287654.

Streszczenie

Artykuł zawiera zestawienie reprezentacji graficznych, do których możliwa jest transformacja kodu źródłowego aplikacji wielowątkowych. Zestawienie powszechnie wykorzystywanych reprezentacji, jakimi są Control Flow Graph i Sieci Petriego, pozwoliło na analizę tych reprezentacji, pod kątem przydatności do znajdywania popularnych i niepożądanych zjawisk w aplikacjach wielowątkowych. Jako alternatywa dla Control Flow Graph i Sieci Petriego przedstawiono reprezentację Systemów Współbieżnych Procesów. Wszystkie trzy reprezentacje zostały wykorzystane do reprezentacji przykładowej aplikacji napisanej w języku C, zawierającej zjawisko race condition. W podsumowaniu dokonana została ocena,

która zależała od tego czy dana reprezentacja pozwoli odnaleźć wspomniane zjawisko.

Summary

The article contains a list of graphical representations to which it is possible to transform the source code of multithreaded applications. Comparison of commonly used representations, such as Control Flow Graph and Petri Network, allowed to analyze these representations in terms of their usefulness in finding popular and undesirable phenomena in multithreaded applications. As an alternative to Control Flow Graph and Petri Network, the representation of Concurrent Processing Systems is presented. All three representations were used to represent a sample C-language application containing race condition. In conclusion, an assessment was made, which depended on whether the representation would allow to find the phenomenon.

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej
ISSN:1897-7421
Nr 13, Listopad 2018
Str: 27-36

Grzegorz Górski
Paweł Koziołko
Zakład Systemów Multimedialnych i Sztucznej Inteligencji
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
ul. J.J. Śniadeckich 2
75-453 Koszalin

Semantyczne ataki na aplikacje internetowe wykorzystujące język HTML i arkusze CSS

Słowa kluczowe: podatności, aplikacje internetowe, zabezpieczenia, HTML, CSS, inwigilacja

1. Wprowadzenie

Istotnym wyzwaniem projektowym dla współczesnych aplikacji internetowych są ich podatności na wycieki informacji chronionych. Są to systemy dostępne w sieci publicznej, często dla nieautoryzowanych użytkowników. Efektywny poziom bezpieczeństwa informacji aplikacji internetowej zależy przede wszystkim od świadomości projektanta i programisty.

Artykuł przedstawia wybrane, istotne podatności języków HTML i CSS oraz prezentuje proste mechanizmy dzięki którym zwykły użytkownik może rozpoznać prawidłowo działającą aplikację, od jej wersji zainfekowanej tzn. przygotowanej do nieautoryzowanego pozyskania informacji. W artykule opisano także sposób zalecany postępowania w takim przypadku oraz zaproponowano rozwiązania pozwalające zwiększyć poziom bezpieczeństwa danych osobowych użytkownika.

Ten artykuł ma charakter wyłącznie edukacyjny i jego zadaniem jest zwiększenie świadomości programistów w zakresie różnego rodzaju podatności aplikacji internetowych.

2. HTML – podatności i metody zabezpieczeń

Zaprezentowane poniżej klasy ataków posiadają status krytycznych zagrożeń bezpieczeństwa wg metodologii OWASP (Open Web Application Security Project).

2.1. Ataki typu Future Proof

Język HTML w aktualnej wersji 5 jest zgodny z wcześniejszymi wersjami, co z jednej strony umożliwia powtórne użycie wcześniej stworzonego kodu, a z drugiej strony powoduje liczne zagrożenia bezpieczeństwa. Wybrane elementy składni odpowiadające za bezpieczeństwo wykonywanych operacji różnią się istotnie pomiędzy wersjami np. 4 i 5. Używanie niezaktualizowanych aplikacji z nowymi interpreterami tego języka skutkuje zgodnością na poziomie funkcjonalnym, lecz w odniesieniu do wymagań niefunkcjonalnych np. bezpieczeństwa ta zgodność już automatycznie nie występuje.

Jedną z podstawowych metod zabezpieczenia aplikacji internetowych przed tego typu atakami jest filtrowanie treści dodawanych przez użytkowników oparte na tzw. blacklistach, czyli listach tagów niedozwolonych.

W specyfikacji języka HTML5 wprowadzono wiele nowych atrybutów nie stosowanych w poprzednich wersjach. Jednym z nich jest „autofocus", który przenosi kursor użytkownika bezpośrednio do pola tekstowego skojarzonego z tym atrybutem. Może to być wykorzystane do przeprowadzenia ataku polegającego wprowadzeniu i wykonaniu obcego kodu do strony HTML. Łącząc ze sobą dwa atrybuty onfocus i autofocus można doprowadzić do potencjalnie niebezpiecznej sytuacji uruchomienia nadmiarowego kodu np. funkcji w języku JavaScript w momencie kiedy wybrane pole zostanie zaznaczone.

```
<input onfocus="alert(1)" autofocus>
```

Rys. 1. Przykład kodu ataku opartego o autofocus i onfocus

Kolejnym przykładem wykorzystującym atrybut autofocus jest dodanie co najmniej dwóch różnych pól z tym atrybutem z których pierwsze zawiera obsługę zdarzenia za pomocą atrybutu „**onblur**". Zdarzenie to zostanie uruchomione w momencie odznaczenia pola w którym jest określone, co nastąpi niezwłocznie, gdyż na drugim z pól został również ustawiony atrybut autofocus. Zaznaczenie drugiego pola w takim przypadku odbywa się bez ingerencji użytkownika.

```
<input onblur="alert(1)" autofocus><input autofocus>
```

Rys. 2. Przykład kodu ataku opartego o autofocus i onblur

Nieco inną odmianą podatności polegającej na wykorzystaniu atrybutu autofocus dla kilku pól jest atak z wykorzystaniem atrybutu onscroll zamiast onblur. W takim przypadku uruchomienie nieautoryzowanego kodu następuje w wyniku przesunięcia strony przez użytkownika lub samoistnie przez odpowiednio spreparowany kod. Oba powyższe ataki zakładają modyfikację aplikacji internetowej, która jest niewidoczna dla zwykłego użytkownika ani dla okresowo uruchamianych automatów testów regresywnych.

Analizując podatności pól tekstowych języka HTML można przedstawić kolejne potencjalne zagrożenia związane z zalecanym do stosowania kontenerem **from.** Kontener ten powinien być używany jako otoczenie dla fragmentu kodu, który zawiera pól tekstowych.

Standardowy sposób działania formularza zakłada, że wszystkie zmiany (np. zmiana wartości w polach tekstowych, wybór opcji z listy rozwijanej, wysłanie samego formularza) są aktywne tylko w obrębie danego kontenera (tagu) **form**. Możliwe jest przeprowadzenie ataku polegające na odwołanie się do formularza spoza kontenera z użyciem identyfikatora tego formularza. Przykład prezentujący odwołanie poza standardowo określonymi granicami kontenera zawarto na rys. 3.

```
<form id="testForm" onforminput="alert(1)">
  <input>
</form>
<button form="testForm" onformchange="alert(2)">x</button>
```

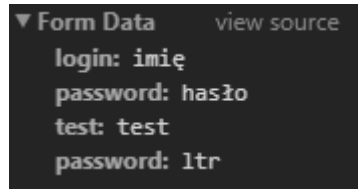
Rys. 3. Przykład inwigilacji formularza z atrybutami onformchange, onforminput

Obrona przed takimi atakami wymaga świadomości programisty polegającej na unikaniu ustawiania identyfikatora formularzy i blokowania stosowania atrybutów „form", „onformchange" i „onforminput". Zasięg zaprezentowanego powyżej przykładu jest ograniczony do użytkowników używających przeglądarki Opera.

Zarówno przeglądarki Opera i Chrome obsługują kolejny potencjalnie niebezpieczny atrybut „dirname", który może być wykorzystany przez atakującego. W takim przypadku przeglądarka udostępni informację na temat kierunku pisania tekstu innego pola w formularzu dodając go do treści żądania wysłanego do serwera. Poprzez umieszczenie atrybutu „dirname", następuje nadpisanie danych wprowadzonych przez użytkownika. Zmodyfikowane pole przyjmie wartość "ltr" lub "rtl" - w zależności od faktycznego kierunku przepływu tekstu. Na rysunkach 4 i 5 zaprezentowano tego typu atak w celu uniemożliwienia wprowadzenia poprawnego hasła. Użytkownik pomimo wprowadzania poprawnego hasła nie zostanie uwierzytelniony przez system.

```
<form method="POST">
    <input name="login"/>
    <input name="password" type="password"/>
    <input name="test" dirname="password" />
    <input type="submit" >
</form>
```

Rys. 4. Przykład kodu wykorzystującego atrybut dirname



Rys. 5. Przykład żądania wysłanego do serwera zmodyfikowanego przez atrybut dirname

Atak z wykorzystaniem atrybutu dirname nie dotyczy przeglądarek Internet Explorer oraz Firefox, gdyż nie jest on jeszcze przez nie obsługiwany.

Kolejna podatność dotyczy trzech najbardziej popularnych przeglądarek tzn. Opera, Chrome i Firefox, które umożliwiają wykorzystanie do obsługi błędu mechanizmu przetwarzającego plik źródłowy ograniczony znacznikami „**video**" i „**audio**" poprzez atrybut „**onerror**". Taka podatność może zostać wykorzystana w przypadku żądania obsługi przez aplikację internetową nieistniejącego lub uszkodzonego pliku wideo. W takim przypadku może zostać uruchomiony nadmiarowy kod zawarty pomiędzy ww. znacznikami.

```
<video><source onerror="alert(1)">
```

Rys. 6. Przykład kodu wykorzystującego atrybut onerror

Jedyną metodą ochrony przed atakiem za pośrednictwem atrybutu onerror jest usunięcie tego atrybutu z listy dopuszczalnych ciągów znaków (whitelist).

3. Rodzina ataków bezpośrednio związanych historią przeglądania stron

Do specyfikacji HTML5 wprowadzono dodatkowy mechanizm **history API,** który pozwala uzyskać dostęp do historii przeglądania stron zarejestrowanych w przeglądarce. Jest to realizowane z wykorzystaniem obiektu „**window**" drzewa DOM. Za pośrednictwem obiektu historii uzyskujemy dostęp do historii zapisanej przez przeglądarkę. Udostępnione metody zwykle stosowane do przeglądania historii użytkownika umożliwiają także przeprowadzenie ataku polegającego na modyfikacji stosu historii.

Do przygotowania ataku mogą zostać wykorzystane poniższe dwie funkcje udostępnione przez History API:

- **history.pushState** (data, title, url)
- **history.replaceState** (data, title, url)

gdzie:

data – dane, które mogą być odczytane w skrypcie,

title – tytuł strony, widoczny w historii przeglądarki (obecnie parametr ignorowany przez Firefox)

url – adres wynikowy, widoczny w pasku przeglądarki

Metoda **history.pushState()** pozwala zmienić adres URL w pasku przeglądarki na inny, zgodny z zasadą Same Origin Policy, co najczęściej jest ograniczone do dokładnie takiej samej nazwy domeny. Zmiana adresu w pasku przeglądarki może być wykorzystana do podania innego adresu (linku) zawartego w nagłówku HTTP podczas komunikacji z innym serwerem.

Metoda **history.replaceState()** działa w sposób zbliżony do history.pushState(), z tą jednak różnicą, że replaceState() modyfikuje ona bieżącą historię zamiast tworzyć nową. Należy jednak nadmienić, że funkcja: history.pushState() uniemożliwia utworzenie nowego wpisu w globalnej historii przeglądarki, co skutkuje brakiem możliwości powrotu do poprzedniej strony.

3.1. Rodzina podatności Reflected XSS

Ataki typu Reflected XSS wykorzystują mechanizm pozwalający na umieszczenie w parametrze zapytania http dodatkowego kodu, który zostanie wyświetlony w wygenerowanej aplikacji. Jednym z najczęściej stosowanych funkcjonalności używającej tego mechanizmu jest obsługa wyszukiwania w obrębie wyświetlanej strony.

```
http://domena.com/szukaj?q=<script>alert(1)</script>
```

Rys. 7. Przykład adresu www z zaszytym atakiem reflected XSS

W przypadku gdy aplikacja wyświetli stronę z wyrażeniem q wstawionym do jej kodu HTML, będzie to atak XSS typu odwzorowanego. Po wykonaniu takiego ataku, adres www wraz z niebezpieczną zawartością zostaje wyświetlony przez przeglądarkę w pasku adresu URL. W celu ukrycia takiej informacji tzn. ukrycia ataku XSS można użyć metody history.pushState() i zmienić aktualny adres www.

```
http://domena.com/szukaj?q=<script>alert(1); history.pushState({}, "",
location.href.split("?").shift())</script>
```

Rys. 8. Przykład adresu www z zaszytym atakiem reflected XSS i maskowaniem adresu www

Jedyną formą obrony przed tego typu atakiem jest zapewnienie aktualnych list filtracyjnych serwera aplikacji. Ataki i metody ich przeciwdziałania po stronie serwerowej wychodzą poza zakres niniejszego artykułu.

4. Wybrane ataki na arkusze styli CSS

CSS jest to język programowania służący do modyfikowania wyglądu aplikacji webowych, który może zostać wykorzystany do kradzieży wrażliwych danych z aplikacji internetowej użytkownika.

Szczególnie podatnym na ataki procesem jest obsługa przez CSS komunikacji z serwerami zewnętrznymi.

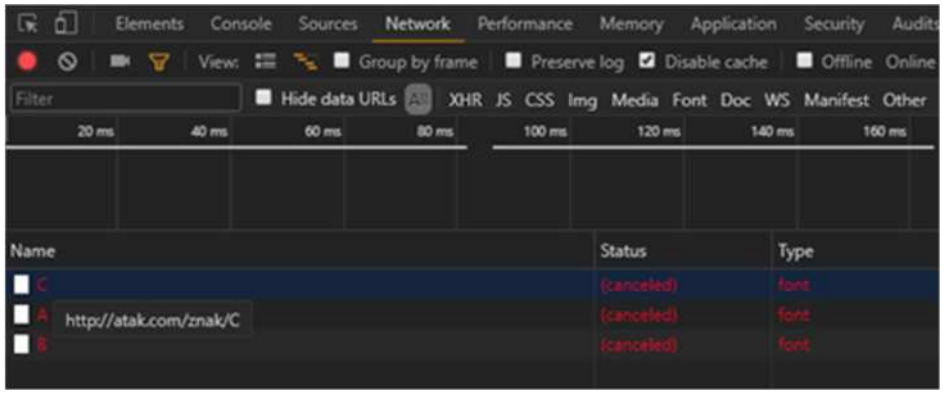
4.1. Ataki za pomocą font-face

Jednym z ataków na arkusz styli CSS polega na wykorzystaniu właściwości **unicode-range.** Scpecyfikacja CSS pozwala na ustawienie specjalnego kroju czcionki **@font-face. P**osiada on właściwość określenia zakresu (unicode-range), która to definiuje jakie znaki mogą występować w pliku pobieranym z zewnętrznego serwera. Atakujący posiadający taką informację może przygotować zestaw czcionek dla każdego znaku (unikodu), która będzie wysyłała żądanie do serwera hackera o przesłanie czcionki z konkretną literą. W kolejnym kroku użytkownik wpisując dane w polu tekstowym, wykorzystującym zhackowany krój czcionki, nieświadomie wysyła je bezpośrednio do serwera atakującego. Mechanizm działania przeglądarek zakłada, że raz wczytana czcionka dla konkretnego znaku nie będzie ponownie pobierana. Oznacza to, że każde kolejne wprowadzenie wybranego znaku spowoduje odwołanie do pamięci przeglądarki. Należy jednak zauważyć, że

atak tego typu pozwoli na stworzenie kompletnego alfabetu znaków w przypadku gdy każdy z nich zostanie chociaż jeden raz wprowadzony przez użytkownika.

```
@font-face {
    font-family: 'atackfont';
    src: url(//atak.com/znak/A);
    unicode-range: U+0041;
}
@font-face {
    font-family: 'atackfont';
    src: url(//atak.com/znak/B);
    unicode-range: U+0042;
}
.unsecured-field {
    font-family: 'atackfont';
}
```

Rys. 9. Deklaracja czcionki



Rys. 10. Wykaz żądań http po wpisaniu znaków CAB w polu tekstowym z klasą unsecred-field

4.2. Ataki poprzez selektor

Istnieje możliwość zaatakowania aplikacji internetowej przy użyciu selektorów CSS, które określają elementy drzewa DOM wraz z przypisaną do nich regułą.

Selektor value wskazuje tylko na te elementy drzewa DOM, których atrybut value przyjmuje zadana wartość. W przypadku gdy w arkuszu styli zostanie umieszczone polecenie input[value=A] to przeglądarka wskaże na pole tekstowe, które posiada dokładnie wartość A w atrybucie value.

Porównując atak z wykorzystaniem selektora do ataku za pomocą font-face należy zauważyć, że atak zostanie zainicjowany tylko w przypadku, kiedy atrybut value będzie miał wcześniej ustawioną wartość, a nie podczas wpisywania tekstu przez użytkownika, tak jak to miało miejsce przy ataku **font-face**.

Atak z selektorem value będzie wykorzystywał specjalnie przygotowany słownik wartości. Jego tworzenie można usprawnić wstawiając do kodu arkusza CSS znak ^ przed wczytywaną wartością co sprawi, że selektor będzie sprawdzał czy wartość w polu tekstowym zaczyna się od wartości zadanej w selektorze. W takim przypadku jeśli wprowadzana wartość będzie dłuższa od zadanej, ale początek będzie zawierał wartość podaną w selektorze to atakujący otrzyma szczątkową informację o pożądanej treści, np. może to być część hasła użytkownika. Przesłanie tej szczątkowej informacji do hakera może być związane z wykonaniem procedury komunikacji z serwerem atakującego np. przy pobraniu pliku graficznego jako tła elementu określonego selektorem.

```
input[value=A] {
    background-image: url(//atak.com/obraz/A.jpg);
}
input[value=B] {
    background-image: url(//atak.com/obraz/B.jpg);
}
input[value=C] {
    background-image: url(//atak.com/obraz/C.jpg);
}
```

Rys. 11. Przykład ataku poprzez selektor

5. Podsumowanie

Zaprezentowane w artykule podatności języka HTML i arkuszy styli CSS mogą powodować ataki na aplikacje internetowe po stronie klienta. Zaprezentowane metody ochrony oprócz świadomości programisty tworzącego kod aplikacji internetowej mogą wymagać także uruchomienia dostępnych mechanizmów filtracji po stronie serwera.

Bibliografia

1. Hague, Matthew, Lin, Anthony W., Ong, C. -H. Luke, 2015, *Detecting Redundant CSS Rules in HTML5 Applications: A Tree Rewriting Approach*
2. Mazinanian, Davood, Tsantalis, Nikolaos, 2016, An empirical study on the use of CSS preprocessors
3. Hale, Matthew L., Hanson, Seth, 2015, *A Testbed and Process for Analyzing Attack Vectors and Vulnerabilities in Hybrid Mobile Apps Connected to RESTful Web Services*
4. Mohammadi, Mahmoud, Chu, Bill, Lipford, Heather Richter, 2017, *Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testing*
5. Jan, Sadeeq, Nguyen, Cu D., Arcuri, Andrea, Briand, Lionel, 2017, A Search-based Testing Approach for XML Injection Vulnerabilities in Web Applications
6. Choi, Su Yeon, Lee, Hae Young, 2016, *Toward Automated Scanning for Code Injection Vulnerabilities in HTML5-Based Mobile Apps*
7. Dayal, Mohit, Singh, Nanhay, Raw, Ram Shringar, 2016, *A Comprehensive Inspection Of Cross Site Scripting Attack*
8. Gupta, Shashank, Gupta, B. B., 2016, XSS-SAFE: A Server-Side Approach to Detect and Mitigate Cross-Site Scripting (XSS) Attacks in JavaScript Code
9. Pan, Jinkun, Mao, Xiaoguang, 2017, *Detecting DOM-Sourced Cross-Site Scripting in Browser Extensions*
10. Liu, Shukai, Yan, Xuexiong, Wang, Qingxian, Zhao, Xu, Chai, Chuansen, (Sun, Yajing ,2016, *A Protection Mechanism against Malicious HTML and JavaScript Code in Vulnerable Web Applications*
11. https://developer.mozilla.org/en-US/docs/Web/API/History_API
12. Wu, Da-Chun, Su, Hsiu-Yang, 2013, *Information Hiding in EPUB Files by Rearranging the Contents of CSS Files*

Streszczenie

Języki programowania HTML i CSS to najczęściej wykorzystywane technologie do tworzenia aplikacji internetowych, których semantyka w powszechnym przekonaniu nie jest zbyt skomplikowana. Prosta składnia języka skutkuje efektywnością implementacji co jednak stoi w sprzeczności z koniecznością zapewnienia ochrony informacji i minimalizacji prawdopodobieństwa wycieku poufnych informacji. W artykule dokonano przeglądu najnowszych ataków na aplikacje internetowe oraz zaprezentowano skuteczne metody ochrony.

Abstract

The programming languages HTML and CSS are the most commonly used technologies for creating internet applications, the semantics of which are generally not too complicated. The simple language syntax results in implementation efficiency, that is contrary to ensure information protection and minimize the likelihood of confidential information leakage. The article reviews the latest attacks on Internet applications and presents effective protection methods.

Keywords: Vulnerabilities, web applications, security, HTML, CSS, surveillance.

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 37-45

Grzegorz Górski
Paweł Koziołko
Zakład Systemów Multimedialnych i Sztucznej Inteligencji
Wydział Elektroniki i Informatyki
Politechnika Koszalińska

Analiza skuteczności wybranych metod ochrony anonimowości stosowanych w przeglądarkach internetowych

Słowa kluczowe: Podatności, aplikacje internetowe, zabezpieczenia, cookies, inwigilacja

1. Wprowadzenie

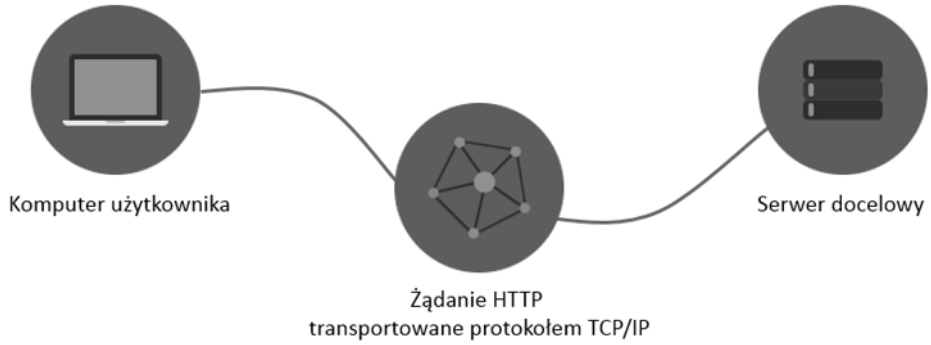
Na podstawie informacji Głównego Urzędu Statystycznego, w 2017 roku w Polsce ponad 78% gospodarstw domowych posiadało Internet szerokopasmowy [1]. Powszechny dostęp do usług oferowanych przez sieć globalną powoduje, że informacje o aktywności poszczególnych użytkowników są narażone na te same zagrożenia, co inne informacje przetwarzane w sieci Internet. Bardzo często stosowane są mechanizmy, których głównym celem jest ułatwienie dostępu użytkownika do wybranych zasobów. Jest to możliwe tylko i wyłącznie dzięki gromadzeniu informacji o preferencjach konsumenta, które są przez niego podawane. Taki zbiór danych - gromadzony jest zarówno na serwerach usługodawców, jak i w systemach operacyjnych, aplikacjach użytkowanych komputerów oraz urządzeń mobilnych stanowi bardzo dokładny opis preferencji użytkownika i może być skutecznie wykorzystany do jego identyfikacji, poszukiwania lub w najlepszym przypadku przesyłania reklam spersonalizowanych tymi informacjami.

Przeglądarki internetowe zapewniają użytkownikom dostęp do wyświetlania wysokiej jakości usług www, które wymagają informacji nie tylko na temat przeglądarki ale także środowiska systemu operacyjnego użytkownika. Dzięki różnym współczesnym narzędziom informatycznym do serwera usługowego bez wiedzy użytkownika przesyłane są szczegółowe informacje związane z konfiguracją

sprzętu i oprogramowania w celu optymalizacji korzystania użytkownika z zasobów Internetu. Pewien zasób informacji o swoich klientach jest niezbędny do świadczenia usług, jednak wiele portali gromadzi nadmiarowe informacje, które same w sobie stają się towarem oferowanym innym usługodawcom bez zgody i wiedzy użytkowników. Należy zauważyć, że nawet niewielkie różnice w informacjach o użytkownikach gromadzonych przez punkty dostępu do usług mogą być skutecznie wykorzystane do przeprowadzenia ataku na urządzenie klienta.

2. Główne obszary inwigilacji użytkownika Internetu

W literaturze naukowej zajmującej się ochroną bezpieczeństwa zdefiniowano trzy główne obszary inwigilacji tj.: urządzenie końcowe wykorzystywane przez użytkownika (komputer lub urządzenie mobilne), treść żądania http przesyłana protokołem TCP/IP a także serwer docelowy. W artykule opisano wybrane grupy zagrożeń z pierwszego z wyżej wymienionych obszarów.



Rys. 1. Główne obszary inwigilacji użytkownika Internetu

3. Podatności po stronie klienta

Urządzenie końcowe klienta jest jednostką najbardziej narażoną na ataki ze strony osób trzecich. Poprzez instalowanie zewnętrznego oprogramowania, przeglądanie stron www oraz podłączanie do niezaufanych, ogólnodostępnych sieci Wi-Fi użytkownicy nieświadomie zwiększają prawdopodobieństwo wycieku danych wrażliwych.

Do przeglądania zasobów sieci Internet wymagana jest przeglądarka internetowa, która tłumaczy język HTML, JavaScript oraz kaskadowe arkusze styli (CSS) i wyświetla wcześniej zaprogramowane strony www. Przeglądarki, oprócz tłumaczenia kodu dynamicznie prezentowanych stron www, zapamiętują również pewne informacje w celu zwiększenia komfortu korzystania z Internetu.

3.1. Pliki cookies

Pliki cookies, zwane także ciasteczkami, są to tworzone na komputerze klienta podczas komunikacji przeglądarki z serwerem, na którym znajduje się strona internetowa. Zawierają dane umożliwiające identyfikację użytkowników. Są jednym ze sposobów zapamiętywania pól formularzy, takich jak login czy dane adresowe, dzięki czemu nie ma konieczności ponownego ich wypełniania przy każdym pobieraniu lub odświeżaniu strony www. Poza polami formularzy, coraz częściej zapamiętywane są zachowania użytkownika na określonej witrynie w celu tworzenia tzw. odcisków palca użytkownika. Mechanizm ten jest ściśle powiązany z zagadnieniami UX (ang. User Experience). Zbieranie i zapisywanie informacji o kolejnych krokach może być wykorzystane do skierowania użytkownika do zachowań oczekiwanych przez właściciela strony internetowej. Najczęstszym przykładem są sklepy internetowe. Na urządzeniu klienta zapisywane są grupy artykułów, które wyświetlił odwiedzający sklep. W przypadku kolejnego podłączenia do strony zostają wyświetlane produkty z wcześniej wyświetlonej grupy w określonych miejscach na stronie internetowej, które zawierają reklamy lub podpowiedzi.

22 marca 2013 roku weszły w życie znowelizowane przepisy ustawy Prawa Telekomunikacyjnego. Najważniejsza zmiana nakłada na właścicieli serwisów internetowych obowiązek informowania użytkowników strony o plikach, które serwis umieszcza w komputerze użytkownika oraz o tym, w jakim celu to robi.

3.2. Pliki supercookies

Pliki super cookies spełniają te same funkcje, co zwykłe ciasteczka, ponieważ mogą zawierać prawie wszystkie informacje, w tym historię przeglądania, dane uwierzytelniające lub dane kierowane do reklamodawców.

Zasadniczą różnicą między plikami cookies, a super cookies jest fakt, że te ostatnie mogą być zapisywane są w ukrytej lokalizacji na komputerze klienta, przez co stają się trudne do usunięcia. Dodatkowo nie są stosowane mechanizmy automatycznego, okresowego usuwania takich plików. Biorąc pod uwagę rozmiar pojedynczego pliku super cookie często znacznie większy niż 100kB (w przypadku zwykłych plików cookies jest to z reguły ok. 4kB) przy intensywnym korzystaniu z usług sieci może to mieć wpływ na wykorzystanie zasobów w urządzeniu użytkownika. Dodatkowe zagrożenie powoduje fakt, że informacje zawarte w plikach super cookies dostępne są nie tylko w obrębie witryny uruchamianej na konkretnej przeglądarce, ale w obrębie każdej domeny, niezależnie od przeglądarki, na której zostały zapisane.

3.3. Pliki zombie cookies i evercookie

Pliki określane mianem „zombie cookies" to ciasteczka samoodnawialne po zniszczeniu (np. skasowaniu przez użytkownika). Kopie zapasowe ciasteczek tworzone są poza dedykowaną pamięcią przeglądarki przeznaczoną do zapamiętywania plików cookies lub w specyficznych przypadkach - online. Pliki tego typu mogą zostać zapisane na urządzeniu klienta bez jego wiedzy i akceptacji, nawet przy włączonej opcji blokady zapisu plików cookies w przeglądarce.

Kolejną odmianą plików gromadzących informacje o użytkowniku będące implementacją w języku JavaScript narzędzi informatycznych (API) do tworzenia tzw. permanentnych ciasteczek. Ten rodzaj pliku wykrywa akcję usunięcia ciasteczka i tworzy je na nowo z kopii zapasowej. Kopie zapasowe są tworzone w zależności od dostępności modułu na urządzeniu klienta [4] w następujących miejscach:

- Standardowych plikach cookies HTTP
- HTTP Strict Transport Security (HSTS) Pinning
- Local Shared Objects (Flash Cookies)
- Silverlight Isolated Storage
- Automatycznie generowanych wartościach RGB przy pomocy force-cached PNG i HTML5 Canvas
- Web History
- HTTP ETags
- Web cache
- window.name caching
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite
- HTML5 IndexedDB
- Java JNLP PersistenceService
- Java CVE-2013-0422 exploit (applet sandbox escaping)

3.4. Skrypty JavaScript

Najczęściej wykorzystywanym mechanizmem do pozyskiwani informacji o użytkowniku są skrypty napisane w języku JavaScript. Obecnie około 95% stron www posiada zaimplementowane takie skrypty [2]. Podstawową funkcjonalnością jest wprowadzenie interaktywności między użytkownikiem a witryną. Aplikacje JS pomagają w identyfikacji użytkownika, ale poprzez dostęp do informacji o specy-

ficznej konfiguracji komputera, systemu operacyjnego, czy wersji przeglądarki, można dokonać implementacji, która stanie się narzędziem do inwigilacji i przekazywania informacji o użytkowniku.

Przykład takiej implementacji prezentuje artykuł [15], w zaprezentowane dwa podstawowe aspekty związane z poprawną identyfikacją użytkownika – pozytywną cechą unikalnej identyfikacji jest możliwość ochrony przed kradzieżą sesji zestawionej, np. z bankiem. Z drugiej jednak strony, można wykorzystać takie narzędzie do permanentnego śledzenia aktywności użytkownika.

Mechanizm „Fingerprinting", czyli oznaczanie unikalnych elementów wykorzystywanych przez klienta, w przytoczonym artykule dotyczy nie tylko rozpoznania wersji przeglądarki użytkownika, ale także określa wersji systemu operacyjnego i architektury komputera.

4. Wybrane metody ograniczające możliwość zbierania informacji o użytkowniku

Twórcy rozwiązań i technologii informatycznych, analizując podstawowe obszary architektury komputera oraz aplikacji internetowych monitorowane pod kątem zbierania informacji o użytkowniku, opracowali metody utrudniające lub ograniczające dostęp do tego typu informacji przez podmioty do tego nieuprawnione.

4.1. Tryb incognito

Obecnie każda przeglądarka posiada tzw. tryb „incognito". Według założeń producentów taki tryb prywatny zapewnia anonimowość przeglądającemu strony www. Przeglądarka z aktywną tą funkcjonalnością nie gromadzi historii przeglądania, powinna blokować zapis plików cookies oraz tworzyć czyste środowisko za każdym razem kiedy jest uruchamiana. Żaden tekst podany w polach tekstowych lub w polach wyszukiwania nie powinien zostać dodany do listy podpowiedzi automatycznego wypełniania formularzy. Tryb prywatny powinien odseparować dane przeglądarki w trybie standardowym i nie dopuścić do wycieku wrażliwych informacji [5].

Producenci przeglądarek internetowych dodają informację o trybie prywatnym, która mówi, że tryb ten nie zabezpiecza podatności związanych z architekturą protokołu TCP/IP w związku z czym dostawcy Internetu jak i osoby trzecie nasłuchujące komunikacji wychodzącej z komputera użytkownika trybu prywatnego mogą śledzić jego aktywność w sieci. Badania opublikowane w pracy [7] pokazują, że w roku 2107 jedynie przeglądarka Opera była w stanie zablokować większość podatności związanych z wyciekiem tożsamości użytkownika.

DIGITAL CITIZEN	Google Chrome	Internet Explorer	Microsoft Edge	Mozilla Firefox	Opera
Deleted data and files					
Cookies	Yes	Yes	Yes	Yes	Yes
Data from forms	Yes	Yes	Yes	Yes	Yes
Temporary files (cache)	Yes	Yes	Yes	Yes	Yes
Browsing history	Yes	Yes	Yes	Yes	Yes
Search history	Yes	Yes	Yes	Yes	Yes
Recovers closed tabs	No	Yes	No	Yes	No*
Disables add-ons and toolbars	Yes	Yes	Yes	No	Yes
Blocks trackers and advertising	No	Optional	No	Yes	Optional
VPN encryption	No	No	No	No	Optional

Rys. 2. Badania skuteczności trybu prywatnego najbardziej popularnych przeglądarek internetowych w 2017 r.

4.2. Dodatki do przeglądarek

Dodatki (ang. Plug-in) do przeglądarek www to miniaplikacje rozszerzające domyślne funkcjonalności. Takie rozszerzenia powinny być instalowane wyłącznie ze sklepów producentów. Z jednej strony mogą utworzyć interfejs ułatwiający odczytywanie wiadomości email, dodać przycisk pobierający pliki audio i wideo bezpośrednio z odtwarzacza HTML5 umieszczonego na stronie www, ale także blokować mechanizmy tworzące wirtualne odciski palców, czyli informacje pozwalające na identyfikację użytkownika na podstawie jego aktywności.

Jednym z takich dodatków jest Ghostery, którego głównym zadaniem jest blokowanie skryptów śledzących aktywność użytkownika oraz wtyczek umożliwiających korzystanie z portali społecznościowych czy komentarzy (tzw. trackerów) [6]. Wpisanie adresu strony do przeglądarki z tym uruchomionym dodatkiem powoduje przeanalizowanie i wyświetlenie zablokowanej listy skryptów śledzących. Szczegółową listę trackerów wraz z ich opisem i dokładnym adresem URL można zobaczyć klikając w ikonę Ghostery (zawierającą informację o liczbie blokad), następnie przechodząc do zakładki Widok szczegółowy.

Podobnie jak w przypadku aplikacji antywirusowych, prawidłowe działania dodatku Ghostery wymaga nieustannej aktualizacji biblioteki skryptów śledzących.

Rys. 3. Widok szczegółowy z listą zablokowanych skryptów śledzących

5. Podsumowanie

Zaprezentowane w artykule zagrożenia oraz metody ochrony informacji o aktywności użytkownika implementowane w przeglądarkach internetowych pokazały, że nie istnieją rozwiązania pozwalające na wyeliminowanie tego typu zagrożeń wyłącznie za pomocą oprogramowania instalowanego na urządzeniach końcowych użytkownika. Jest to jedyny element systemu informatycznego opartego na usługach w sieci globalnej na którym klient posiada częściową kontrolę. Ciągłe mutacje skryptów śledzących użytkownika [8] zmuszają producentów przeglądarek oraz dodatków w postaci rozszerzeń zapewniających ochronę tożsamości do aktualizowania swojego oprogramowania. Przygotowanie rozwiązania uniwersalnego wymagałoby opracowania analizatora kontekstowo-zależnych skryptów uruchamianych przez przeglądarki www. Jest to oczywiście niemożliwe, gdyż wymagałoby implementacji automatu o nieskończonej liczbie stanów. Metody

ochrony anonimowości w sieci Internet powinny stanowić większy system pośredniczący w dostępie do usług sieciowych. W takim rozwiązaniu odpowiednio skonfigurowane urządzenie klienta jest tylko częścią mającą z jednej strony funkcję prezentacyjną a z drugiej separującą dane przechowywane na komputerze użytkownika przed usługodawcą.

Bibliografia

1. Główny Urząd Statystyczny, Społeczeństwo informacyjne w Polsce 2017: https://stat.gov.pl/download/gfx/portalinformacyjny/pl/defaultaktualnosci/5497/2/7/1/spoleczenstwo_informacyjne_w_polsce_w_2017.pdf
2. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints, Pierre Laperdrix; Walter Rudametkin; Benoit Baudry, 2016 IEEE Symposium on Security and Privacy
3. https://w3techs.com/technologies/details/cp-javascript/all/all
4. Sandvine, Global Internet Phenomena Raport, 2014: https://www.sandvine.com/downloads/general/globalinternet-phenomena/2014/1h-2014-global-internetphenomena-report.pdf
5. https://support.mozilla.org/pl/kb/Przegl%C4%85danie%20w%20trybie%20prywatnym
6. Sam Macbeth, Tracking the Trackers: Analysing the globaltracking landscape with GhostRank
7. https://www.digitalcitizen.life/what-is-private-browsing-which-browser-is-best
8. A. Narayanan, D. Reisman, S. Englehardt, C. Eubank, P. Zimmerman, Cookies that give you away: Evaluating the surveillance implications of web tracking
9. New Web Code Draws Concern Over Privacy Risks, https://www.nytimes.com/2010/10/11/business/media/11privacy.html?hp
10. Soltani, S. Canty, Q. Mayo, L. Thomas, C.J. Hoofnagle, Flash Cookies and Privacy
11. J. Mayer and J. Mitchell "Third-Party Web Tracking: Policy and Technology", IEEE Symposium on Security and Privacy, 2012
12. Karwowski Maciej, "Ocena skuteczności mechanizmów zapewnienia prywatności w sieci Internet", praca dyplomowa wrzesień 2014
13. M. Ayenson, D. Wambach, A. Soltani, N. Good, C. Hoofnagle " Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning„
14. G. Aggarwal, E. Burzstein, C. Jackson, D. Boneh, "An Analysis of Private Browsing Modes in Modern Browsers
15. K. Mowery „Fingerprinting Information in JavaScript Implementations" – Proceedings of W2SP 2011. IEEE Computer Society, May 2011.

Streszczenie

Praca prezentuje metody śledzenia użytkownika przeglądającego Internet poprzez właściwości przeglądarek internetowych jak i słabości wynikające z architektury sieci Internet. Ponadto, w pracy przedstawiono zestaw narzędzi maskujących tożsamość użytkownika. W pracy przedstawiono innowacje w HTML5 zapewniające dostęp do wysoce wyróżniających się atrybutów użytkownika, w szczególności za pomocą interfejsu Cookie, który opiera się na wielu warstwach systemu użytkownika.

Abstract

The work presents methods of tracking the user browsing the Internet through the properties of web browsers as well as weaknesses resulting from the architecture of the Internet. In addition, the work presents a set of tools masking the user's identity. The paper presents innovations in HTML5 that provide access to highly-distinguished attributes of the user, in particular using the Cookie interface, which is based on many layers of the user's system.

Keywords: Vulnerabilities, web applications, security, cookies, surveillance

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 47-54

Grzegorz Górski
Marcin Nowacki
Zakład Systemów Multimedialnych i Sztucznej Inteligencji
Wydział Elektroniki i Informatyki
Politechnika Koszalińska

Analiza zagrożeń bezpieczeństwa dla współczesnych platform mobilnych

Słowa kluczowe: zaufany system informatyczny, urządzenia mobilne, analiza bezpieczeństwa

1. Wprowadzenie

Powszechna dostępność telefonii komórkowej istotnie zmieniła sposób funkcjonowania ludzi na całym świecie. O ile początkowo telefonów komórkowych używano główne do realizacji połączeń głosowych oraz wysyłania wiadomości tekstowych, to obecnie pełnią one rolę mobilnych komputerów. W telefonach przechowywane i przetwarzane są zarówno informacje biznesowe jak i prywatne w większości dostępnych postaci i formatów. Powszechny dostęp do informacji jest obecnie traktowany jako prawo jednostki gwarantowane przez państwo.

Nie byłoby to oczywiście możliwe bez nieustannego rozwoju technologicznego, tj. nowych procesów wytwarzania układów scalonych, nowych modeli coraz bardziej wydajnych procesorów oraz innowacyjnych narzędzi i technologii informatycznych wykorzystywanych do tworzenia rozmaitych usług w sieci Internet. Prowadzone prace badawcze dotyczą także zagadnień bezpieczeństwa danych przetwarzanych na urządzeniach mobilnych. Niestety w tej kwestii świadomość użytkowników telefonii komórkowej jest ciągle bardzo niska. W wielu przypadkach nie są oni świadomi, że korzystanie z publicznych sieci istotnie zwiększa ryzyko nieautoryzowanego dostępu do poufnych informacji zgromadzonych w telefonie.

2. Powszechne zagrożenia w systemach mobilnych

W artykule opisano wybrane grupy zagrożeń (klasy ataków), które w większości mogą być wykonane bez świadomości użytkownika, nawet w przypadku jeśli posiada on ponadprzeciętną wiedzę dotyczącą szeroko pojętej informatyki.

Do tych najbardziej popularnych należą:

- błędy w mobilnych systemach operacyjnych,
- korzystanie z sieci internetowej poprzez niezabezpieczone punkty dostępu Wi-Fi, m.in. w miejscach publicznych,
- instalowanie aplikacji mobilnych z niezaufanych źródeł,
- niedostateczna weryfikacja aplikacji przez tzw. „Sklepy aplikacji",
- brak świadomości użytkowników wobec zagrożeń, które mogą wystąpić w skutek niewłaściwego korzystania z urządzeń mobilnych.

Poza wymienionymi powyżej zagrożeniami, informacje zgromadzone na urządzeniach mobilnych - podobnie jak w przypadku komputerów osobistych – są podatne na znane rodzaje ataków takie jak: phishing (wyłudzenie informacji od użytkownika), backdoor (nieautoryzowane kanały dostępu do informacji), malware (modyfikacje kodu aplikacji lub podłączane złośliwego kodu do aplikacji użytkowych) oraz na ataki wykorzystujące komponenty sprzętowe telefonów komórkowych np. kamerę, czytnik linii papilarnych czy też skaner twarzy.

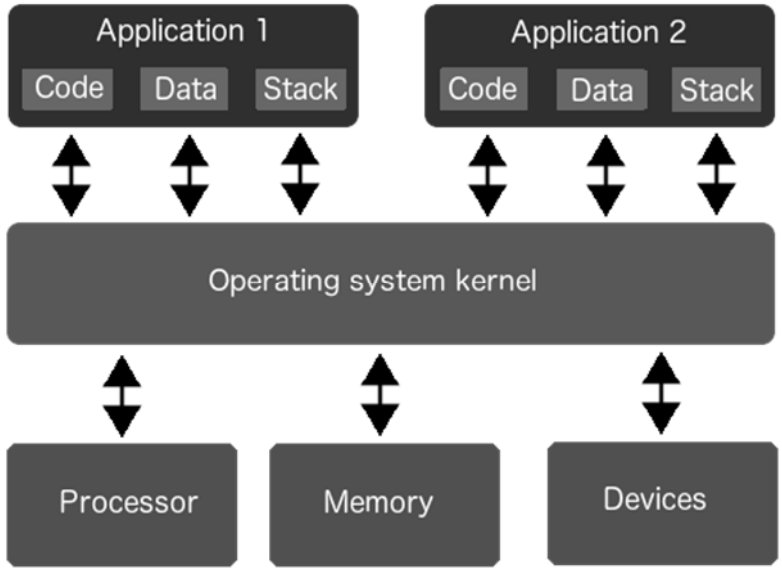
3. Zaufany system komputerowy

Celem każdej analizy zagrożeń jest wskazanie słabych ogniw systemów informatycznych, które mogą być wykorzystane do stworzenia nieautoryzowanego kanału dostępu do danych. Pierwotnym problemem jest określenie oczekiwań użytkowników, którzy chcą korzystać z urządzeń mobilnych przy jednoczesnym zapewnieniu akceptowalnego poziomu bezpieczeństwa, a zatem oczekują produktów, do których będą mieli zaufanie. Istnieje wiele definicji zaufanego system komputerowego. Jednakże każda z nich wskazuje, że jest to dany zestaw sprzętu i oprogramowania, na którym jego użytkownik może w pełni polegać oraz posiada nad nim całkowitą kontrolę.

Taka prosta definicja jest niezmiernie trudna do realizacji w tzw. otwartych platformach mobilnych, w których użytkownicy wymieniają się aplikacjami, przesyłają pomiędzy sobą informacje korzystając jednocześnie z publicznych, otwartych sieci pakietowych. Poniżej przedstawiono kilka klas ataków, które są możliwe do realizacji, a które pozostają poza kontrolą właściciela telefonu komórkowego.

3.1. Atak od środka

Większość współcześnie stosowanych systemów mobilnych jest zbudowanych w oparciu o architekturę z jądrem systemu operacyjnego (Rys. 1). Ten element systemu mobilnego pośredniczy w komunikacji pomiędzy zasobami sprzętowymi takimi jak: procesor, pamięć operacyjna, urządzenia zewnętrzne (wyświetlacz, kamera, żyroskop, mikrofon itp.) oraz aplikacjami (zarówno tymi zainstalowanymi przez użytkownika jak i dostarczonymi razem z systemem operacyjnym). Jądro systemowe posiada pełny dostęp do segmentu kodu, danych oraz stosu każdej działającej w systemie operacyjnym aplikacji.



Rys. 1. Architektura systemu operacyjnego opartego na jądrze systemowym

Taka architektura systemu operacyjnego sprawia, że możliwe jest wykonanie tzw. „ataku od środka" (Man-Inside-Attack). Poniżej zaprezentowano przykładowy jego scenariusz.

Haker umieszcza w urządzeniu mobilnym „złośliwą" aplikację lub skrypt, którego zadaniem jest uzyskanie kontroli nad jądrem systemowym. Takie działanie może zostać wykonane, np. poprzez atak typu phishing lub poprzez backdoor. W momencie, gdy taka kontrola zostanie uzyskana, możliwe jest przeprowadzanie analizy przesyłanych danych pomiędzy jądrem a aplikacją mobilną. Przeprowadzenie tego ataku jest zadaniem, które wymaga od hakera znajomości struktury warstwy jądra systemu operacyjnego będącego celem ataku. Istnieją również inne sposoby ataków, które mogą być przeprowadzone w prostszy sposób.

3.2. Atak wykonany przez inną aplikację

We współczesnych systemach mobilnych podjęto próbę wzajemnej separacji uruchamianych aplikacji. Do tego celu wykorzystywane jest rozwiązanie typu „sandbox", w którym każdy program posiada swoje środowisko uruchomieniowe. Utrudnia to także zablokowanie systemu poprzez wykorzystanie wszystkich zasobów przez jedną wadliwie działającą aplikację. Celem twórców takiego rozwiązania było zablokowanie bezpośredniego dostępu do danych pomiędzy aplikacjami. Projektanci systemów operacyjnych nie wyeliminowali wszystkich metod nieautoryzowanego dostępu do zasobów współdzielonych przez poszczególne programy. Taki atak może być wykonany, np. poprzez nagrywanie ekranu urządzenia. W sklepach aplikacji mobilnych dostępne są aplikacje, które umożliwiają użytkownikowi rejestrację interakcji użytkownika z inną aplikacją. Aplikacje nagrywające posiadają swój interfejs użytkownika, który jest widoczny w trakcie ich działania. To powoduje, że użytkownik posiada nad taką aplikacją „kontrolę", ale nie może być w pełni pewny, czy ekran jego urządzenia nie jest szpiegowany.

Rys. 2. Ekran autoryzacji na stronie internetowej banku

Skoro jednak istnieje mechanizm dostępu jednej aplikacji do zasobów, które powinny być wyłącznie używane przez drugą aplikację, to możliwe jest stworzenie takiego rodzaju aplikacji o podobnym działaniu, która nie będzie prezentowała jakiegokolwiek interfejsu użytkownika oraz będzie nagrywała ekran w sposób niewidoczny. Złośliwa aplikacja może być uruchomiona w tle jako proces o nazwie, np. „Aktualizacja systemowa". W tej sytuacji użytkownik nie jest w stanie wykryć zagrożenia. Skutki takiego rodzaju ataku mogą być bardzo poważne, np. uzyskanie poufnych identyfikatorów dostępu użytkownika podczas jego logowania do bankowości elektronicznej. Klawiatura używana w urządzeniach mobilnych jest widoczna na ekranie, więc aplikacja nagrywająca ekran może w łatwy sposób śledzić sekwencję wybieranych w niej znaków (Rys. 2).

3.3. Atak z wykorzystaniem warstwy sprzętowej

Większość używanych systemów mobilnych zakłada, że najważniejszym atrybutem informacji jest jej dostępność. Użytkownicy wybierają telefony z coraz większymi zasobami pamięci, aby wszystkie istotne informacje były dostępne na każde żądanie. Potencjalny dostęp do takich informacji przez producentów urządzeń mobilnych staje się coraz większą przewagą konkurencyjną umożliwiającą lepszy dobór produktów pod upodobania klientów, skuteczne dostarczanie reklam pozycjonowanych prywatnymi danymi użytkownika. Nieautoryzowany dostęp do takich informacji może być realizowany przez nieudokumentowane funkcjonalności mobilnych systemów operacyjnych. Pośrednio mogą to zauważyć także dociekliwi użytkownicy korzystając z najbardziej popularnych wyszukiwarek, których wyniki są bardziej spersonalizowane w przypadku, gdy wyszukujący używa zarówno przeglądarki jak i systemu operacyjnego dostarczanego przez producenta usługi wyszukującej. W 2018 roku zostały opublikowane prace przedstawiające realne możliwości przeprowadzenia ataku z wykorzystaniem warstwy sprzętowej. Potencjalnie także producenci układów scalonych mogą mieć dostęp do danych przetwarzanych na urządzeniach mobilnych.

Atak określany terminem Meltdown polega na pokonaniu zabezpieczeń pomiędzy systemem operacyjnym a działającymi w nim aplikacjami. Dzięki temu program posiada możliwość uzyskania dostępu do pamięci procesora, a tym samym do danych zapisanych w innych programach oraz systemie operacyjnym. Tego typu podatność jest luką sprzętową, która dotyczy najbardziej popularnych procesorów z rodziny Intel, AMD oraz ARM, a zatem może być zrealizowana na każdym urządzeniu bez względu na system operacyjny. Producenci systemów operacyjnych przygotowali stosowne aktualizacje oprogramowania, które chronią przed tym zagrożeniem, lecz ich zainstalowanie ma zauważalny wpływ na wydajność działania systemu.

Spectre to kolejny rodzaj ataku wykorzystujący warstwę sprzętową, który wynikiem jest przełamanie zabezpieczeń pomiędzy środowiskami uruchomienio-

wymi aplikacji. Umożliwia on uruchomienie w systemie operacyjnym procesu posiadającego uprawnienia zwykłego użytkownika, który w sposób nieautoryzowany uzyskuje dostęp do segmentu kodu, danych lub stosu innych procesów działających w systemie. W tym ataku użyto powszechnie stosowanego w procesorach mechanizmu przewidującego instrukcje warunkowe oraz rozgałęzienia programów w celu przyśpieszenia realizacji kolejnych instrukcji. W przeciwieństwie do ataku Meltdown, Spectre jest nieusuwalny poprzez modyfikację oprogramowania systemowego. Jedynym rozwiązaniem jest wymiana procesora na nową generację, co w przypadku urządzenia mobilnego jest trudne lub nawet niemożliwe do realizacji, gdyż producenci nie oferują możliwości tego typu modyfikacji urządzeń mobilnych.

4. Urządzenie mobilne jako system zaufany

Zaliczenie urządzenia mobilnego do kategorii zaufanych systemów komputerowych wymagałoby spełnienia warunków podanych definicji systemu zaufanego. Przedstawione powyżej klasy ataków, które są reprezentatywne dla konkretnych luk bezpieczeństwa opisywanych w literaturze dowodzą, że współczesne systemy mobilne nie mogą być traktowane jako systemy zaufane.

Problematyka bezpieczeństwa informacji przetwarzanych przez urządzenia mobilne jest istotnym tematem badań zmierzających do stworzenia rozwiązań, które umożliwią zaklasyfikowanie urządzeń mobilnych do grupy systemów zaufanych.

Jednym z potencjalnych rozwiązań może być użycie zewnętrznej karty mikroprocesorowej (smart card), wyposażonej w pamięć do przechowywania danych oraz mikroprocesor kryptograficzny do ich przetwarzania. Karta za pośrednictwem zewnętrznego czytnika kart może być podłączona do urządzenia mobilnego za pomocą specjalnego złącza (USB) lub przy użyciu technologii bezprzewodowej (NFC). Na karcie może być przechowywany sekret np. klucz prywatny wykorzystany m.in. w procesie autoryzacji w systemie bankowym lub systemach opieki medycznej. Karty mikroprocesorowe wyposażone w koprocesor kryptograficzny wraz z chronioną przestrzenią pamięci uniemożliwiają bezpośredni dostęp do przechowywanego sekretu. Próba takiego dostępu wiązałaby się ze zniszczeniem struktury układu i nieodwracalną utratą sekretu. Możliwe jest natomiast wykonywanie operacji przez koprocesor kryptograficzny z użyciem sekretu i przesłanie wyników do urządzenia mobilnego. Standardowe czytniki danych nie są jednak wyposażone w klawiatury dedykowane niezbędne do wpisania kodu PIN niezbędnego do autoryzacji operacji wykonywanej przez kartę mikroprocesorową. Użycie klawiatury ekranowej z urządzenia mobilnego, czyni jednak rozwiązanie podatnym na atak polegający na przechwyceniu kodu PIN i wykonaniu operacji bez wiedzy właściciela karty, jeśli tylko karta z czytnikiem ma

zestawiony kanał komunikacyjny z urządzeniem mobilnym. Jest to modyfikacja powszechnie znanego ataku na bankomaty.

5. Podsumowanie

Zaprezentowana w artykule analiza bezpieczeństwa współczesnych aplikacji mobilnych pokazała, że istnieją podatności, które są bardzo trudne lub niemożliwe do wyeliminowania za pomocą elementów wbudowanych w telefon komórkowy zarówno w odniesieniu do dodatkowego sprzętu jak i oprogramowania (mechanizmów zaimplementowanych w mobilnym systemie operacyjnym). Zawarta w pracy definicja systemu zaufanego stawia wysokie wymagania, które potencjalnie mogłyby spełnić rozwiązania z niezależnym urządzeniem (komponentem sprzętowym) realizującym procesy autoryzacji i kodowania danych.

Bibliografia

1. Syed Farhan Alam Zaidi, Munam Ali Shah, Muhammad Kamran, Qaisar Javaid, Sijing Zhang „*A Survey on Security for Smartphone Device*", International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4 (2016).
2. Murat Yesilyurt, Yildiray Yalman, „*Security Threats on Mobile Devices and their Effects: Estimations for the Future",* International Journal of Security and Its Applications Vol. 10, No. 2 (2016).
3. N. Asokan, Jan-Erik Ekberg, Kari Kostiainen, Anand Rajan, Carlos Rozas, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann, „*Mobile Trusted Computing*", Proceedings of the IEEE | Vol. 102, No. 8 (2014).
4. Jalaluddin Khan, Haider Abbas, Jalal Al-Muhtadi, „*Survey on Mobile User's Data Privacy Threats and Defense Mechanisms*", Procedia Computer Science 56 (2015).
5. H Wonjong Kim, Seungchul Kim, Younghwan Bae, Sungik Jun, Youngsoo Park, and Hanjin Cho, "*A Platform-Based SoC Design of a 32-Bit Smart Card*", ETRI Journal, Volume 25, Number 6 (2003).
6. M, Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh,, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M.Hamburg: „*Meltdown: Reading Kernel Memory from User Space*" https://meltdownattack.com/
7. „Spectre and Meltdown processor flaws threaten billions of computers and mobile devices", Computer Fraud & Security Vol. 2018, Issue 1, Elsevier
8. G. Górski, „*Token programowy dla urządzeń mobilnych wspierający silne uwierzytelnianie użytkowników z wykorzystaniem infrastruktury klucza publicznego*", Przegląd Telekomunikacyjny 8-9/2013, str. 1231-1236, 2013

9. G. Górski, „*Algorytm weryfikacji haseł wykorzystujący badanie integralności danych*”, Przegląd Telekomunikacyjny 8-9/2012, str. 842-848, 2012
10. G. Gorski, „*System płatności mobilnych wykorzystujący biometryczną identyfikację użytkowników oraz infrastrukturę klucza publicznego*”, - Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne –No. 8-9/2015 – str, 1489-1495, 2015

Streszczenie

W artykule zostały opisane wybrane grupy zagrożeń (klasy ataków) dla współczesnych systemów mobilnych, które w większości mogą być wykonane bez wiedzy użytkownika. Poszczególne podatności sklasyfikowano w trzech grupach jako zagrożenia pochodzące od innych aplikacji działających na urządzeniu mobilnym, wynikające z niedoskonałości lub ukrytych funkcjonalności systemu operacyjnego oraz ataków z wykorzystaniem warstwy sprzętowej telefonu. Wprowadzono także definicję oraz wymagania jakie stawiane są dla systemu zaufanego. Obecny poziom zabezpieczeń w konfrontacji z zaprezentowanymi podatnościami uniemożliwia w obecnym stanie zaklasyfikowanie systemów mobilnych do grupy systemów zaufanych.

Abstract

The article describes selected groups of attacks for modern mobile systems which can mostly be executed without the user awareness. The particular vulnerabilities have been classified into 3 groups as threats from other applications executed in a mobile device resulting from defects or the other hidden functionalities present in the operating system and the attacks executed by using the hardware layer. A definition and requirements for a trusted system have been introduced. The current level of security in confrontation with the presented vulnerabilities does not allow in the current state to classify the mobile systems into the group of trusted systems.

Keywords: Trusted computer system, mobile devices, security analysis

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 55-61

Jakub Ślepecki
Michał Rydzewski
Paweł Kisiel
Paweł Poczekajło
Studenckie Koło Naukowe Pasjonatów Elektroniki
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
ul. JJ Śniadeckich 2, 75-453 Koszalin

Konsola do gier bazująca na płytce Arduino Due

Słowa kluczowe: konsola do gier, automat do gier, Arduino, ARM, SAM, VGA, pong, bullethell

1. Wstęp

Konsole do gier stanowią niezwykle popularną i ciągle rozwijaną formę rozrywki [1][2]. Urządzenia te przyjmują najróżniejsze formy od klasycznych automatów z salonów gier, przez konsole do użytku domowego, po małe urządzenia przenośne. Ich głównym przeznaczeniem jest umożliwianie użytkownikom czerpania przyjemności z różnego typu rozgrywki.

Koło Pasjonatów Elektroniki (działające na Wydz. Elektroniki i Informatyki Politechniki Koszalińskiej), mające pewne doświadczenie w budowie gier typu „arcade" [3], skonstruowało własną konsolę stylizowaną na klasyczny automat oraz napisało na nią dwie gry. Konstrukcja powstała w oparciu o płytkę prototypową z układem typu SAM (Smart ARM-based Microcontroller).

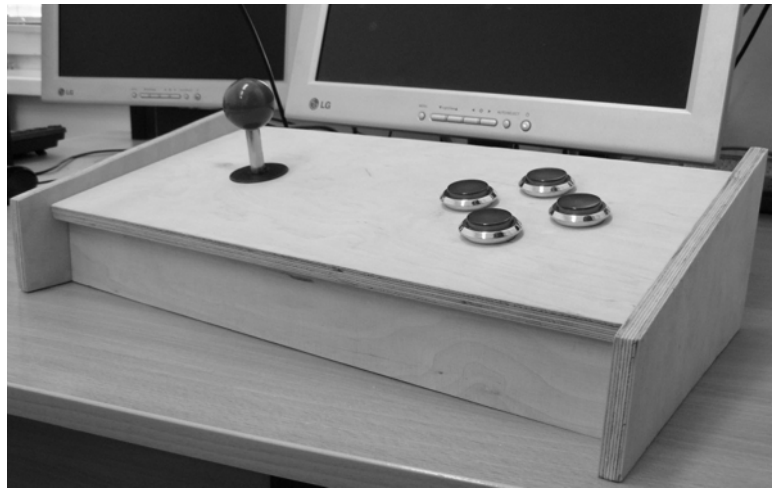
2. Opis konsoli

Wykonana konsola składa się z głównego modułu sterującego, kontrolera oraz modułu obsługi VGA. Jako układ sterujący wykorzystano płytkę Arduino Due z mikrokontrolerem AT91 SAM3X8E, którego parametry zebrano w tabeli 1. Jest on odpowiedzialny za działanie aktualnie uruchomionej gry, przyjmowanie informacji z kontrolera oraz przekazywanie danych o grafice do wyświetlenia.

Płytka ta jest umieszczona wewnątrz obudowy ze sklejki, na której umieszczono kontroler. Składają się na niego joystick oraz cztery podświetlane przyciski. Ponadto, w bocznej ściance obudowy znajduje się dodatkowy przycisk resetujący konsolę. Gotowy automat przedstawiony jest na rysunku 1. Urządzenie podłączone jest do zewnętrznego monitora za pośrednictwem złącza VGA, co dodatkowo wymagało przygotowania i zastosowania specjalnego modułu adaptera.

Tabela 1. Podstawowe parametry AT91 SAM3X8E [4]

Mikroprocesor	ARM Cortex-M3
Częstotliwość	84 MHz
Ilość pamięć Flash	2 x 256 KB
Ilość pamięci RAM	94 KB
Ilość pamięci ROM	16 KB
Temperatura pracy	od -40 °C do 85 °C
Napięcie zasilania	1,62V - 3,6V
Interfejsy komunikacyjne	UART, SPI , I2C, USB, CAN



Rys. 1. Gotowa konsola do gier

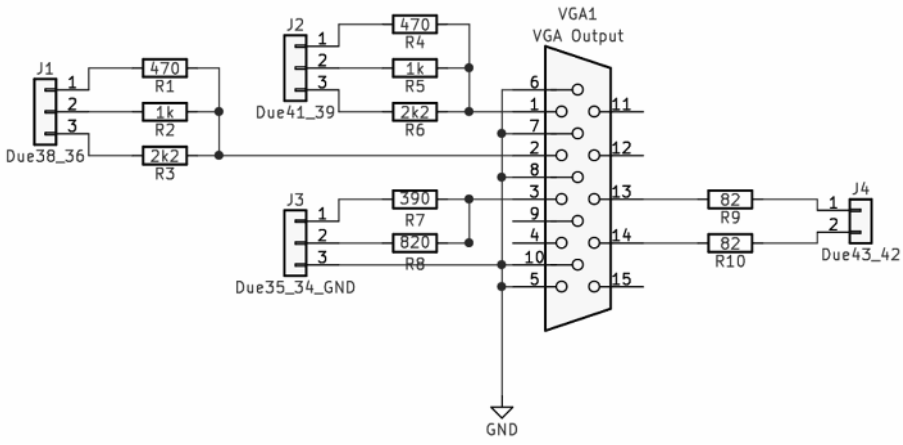
3. Obsługa VGA

Do obsługi monitora poprzez złącze VGA wykorzystano specjalnie przygotowany adapter w postaci samodzielnie zaprojektowanej i wykonanej płytki PCB. Jego zadaniem jest przekształcanie sygnału cyfrowego z wyjść Arduino Due

na sygnał analogowy o określonych poziomach napięcia (tabela 2). Układ został wykonany zgodnie ze schematem widocznym na rys. 2.

Tabela 2. Poziomy napięcia na pinach R,G i B portu VGA dla różnych konfiguracji wybranych pinów wyjściowych procesora

Numery pinów na płytce Arduino DUE	Możliwa konfiguracja wskazanych pinów							
	[0,0,0] / [0,0]	[0,0,1] / [0,1]	[0,1,0] / [1,0]	[0,1,1] / [1,1]	[1,0,0]	[1,0,1]	[1,1,0]	[1,1,1]
[41,40,39]	0V	2,88V	2,38V	1,96V	1,34V	0,92V	0,42	3,3V
[38,37,36]	0V	2,88V	2,38V	1,96V	1,34V	0,92V	0,42	3,3V
[35,34]	0V	2,24V	1,06V	3,3V	-	-	-	-



Rys. 2. Schemat zbudowanego adaptera VGA

Oprócz adaptera, do obsługi VGA przez Arduino zastosowano również biblioteki DueVGA [5]. Ze względu na ograniczoną pamięć układu i wynikający z tego brak podwójnego buforowania, rozdzielczość obrazu została zawężona do 320×240 pikseli. Biblioteka DueVGA udostępnia 256 kolorów, zatem obraz tworzony z jej użyciem przypomina 8-bitowe gry, doskonale pasując do stylu zaprojektowanej konsoli.

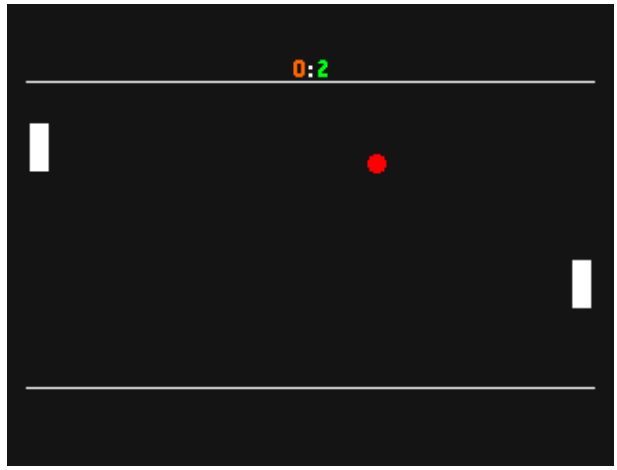
4. Gry

Na zbudowany automat zostały napisane dwie gry. Pierwsza z nich to prosta gra typu „pong”. Na etapie projektowania, posłużyła przede wszystkim do testowania obsługi VGA. Docelowym programem była druga gra, tym razem z rodzaju

„bullethell". Należy zaznaczyć, że nie jest możliwa zmiana gry podczas działania konsoli, gdyż stanowią one odrębne programy, z których tylko jeden może znajdować się w danej chwili w ograniczonej pamięci układu.

4.1. Pong

Gra oferuje prostą rozgrywkę dla dwóch graczy, polegającą na odbijaniu piłki przy użyciu umieszczonych po bokach ekranu dwóch paletek. Każdy gracz kontroluje ruch jednej z nich. Jeśli graczowi nie uda się odbić nadlatującej piłki, to przeciwnik otrzyma punkt. Do sterowania paletkami służy joystick oraz dwa przyciski. Po każdym zdobytym punkcie piłka pojawia się ponownie na środku ekranu i oczekuje, aż któryś z graczy poruszy paletką. Wówczas piłka zaczyna się poruszać w losowo wybranym kierunku. Gra wyświetla w górnej części ekranu wynik, który jest resetowany po uzyskaniu przez jednego z graczy 10 punktów. Wyzerowanie licznika oznacza koniec partii. Na rysunku 3 przedstawiono zrzut ekranu z gry Pong.

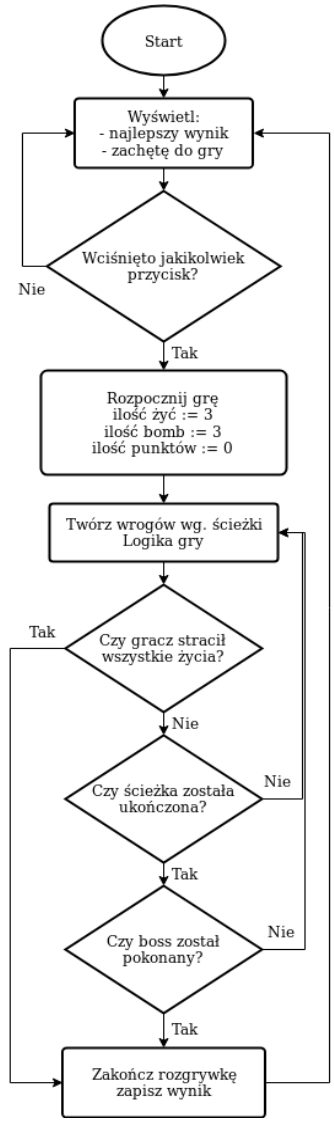


Rys. 3. Zrzut ekranu z gry Pong

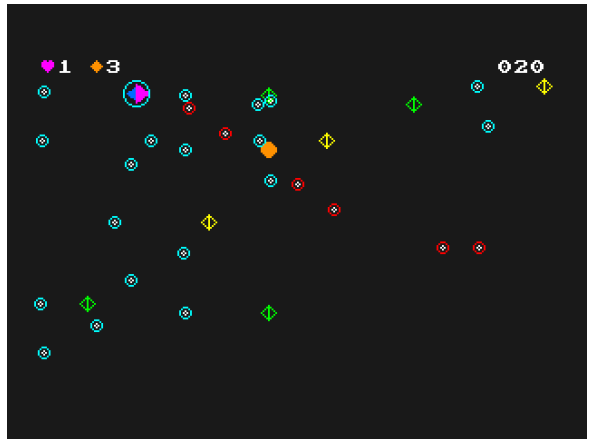
4.2. Bullethell

To dynamiczna i wymagająca rozgrywka dla jednej osoby. Gracz za pomocą joysticka steruje statkiem kosmicznym. Celem gry jest niszczenie przeciwników nadlatujących zza prawej krawędzi ekranu i unikanie wystrzeliwanych przez nich pocisków. Gracz może strzelać, użyć bomby likwidującej znajdujące się na ekranie pociski lub zmniejszyć swą prędkość, by uzyskać większą precyzję manewrów. Akcje te są przypisane do przycisków automatu. W grze istnieją trzy rodzaje wrogich statków. Dwa z nich, to przeciwnicy stosunkowo łatwi do zniszczenia, którzy zmierzają w lewo, aż opuszczą pole gry. Po przebyciu trasy złożonej ze 100

wrogów, gracz zmierzy się z jeszcze jednym przeciwnikiem, znacznie trudniejszym do pokonania. Po jego zniszczeniu lub po utracie trzech żyć gra kończy się i wyświetlone zostaje menu z wynikiem ostatniej rozgrywki oraz najwyższym uzyskanym wynikiem. Punkty przyznawane są za niszczenie wrogów oraz za zachowane życia i bomby. Rysunek 4 przedstawia schemat blokowy algorytmu gry, natomiast na rysunku 5 widoczny jest zrzut ekranu z rozgrywki.



Rys. 4. Schemat blokowy algorytmu gry Bullethell



Rys. 5. Zrzut ekranu z gry Bullethell

5. Podsumowanie

Konsola oraz napisane na nią gry działają zgodnie z zamierzeniami projektantów. Cała planowana funkcjonalność została osiągnięta. Przygotowane urządzenie jest drugim projektem Koła Pasjonatów Elektroniki stanowiącym rodzaj automatu do gier. Obie konstrukcje są wykorzystywane na imprezach promujących Wydział oraz Uczelnię. Jednocześnie budowa była doskonałą okazją do poszerzenia wiedzy z zakresu technik mikroprocesorowych. Podczas realizacji projektu okazało się, że płytka prototypowa Arduino Due, ze względu na ograniczoną pamięć, nie jest zdolna do obsługi dynamicznie zmieniającej się grafiki o rozdzielczości przekraczającej 320x240 pikseli. W związku z tym faktem, powstał pomysł przyszłego przedsięwzięcia mającego na celu konstrukcję kontrolera do gier. Mógłby on zachować stylizowaną formę podobną do obecnie zbudowanej konsoli, a jednocześnie komputer, który zapewniałby znacznie większe możliwości graficzne.

Bibliografia

1. Nielsen T. S., Barros G. A. B., Togelius J., Nelson M. J.: *Towards generating arcade game rules with VGDL*, 2015 IEEE Conference on Computational Intelligence and Games (CIG), Tainan, 2015, str. 185-192. doi: 10.1109/CIG.2015.7317941
2. Lee H., Jeong H., Han J.: *Arcade video game platform built upon multiple sensors*, 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, 2008, str. 111-113. doi: 10.1109/MFI.2008.4648118

3. Ślepecki J., Rydzewski M., Kisiel P., Poczekajło P.: *Prosta gra zręcznościowa typu "arcade" w oparciu o moduły sterujące z mikroprocesorami AVR*, Zeszyty Naukowe Wydz. Elektroniki i Informatyki Politechniki Koszalińskiej, tom 12 (2018), str. 5-15. ISSN 1897-7421
4. Online: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf
5. Online: GitHub -- stimmer/DueVGA: Arduino Due VGA library https://github.com/stimmer/DueVGA (28.09.2018)

Streszczenie

W niniejszym artykule przedstawiono projekt konsoli do gier wykonanej przez Koło Pasjonatów Elektroniki działające przy Katedrze Systemów Cyfrowego Przetwarzania Sygnałów na Wydziale Elektroniki i Informatyki Politechniki Koszalińskiej. Skonstruowany automat wykorzystuje do działania płytkę prototypową Arduino Due, która oprócz obsługi logiki gry przesyła również grafikę na zewnętrzny monitor za pomocą złącza VGA i specjalnie przygotowanego adaptera. Urządzenie jest wyposażone w kontroler w postaci joysticka i czterech przycisków. Całość jest stylizowana na klasyczny automat z salonów gier.

Abstract

In this article, the project of game console is presented. The console was created bu Club of Enthusiasts of Electronics in Faculty of Electonics and Computer Science, Koszalin University of Technology. The machine uses Arduino Due prototype board to run the game. The board is connected to monitor by VGA and special adapter. More over, there are joystick and four buttons used as game controller.

Keywords: game console, Arduino, ARM, SAM, VGA, pong, bullethell

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 63-78

Piotr Ratuszniak
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
ratusz@ie.tu.koszalin.pl

Radosław Łańcucki
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
radek.lancucki@gmail.com

Andrzej Stasiak
isandrzej@gmail.com

Równoległa realizacja przykładowego algorytmu genetycznego z wykorzystaniem akceleratorów GPU

Słowa kluczowe: algorytm genetyczny, programowanie równoległe, akceleracja obliczeń, akceleratory GPU, CUDA, problem komiwojażera

1. Wstęp

Nadrzędnym celem wytworzonego oprogramowania w ramach prowadzonych badań naukowych, było porównanie wydajności sekwencyjnej oraz równoległej realizacji algorytmu genetycznego z wykorzystaniem techniki programowania procesora graficznego GPU (ang. Graphics Processing Unit), tj. GPGPU (ang. General-Purpose computing on Graphics Processing Units). Wykorzystana została dostępna moc obliczeniowa karty graficznej komputera, a w szczególności macierz jednostek wykonawczych w przetwarzaniu współbieżnym poddanego badaniu algorytmu. Jako problem do rozwiązania przez zaimplementowany algorytm genetyczny, oznaczono problem komiwojażera (ang. TSP – Travelling Salesman Problem). To typowe zadanie optymalizacyjne, polegające na jednokrotnym odwiedzeniu każdego miasta znajdującego się na mapie aktualnie rozwiązywanego wariantu. Każda mapa problemu komiwojażera składa się z miast wraz z zakodowanymi ich współrzędnymi. W równoległej realizacji oprogramowania wybrano technologię Nvidia CUDA, która należy do opisanej powyżej techniki programowania GPGPU.

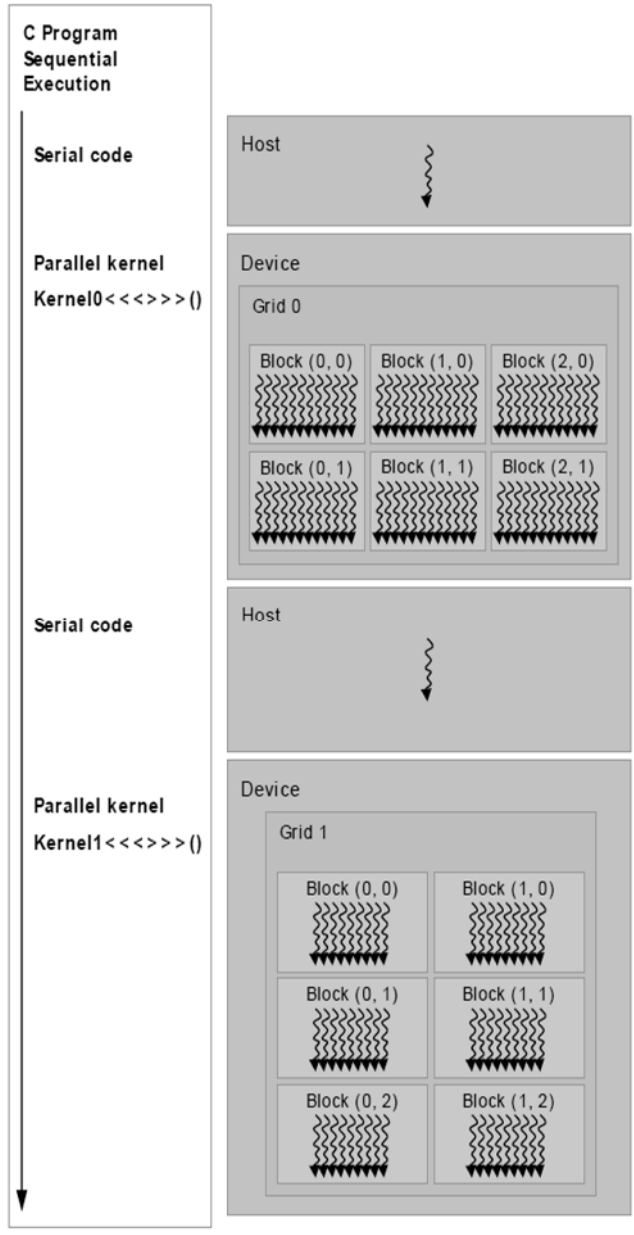
Wykonane oprogramowanie umożliwia rozwiązanie skonfigurowanego przez użytkownika algorytmu genetycznego w wersji sekwencyjnej, wykonywanej

całkowicie na procesorze komputera – CPU (ang. Central Processing Unit) oraz w wersji równoległej, gdzie obliczenia wykonywane są dodatkowo na karcie graficznej komputera. Jednym z założeń wykonanego oprogramowania była całkowita dowolność konfiguracji rozwiązywanego algorytmu genetycznego przez użytkownika, tj. dowolne ustawienie jego poszczególnych parametrów, czy też dowolna konfiguracja mapy rozwiązywanego przez niego problemu komiwojażera. W tym celu użyto biblioteki ManagedCUDA [1].

2. Technologia Nvidia CUDA

Technologia Nvidia CUDA zadebiutowała w lutym 2007 roku i jest rozwijana po dziś dzień. Obecnie wszystkie istniejące na rynku układy graficzne firmy Nvidia: GeForce, Tesla, Quadro; w pełni ją wspierają. Każdy program uruchamiany na karcie graficznej określany jest mianem tzw. jądra obliczeniowego (ang. kernel), które wykonywane jest zarówno przez CPU jak i GPU komputera. Technologia CUDA dostarcza jasno zdefiniowany model operacyjny programowania GPU: a) procesor alokuje oraz inicjalizuje pamięć operacyjną dla uruchomionego programu, b) alokacja pamięci układu GPU, c) skopiowanie wszystkich wymaganych danych z pamięci operacyjnej do pamięci GPU. Następną czynnością jest wywołanie przez procesor komputera jądra obliczeniowego, który operuje na wcześniej przygotowanych danych. Przetwarzanie danych na karcie graficznej odbywa się w sposób asynchroniczny w stosunku do CPU. Stanowi to bez wątpienia wielką zaletę opisywanej technologii, gdzie CPU nie uczestniczy w procesie obliczeniowym. Po wykonaniu zadania przez GPU, dane wynikowe z pamięci GPU są synchronizowane, tj. kopiowane do pamięci operacyjnej CPU. Ostatnim zadaniem CPU jest zwolnienie zarezerwowanych obszarów pamięci urządzenia GPU – dealokacja. Wszystkie jądra obliczeniowe mogą być wykonywane równocześnie przez wiele bloków (ang. blocks) CUDA. Bloki te pogrupowane są w tzw. siatki bloków (ang. grids). Za ich pojedyncze wykonanie odpowiedzialne są natomiast równoległe wątki (tzw. threads). Współczesne karty graficzne są w stanie wykonać nawet do 1024 wątków na blok jednocześnie.

Na rysunku 1 ukazano schemat przykładowego programu realizowanego sekwencyjnie wraz z równoległym wykonaniem po stronie GPU.



Rys. 1. Schemat przykładowego wykorzystania CUDA [2]

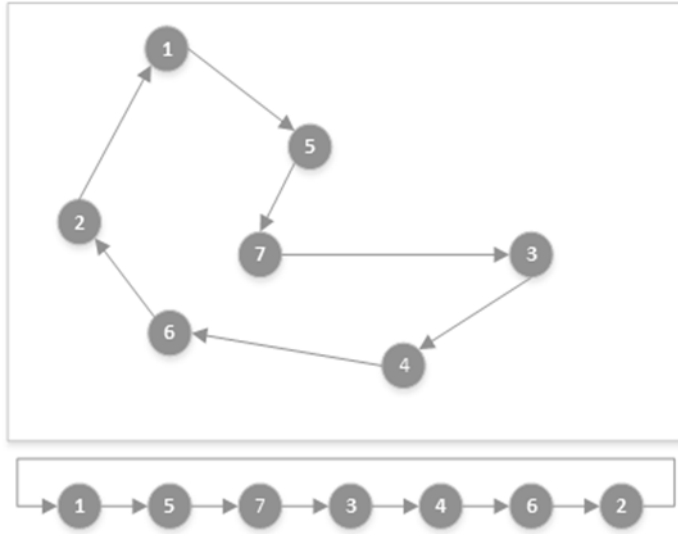
3. Algorytm genetyczny wykorzystany w rozwiązaniu problemu komiwojażera

Tak jak wspomniano we wstępie, utworzone oprogramowanie odpowiedzialne jest za rozwiązanie problemu komiwojażera. Problem ten jest zaliczany do grupy problemów NP-trudnych i jest on typowym problemem optymalizacyjnym. W związku z jego dużą złożonością $(n-1)!$, gdzie n to liczba wierzchołków miast na mapie – znalezienie optymalnego rozwiązania poprzez sprawdzenie wszystkich rozwiązań w przypadku większej liczby miast jest praktycznie niemożliwe z uwagi na bardzo długi czas działania programu. Zastosowanie w tym przypadku opisywanego algorytmu genetycznego pozwoli na znalezienie w rozsądnym czasie rozwiązania suboptymalnego, które jest w większości przypadków akceptowalne z uwagi na czas znalezienia tego optymalnego. Warunkiem zatrzymania zaimplementowanego algorytmu genetycznego jest osiągnięcie wprowadzonej przez użytkownika liczby generacji, bądź jego bezpośrednie zatrzymanie przez użytkownika programu. Poprzez generację algorytmu można rozumieć jeden pełny cykl algorytmu, czyli wykonanie działań oraz obliczeń dla jego całej populacji (wszystkich osobników). Na rysunku 2 przedstawiono schemat blokowy zaimplementowanego algorytmu genetycznego.

Rys. 2. Schemat blokowy algorytmu genetycznego

3.1. Reprezentacja osobników

Ważnym elementem implementacji algorytmu genetycznego jest sposób reprezentacji osobników w populacji. W zaprogramowanej aplikacji zastosowano permutacyjne kodowanie osobników, gdzie w genach chromosomów zapisywana jest informacja o numerze wierzchołka miasta odpowiadającego jego numerowi na mapie problemu komiwojażera. Rysunek 3 wraz z zawartą przykładową mapą, przedstawia omówiony sposób kodowania.



Rys. 3. Przykład kodowania permutacyjnego

3.2. Funkcja oceny i operatory genetyczne algorytmu

Funkcja oceny osobników

Funkcja oceny osobnika jest to miara jakości rozwiązania reprezentowanego przez dowolnego osobnika w populacji. Premiuje ona rozwiązania najbardziej zbliżone do optymalnego rozwiązania problemu komiwojażera, a jej wartość opisana jest wzorem 1, gdzie długość trasy jest zakodowana w chromosomie osobnika poddawanego ocenie.

$$\frac{1}{długośćtrasy} \qquad [1]$$

Operatory genetyczne

W algorytmie genetycznym zaimplementowanym w aplikacji zastosowano następujące operatory genetyczne:

- Krzyżowanie: szansa wystąpienia operatora krzyżowania mieści się w przedziale od 10% do 100% i jest ona ustalana przez użytkownika. Im

większa jego wartość tym większe prawdopodobieństwo jego wystąpienia. W procesie krzyżowania bierze udział dwoje rodziców (dwa skrzyżowane ze sobą chromosomy), w wyniku czego powstają dwa chromosomy potomne. Krzyżowanie to polega na wybraniu pierwszego genu od drugiego rodzica, a następnie na wybraniu kolejnych genów następujących po poprzednio wybranych zarówno od pierwszego, jak i drugiego rodzica. Ponadto żaden z genów w chromosomie nie może się powtarzać. W zaimplementowanym algorytmie zastosowano mechanizm krzyżowania zachłannego, które różni się od standardowego tym, że podczas wybierania kolejnego genu następuje obliczenie długości ścieżki pomiędzy wierzchołkami zakodowanymi w tych genach. Zachłanność polega na tym, że wybrany zostaje ten gen, którego obliczona długość ścieżki jest krótsza. Zastosowanie tego mechanizmu pozwoli w niektórych przypadkach na zwiększenie zbieżności algorytmu.

- Mutacja: szansa wystąpienia operatora jest identyczna jak w przypadku krzyżowania i jest ona również ustalana przez użytkownika. Mutacja w algorytmie genetycznym polega na losowym wybraniu dwóch genów w chromosomie, a następnie zamiana ich miejscami.
- Metody selekcji:
  - elitarna; metoda polegająca na selekcji najlepszych (najlepiej dostosowanych) osobników w populacji, a następnie umieszczenie ich w nowej populacji,
  - rankingowa; polega na sporządzeniu rankingu osobników na podstawie funkcji oceny. Następnie na jego podstawie tworzone jest wirtualne koło ruletki, gdzie największy wycinek koła zajmuje osobnik z najwyższą pozycją w rankingu. Po całej procedurze następuje losowanie, a wylosowane osobniki trafiają do nowej populacji,
  - koła ruletki; utworzone w ramach tej metody koło ruletki symbolizuje sumę ocen przystosowania wszystkich osobników, a poszczególne wycinki koła symbolizują wartość przystosowania danego osobnika w stosunku do całej populacji. Pozostałe czynności wykonywane w ramach tej metody są takie same jak w opisanej wcześniej metodzie rankingowej.

W celu zachowania możliwości pełnej konfiguracji przez użytkownika algorytmu oraz rozwiązywanego przez niego problemu, dokonano zrównoleglenia funkcji celu algorytmu genetycznego. Żaden z wymienionych parametrów nie jest zadeklarowany w oprogramowaniu na stałe, a jego wartość miała być ustalana tak jak wspomniano przez użytkownika aplikacji. Wybór na równoległą realizację funkcji celu padł z uwagi na niewystępowanie w niej czynników losowych oraz stałą liczbę elementów – osobników poddawanych zrównolegleniu. Poprzez czynniki losowe można rozumieć szansę wystąpienia poszczególnych operatorów

genetycznych, co wpływa później bezpośrednio na liczbę osobników biorących udział w obliczeniach. Brak występowania wspomnianej losowości w funkcji celu czyni ją jednocześnie najbardziej podatną na proces zrównoleglenia obliczeń ze wszystkich występujących w algorytmie genetycznym.

3.3. Zrównoleglenie algorytmu

Funkcja celu w algorytmie genetycznym podczas rozwiązywania problemu komiwojażera polega na obliczeniu łącznej długości ścieżki pokonywanej przez każdego z osobników w populacji z osobna. Długość pojedynczego odcinka pokonywanego w ramach zakodowanej w chromosomie osobnika trasy obliczane jest zgodnie ze wzorem 2

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad [2]$$

gdzie: x_2, y_2, x_1, y_1; współrzędne kolejno aktualnie przeliczanego wierzchołka miasta oraz poprzedzającego go wierzchołka miasta.

Powyższe obliczenia wykonywane są dla każdej pary wierzchołków zakodowanych w chromosomie osobnika w populacji, łącznie z połączeniem ostatniego wierzchołka z pierwszym. Po wyliczeniu długości trasy zakodowanej w chromosomie, poddawana ona jest następnie opisanej już ocenie funkcji przystosowania.

Liczba osobników biorących udział w równoległych obliczeniach określona jest wzorem 3

$$liczbapopulacji + K + M \qquad [3]$$

gdzie:

K – liczba osobników powstałych w wyniku krzyżowania,

M – liczba osobników powstałych w wyniku mutacji. Liczba ta jest zmienna dla każdej pojedynczej generacji algorytmu.

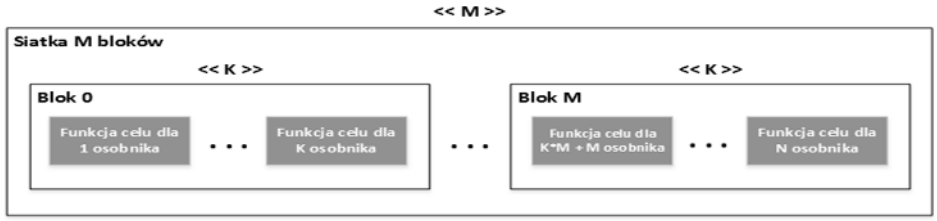
Jądro obliczeniowe programu wykonywanego na karcie graficznej GPU, odpowiedzialne za obliczenie opisywanej funkcji celu przyjmuje natomiast dwa parametry. Pierwszym z nich jest rozmiar bloku, którego wartość określona jest wzorem 4

$$\frac{liczbawątkównablokkartygraficznej}{8} \qquad [4]$$

natomiast drugim rozmiar siatki bloków, której wartość wyliczana jest ze wzoru 5

$$\frac{liczbaosobników+rozmiarbloku-1}{rozmiarbloku} \text{ w} \qquad [5]$$

Powyższe parametry zostały w oprogramowaniu dobrane metodą empiryczną. Na rysunku 4 zaprezentowano schemat przedstawiający równoległe obliczenia wykonywane dla N osobników. Literą M oznaczono rozmiar siatki bloków, natomiast literą K rozmiar pojedynczego bloku.

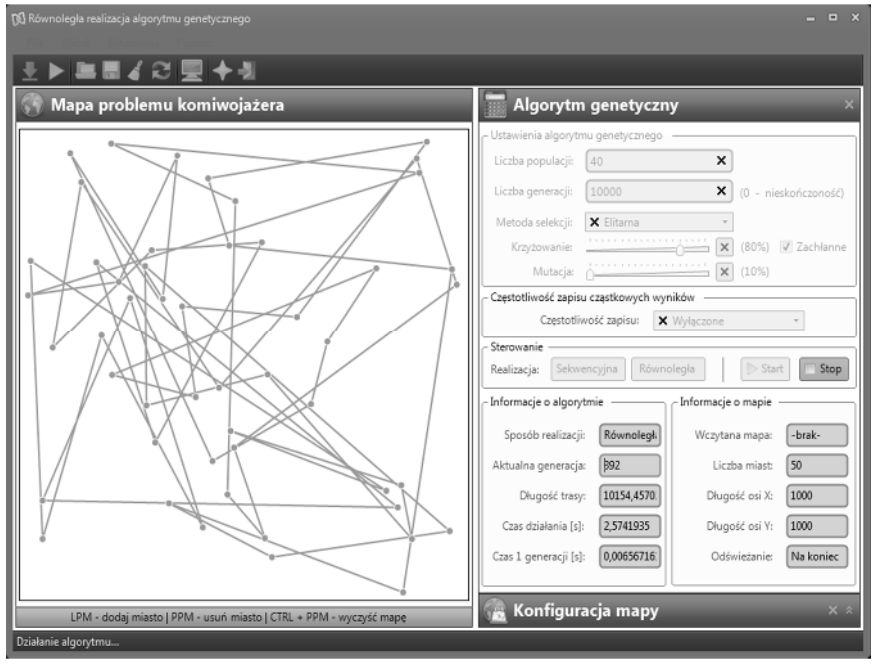


Rys. 4. Równoległa funkcja celu dla N osobników algorytmu genetycznego

Jak można łatwo zauważyć, obliczenia w ramach równoległej funkcji celu wykonywane są asynchronicznie w stosunku do siebie, co oznacza, że aby obliczyć funkcję celu dla 3 osobnika, nie jest konieczne wykonanie obliczeń dla dwóch poprzedzających go osobników. Za obliczenia odpowiedzialne są bezpośrednio rdzenie CUDA karty graficznej.

4. Aplikacja

Opisaną w artykule aplikację wykonano w celu porównania wydajności sekwencyjnej i równoległej wersji algorytmu genetycznego. Porównaniu poddany został czas realizacji algorytmu wykonanego w całości na procesorze CPU testowanego komputera (sekwencyjna wersja) oraz z zastosowaniem karty graficznej, która będzie odpowiedzialna za obliczenie funkcji celu (równoległa wersja) opisywanego algorytmu. Na rysunku 5 przedstawiono główne okno aplikacji.



Rys. 5. Główne okno aplikacji podczas działania algorytmu

W aplikacji można skonfigurować w pełni algorytm genetyczny poprzez ustawienie odpowiednio liczby populacji, liczby generacji algorytmu, metodę selekcji, szansę na wystąpienie operatora krzyżowania, czy też mutacji. Ponadto można w pełni skonfigurować mapę problemu komiwojażera, która ma zostać rozwiązana przez algorytm genetyczny. Można tego dokonać poprzez ustawienie: liczby miast na mapie, bądź też zmianę długości osi X oraz Y mapy. Użytkownik decyduje jaka realizacja algorytmu genetycznego ma zostać uruchomiona przez aplikację – sekwencyjna, czy też równoległa. Podczas działania algorytmu genetycznego w miejscu panelu konfiguracji mapy wyświetlany jest w aplikacji panel informacyjny o postępach jego pracy oraz drugi panel pokazujący informacje o mapie uruchomionego aktualnie problemu komiwojażera.

Aplikację wykonano w obiektowym języku programowania C#. Podczas implementacji opisywanej aplikacji wykorzystano między innymi następujące narzędzia i technologie: Microsoft Visual Studio 2012 Professional, pakiet CUDA Toolkit 6.0, bibliotekę programową ManagedCUDA (implementacje CUDA na platformie .NET), isku .NET, biblioteki: OpenHardwareMonitorLib (odczytywanie parametrów komputera), ComponentFactory.Krypton.Toolkit (wygląd aplikacji), OxyPlot (rysowanie wykresów).

Utworzony program dostarcza szereg funkcjonalności, w tym między innymi: wyświetlenie specyfikacji sprzętu komputera, zapis uzyskanych rezultatów, konfiguracja parametrów algorytmu genetycznego, określenie częstotliwości zapisu wyników cząstkowych, wybór trybu pracy – sekwencyjny/równoległy, natychmiastowe zatrzymanie algorytmu genetycznego, skonfigurowanie mapy problemu komiwojażera, wyświetlanie informacji o aktualnie uruchomionym algorytmie genetycznym oraz informacji o mapie problemu komiwojażera.

```
1  string resource = "ParallelGeneticAlgorithm.Kernels.kernel.ptx";
2  Streamstream =
   Assembly.GetExecutingAssembly().GetManifestResourceStream(resource);
3  CudaKernelrouteLengthCuda = cudaContext.LoadKernelPTX(stream,
   "RouteLength");
4
5  staticFunc<float[], float[], int[], int, float[]>cudaRouteLength =
   (host_mapx, host_mapy, host_route, N) =>
6  {
7  CudaDeviceVariable<float>device_mapx = host_mapx;
8  CudaDeviceVariable<float>device_mapy = host_mapy;
9  CudaDeviceVariable<int>device_route = host_route;
10 CudaDeviceVariable<float>device_length = newCudaDeviceVariable<float>(N);
11 float[] host_length;
12 routeLengthCuda.Run(device_mapx.DevicePointer, device_mapy.DevicePointer,
   device_route.DevicePointer, N, device_length.DevicePointer);
13 host_length = device_length;
14 device_mapx.Dispose();
15 device_mapy.Dispose();
16 device_route.Dispose();
17 device_length.Dispose();
18 returnhost_length;
19 };
20 float[] result = cudaRouteLength(MapX, MapY, Genes, Chromosomes.Count);
```

Rys. 6. Wywołanie jądra obliczeniowego z poziomu języka C#

Z uwagi na główne założenia wykonanego oprogramowania, wykorzystano zewnętrzną bibliotekę obsługującą technologię Nvidia CUDA w środowisku .NET., tj. ManagedCUDA. Biblioteka ta korzysta z plików PTX, które są skompilowanym przez *vncc*kodem maszynowym programu (kernel-a) napisanego w języku CUDA

C. Program ten obsługiwany jest przez funkcje biblioteki ManagedCUDA, która komunikując się z GPU przez jego sterownik graficzny, przekazuje kernel do GPU oraz uruchamia proces wykonawczy programu na rdzeniach CUDA.

Rysunek 6 przedstawia przykład obsługi jądra obliczeniowego (kernel) przez bibliotekę ManagedCUDA. Wspomniane jądro obliczeniowe odpowiedzialne jest za obliczenie długości ścieżki pokonywanej przez każdego kolejnego osobnika w populacji i uruchamiane ono jest w każdej generacji algorytmu genetycznego. Poprzez generację można rozumieć jeden pełny cykl pracy algorytmu genetycznego. Opisywane jądro obliczeniowe uruchamiane jest w każdej z generacji z innymi parametrami – rozmiarem bloku oraz rozmiarem bloku siatki z uwagi na zmienną liczbę elementów (osobników) biorących udział w równoległych obliczeniach.

Na rysunku 6 zaprezentowano przykładową programową obsługę procesów architektury Nvidia CUDA z poziomu języka programowania C#. Standardowy zestaw procesów niezbędny podczas uruchomienia jądra obliczeniowego oraz jego kontrola przy użyciu biblioteki ManagedCUDA, wygląda następująco:

1. Alokacja oraz inicjalizacja pamięci operacyjnej; (Rys. 6 wiersz 20)
2. Alokacja pamięci układu GPU oraz skopiowanie danych z pamięci operacyjnej do pamięci urządzenia; (Rys. 6 wiersze 7, 8, 9, 10 oraz 11)
3. Wywołanie jądra obliczeniowego; (Rys. 6 wiersz 12)
4. Skopiowanie rezultatu wykonania jądra z pamięci GPU do pamięci operacyjnej; (Rys. 6 wiersz 13)
5. Zwolnienie zarezerwowanej pamięci układu GPU; (Rys. 6 wiersze 14, 15, 16 oraz 17).

5. Wyniki

Badanie czasu wykonania zaimplementowanego algorytmu genetycznego realizowano na różnych wybranych platformach sprzętowych. W tabeli 1 przedstawiono specyfikację sprzętową komputerów, na których wykonano badania pomiarowe.

Tabela 1. Specyfikacja sprzętowa platform testowych

Podzespół	Nazwa parametru	Wartość parametru		
		I platforma	**II platforma**	**III platforma**
CPU	Nazwa:	Intel Core i5-4670K	Intel Core 2 Duo P7350	Intel Xeon E5506
	Liczba rdzeni:	4	2	4
	Taktowanie:	3800MHz	1995MHz	2130MHz
GPU	Nazwa:	GeForce GTX 760	GeForce GT 130M	Quadro K5000
	Wersja CUDA:	3.0	1.1	3.0
	Rdzenie CUDA:	1152 (6x192)	32 (4x8)	1536 (8x192)
	Wątki na blok:	1024	512	1024
	Taktowanie rdzenia:	1020MHz	1500MHz	705MHz
	Taktowanie pamięci:	1500MHz	800MHz	2700MHz
	Wielkość pamięci:	2048MB	512MB	4096MB
Pamięć RAM	Wielkość pamięci:	8192MB	4096MB	4096MB

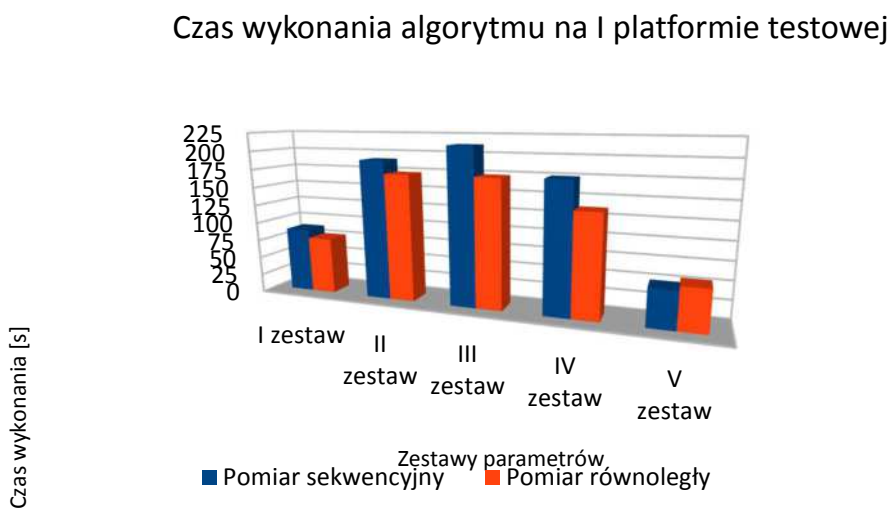
W tabeli 2 przedstawiono zestawy parametrów algorytmu genetycznego, zgodnie z którymi przeprowadzono badania pomiarowe.

Tabela 2. Zestawy parametrów testowych algorytmu genetycznego

Parametr	Wartość parametru				
	I zestaw	**II zestaw**	**III zestaw**	**IV zestaw**	**V zestaw**
Liczba populacji	2500	4000	16500	12500	100
Liczba generacji	2000	2000	1000	1000	10000
Metoda selekcji	rankingowa	koła ruletki	elitarna	elitarna	elitarna
Krzyżowanie	40% (zachłanne)	50% (zachłanne)	80% (zachłanne)	50% (zachłanne)	80%
Mutacja	10%	15%	10%	10%	10%
Liczba miast	60	50	40	55	100
Wczytana mapa	mapa_zestaw_1	mapa_zestaw_2	mapa_zestaw_3	mapa_zestaw_4	mapa_zestaw_5

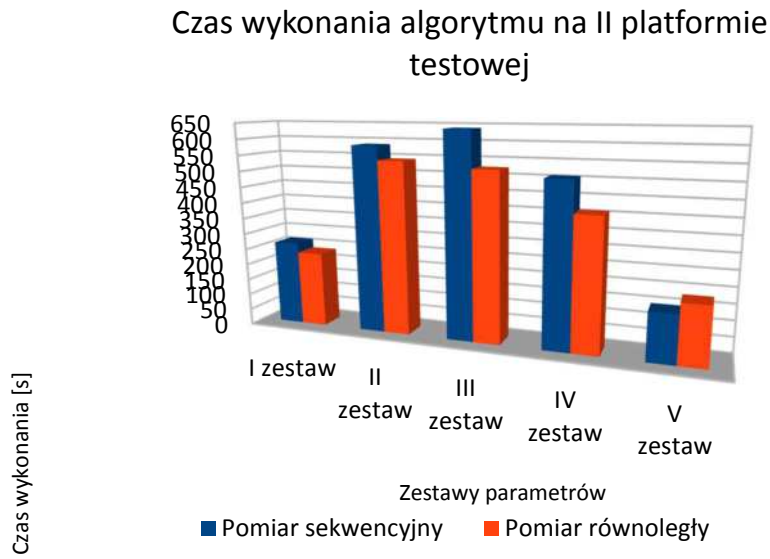
Dobór parametrów wpływa w sposób dominujący na wyniki obliczeń algorytmu genetycznego. Duża liczba miast oraz populacji może spowodować przetwarzanie jednej generacji nawet na poziomie kilkunastu sekund, co spowoduje oczywiście bardzo długi czas uzyskania przez algorytm zbieżności. Losowość działania poszczególnych operatorów genetycznych algorytmu genetycznego również ma wpływ na działanie samego algorytmu oraz na czasy poszczególnych uruchomień nawet dla tej samej konfiguracji sprzętowej.

Na wykresach rysunku 7 kolorem niebieskim oznaczono czas pomiaru sekwencyjnego, natomiast pomarańczowym pomiar równoległej realizacji algorytmu. Należy zwrócić uwagę na różnicę w czasie wykonania pomiędzy obiema realizacjami algorytmu genetycznego dla poszczególnych zestawów parametrów testowych. Rezultaty czasu działania przedstawione na tym rysunku uzyskano na I platformie testowej.



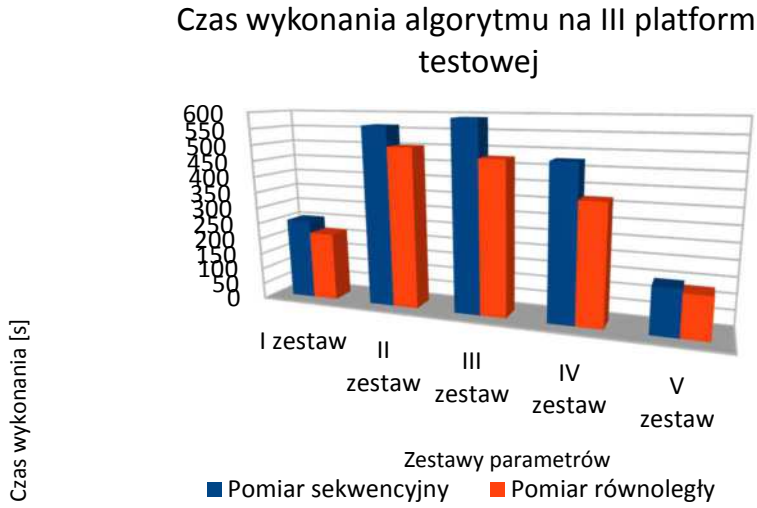
Rys. 7. Wykres czasu wykonania algorytmu genetycznego na I platformie testowej

Na rysunku 8 przedstawiono wykres czasu wykonania sekwencyjnego oraz równoległego na II platformie testowej.



Rys. 8. Wykres czasu wykonania algorytmu genetycznego na II platformie testowej

Na rysunku 9 przedstawiono rezultaty uzyskane na III platformie testowej.



Rys. 9. Wykres czasu wykonania algorytmu genetycznego na III platformie testowej

Czasy uzyskiwane przez równoległą realizację algorytmu genetycznego w większości przypadków były lepsze (tj. krótsze) w porównaniu

z jej sekwencyjnym odpowiednikiem. Opisywana równoległa realizacja okazywała się gorsza jedynie w przypadkach V zestawu parametrów, zarówno na I jak i II platformie testowej. Na wynik pomiaru w przypadku V zestawu parametrów miał wpływ bez wątpienia niski dobór liczby populacji algorytmu genetycznego, a co za tym idzie mała liczba elementów podlegających zrównolegleniu w funkcji celu. Na III platformie testowej czas uzyskany przez równoległą realizację był w każdym badanym przypadku lepszy, a miał na to bez wątpienia wpływ bardzo dobry akcelerator graficzny – Quadro K5000, charakteryzujący się największą liczbą rdzeni obliczeniowych.

6. Podsumowanie

Zgodnie z otrzymanymi rezultatami, zastosowanie technologii Nvidia CUDA pozwoliło przyspieszyć obliczenia algorytmu genetycznego dla przykładowo wybranego problemu. Chęć zachowania możliwości pełnej elastyczności w konfiguracji algorytmu genetycznego oraz mapy problemu komiwojażera jaka ma być przez niego rozwiązana bez ingerencji w kod programu, skutecznie ograniczyła pole manewru odnośnie integracji opisywanej aplikacji ze wspomnianą technologią CUDA. Czynnikami mającymi na to wpływ była przede wszystkim losowość towarzysząca algorytmowi genetycznemu, a co za tym idzie zmienne rozmiary tablic na których operuje się podczas jego działania, czy też zmienna liczba elementów poddawanych przetwarzaniu. Powyższa argumentacja może świadczyć o tym, że technologia GPGPU, do której zalicza się technologia CUDA, posiada swoje ograniczenia i wydaje się być bardziej efektywna dla wybranej grupy problemów. Pomimo świadomości istnienia wymienionych powyżej ograniczeń, na podstawie zrealizowanych badań można stwierdzić, że technologia Nvidia CUDA spełnia pokładane w niej oczekiwania i realizuje swoje zadanie przyspieszając w większości przypadków działanie aplikacji. Przyspieszenie to jest bardziej widoczne dla większej liczby osobników w populacji algorytmu genetycznego.

Zaprogramowana aplikacja dzięki zastosowaniu modułowej budowy oraz interfejsów może być w przyszłości z powodzeniem rozwijana. Głównymi kierunkami jej rozwoju, lecz z pewnością nie jedynymi są:

- dalsza optymalizacja obliczeń wykonywanych po stronie karty graficznej;
- dodanie kolejnych jąder obliczeniowych wykonujących obliczenia z innych jak obecnie obszarów algorytmu genetycznego;
- implementacja kolejnych algorytmów genetycznych.

Wymienione powyżej kierunki rozwoju aplikacji są oczywiście wolnymi sugestiami. Mają one za zadanie głównie zobrazowanie jak rozbudowany i czasochłonny w implementacji jest poruszany problem algorytmów genetycznych

oraz samej technologii GPGPU do której zalicza się Nvidia CUDA, a jej efektywność zależy od zaimplementowanego algorytmu oraz również od doświadczenia projektanta/programisty aplikacji.

Bibliografia

1. http://managedcuda.codeplex.com/documentation,
2. http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, CUDA C Programming Guide,
3. Sanders J.; Kandrot E., CUDA by Example, Addison-Wesley, 2011, ISBN 0-13-138768-5,
4. Karaszewski M., http://problem-komiwojazera.cba.pl/,
5. SinhHoaNguyen, Algorytmy ewolucyjne, http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm,
6. Lubiński T., Algorytmy genetyczne, http://www.algorytm.org/kurs-algorytmiki/algorytmy-genetyczne.html,
7. Lapthorn B., A Simple C# Genetic Algorithm, http://www.codeproject.com/Articles/3172/A-Simple-C-Genetic-Algorithm,
8. http://en.wikipedia.org/wiki/CUDA

Streszczenie

W artykule zaprezentowano praktyczną implementację aplikacji rozwiązującej przykładowy algorytm genetyczny z wykorzystaniem akceleratorów GPU. W tym przypadku zdecydowano się na rozwiązanie za pomocą algorytmu genetycznego typowego problemu optymalizacyjnego, jakim jest problem komiwojażera. Dodatkowo w celu wykorzystania mocy karty graficznej w tworzonej aplikacji wykorzystano technologię programowania na karcie graficznej – technologię Nvidia CUDA.

Abstract

The paper presents a practical implementation of a local desktop application that solves exemplary genetic algorithm with the use of GPU accelerators. In this case decided with the use of genetic algorithm to solve typical optimization problem which is travelling salesman problem. Additionally used Nvidia CUDA programming technology in order to use power of GPU in created application.

Keywords: genetic algorithm, parallel programming, computing acceleration, GPU, CUDA, travelling salesman problem

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej
ISSN:1897-7421
Nr 13, Listopad 2018
Str: 79-87

Svetlana Zhukovetskaya
Computer Engineering Department
Odessa National Academy of Food Technologies
E-mail: szhukovetckaya@i.ua

Air flowing spatial modeling and simulation with SOLIDWORKS CAD

Keywords: SOLIDWORKS CAD, spatial modeling, air flowing simulation

Introduction

Air conditioning system provides a large number of air treatment processes, which could be used to meet various air requirements in confined spaces. Computer engineering comes up with the construction of a spatial model, to be used for simulation and analysis that leads to preliminary identification of potential problems in air-conditioning system.

Object models are simpler systems, with a clear structure and specified relationships between the component parts. That allows to analyse the properties of real objects and their behavior in various situations. Due to the rapid development of software and hardware solutions in computing machinery and the ability to conduct numerical spatial modeling of airflow, it became possible to use computer modeling as the initial method for studying the distribution of the airflow in the laboratory of the ONAFT.

The aim of the work is airflow movement simulation and spatial modeling in the laboratory of ONAFT based on the multi-zone air-conditioning system.

The Object of study. The objects of study are multi-zone air conditioning systems. Multizone system is one of possible solutions to the problem of air conditioning several spaces simultaneously. Multizone system consists of an outdoor unit and a variety of indoor units. Every indoor unit can be controlled centralized or controlled locally (Fig. 1).

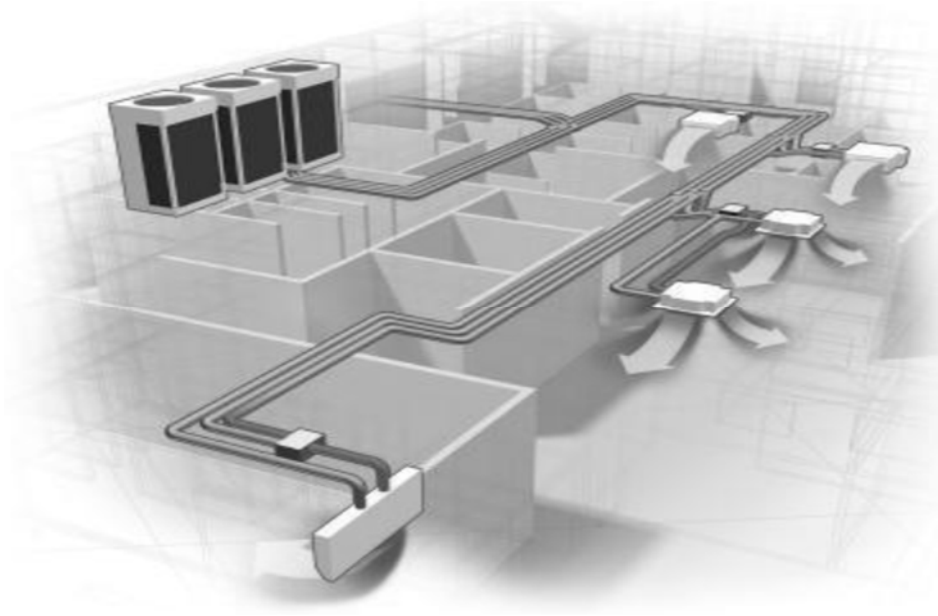


Fig. 1. Schematic representation of a multi-zone air conditioning system

Research methods. The most popular research methods are computer simulations with automatic design systems. SolidWorks had been chosen as a software solution for creating air movement model in the air conditioning laboratory.

SolidWorks is a system that allows you to create size-controlled solid models and can serve as the basis for solving many different engineering problems.

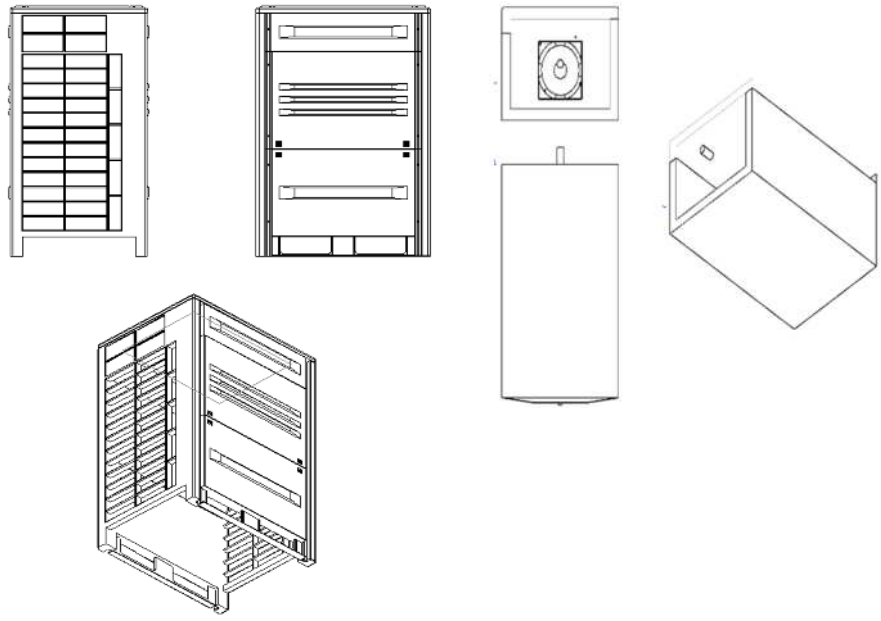
The extensive capabilities of the main module are combined with a large number of special-purpose applications, that makes SolidWorks a powerful software solution that can be flexibly configured to solve almost any design and production challenge [1].

1. Spatial model preparations

The algorithm for conducting a study includes the following steps:

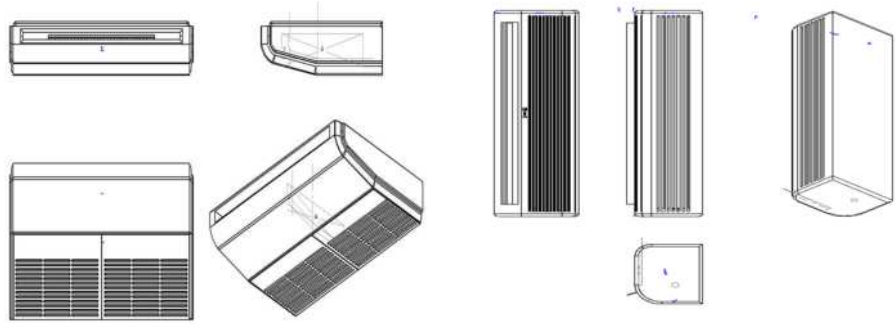
1. The development of a spatial model of the object of study.
2. Determination of the area of calculation.
3. Setting the boundary conditions.
4. The calculation.
5. Visualization of the calculation results.

Floor plan of the laboratory and the drawings of the air conditioning units were used as the initial data for spatial modeling. The result is a spatial solid-state parametric assembly consisting of the parts shown in Figures 2-5.

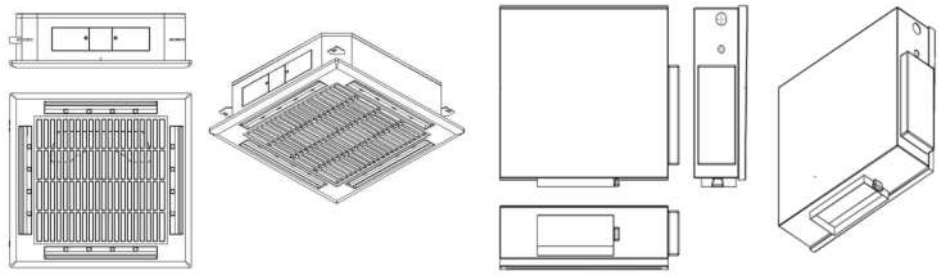


a) b)

Fig. 2. Assembling the part of the outdoor unit: a) case, b) heat exchanger



a) b)



c) d)

Fig. 3. Indoor unit parts: a) subceiling, b) wall mounted, c) cassette, d) channel

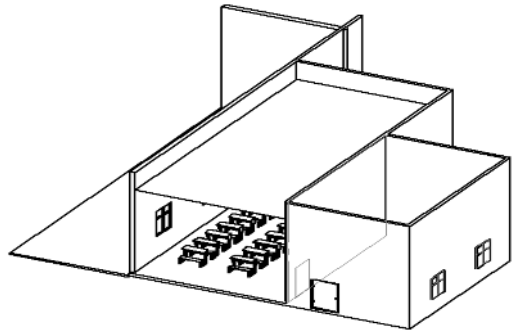


Fig. 4. Assembling the PartsRoom

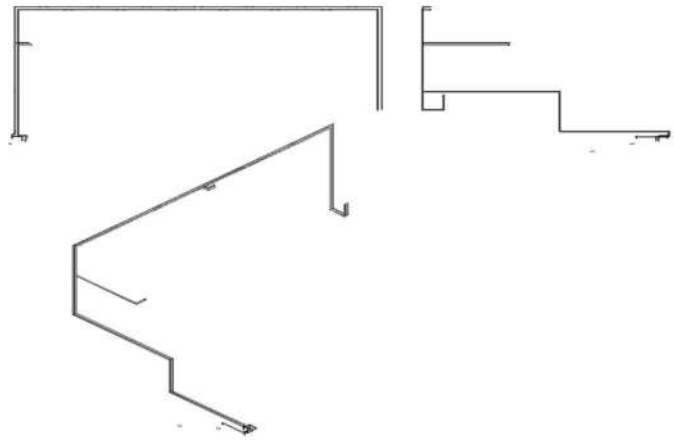
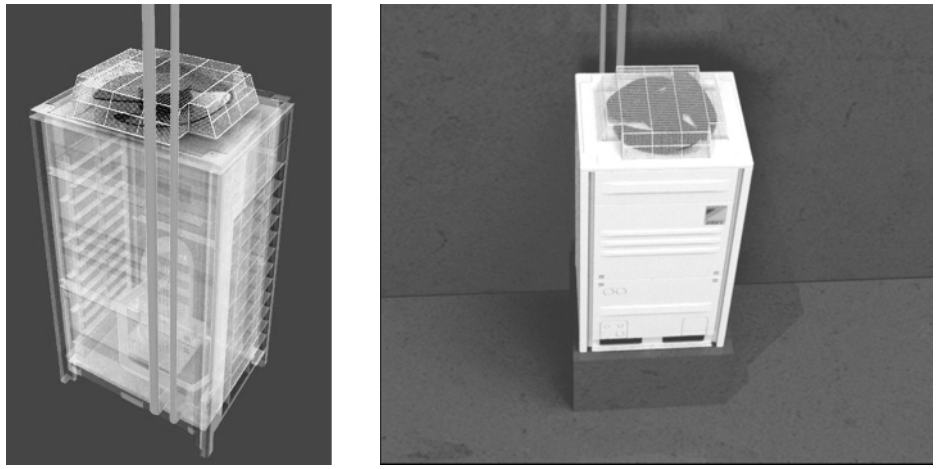


Fig. 5. The pipeline parts

The lighting and texturing of the model are specified for a visual representation. The visualization results are presented in Figures 6-7.



a) b)

Fig. 6. Visualization of the outdoor unit a) transparent, b) non-transparent

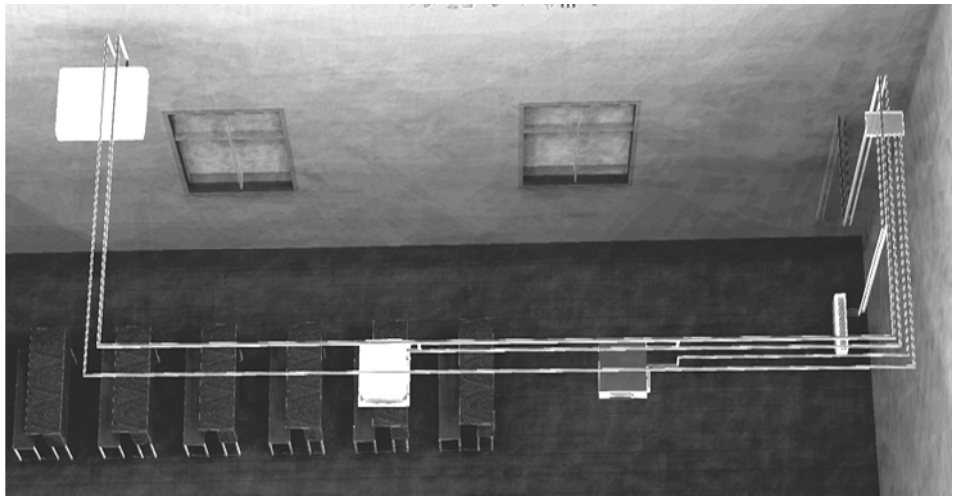


Fig. 7. The finalrender

2. Airflow Simulation

To analyze the movement of air masses, an additional SolidWorks engineering analysis module, Flow Simulation, was used. It allows you to simulate the flow of

gases, control the grid, perform complex thermal calculations, as well as evaluation of rotating objects, create gas-dynamic and thermal models of technical devices, etc.

In the Solidworks Flow Simulation, the motion and heat exchange of the fluid is calculated based on the Navier – Stokes equations. They simulate turbulent, laminar and transient flows [1]. The following initial conditions were defined for the aerodynamic calculations: air was selected as the fluid; ambient environment temperature - 20 °C; the remaining parameters are taken by default.

All calculations of flow and heat transfer in a solid body are performed within the computational domain. It is assumed that the system operates in recirculation mode, i.e. There is no external air intake into the room (Fig. 8).

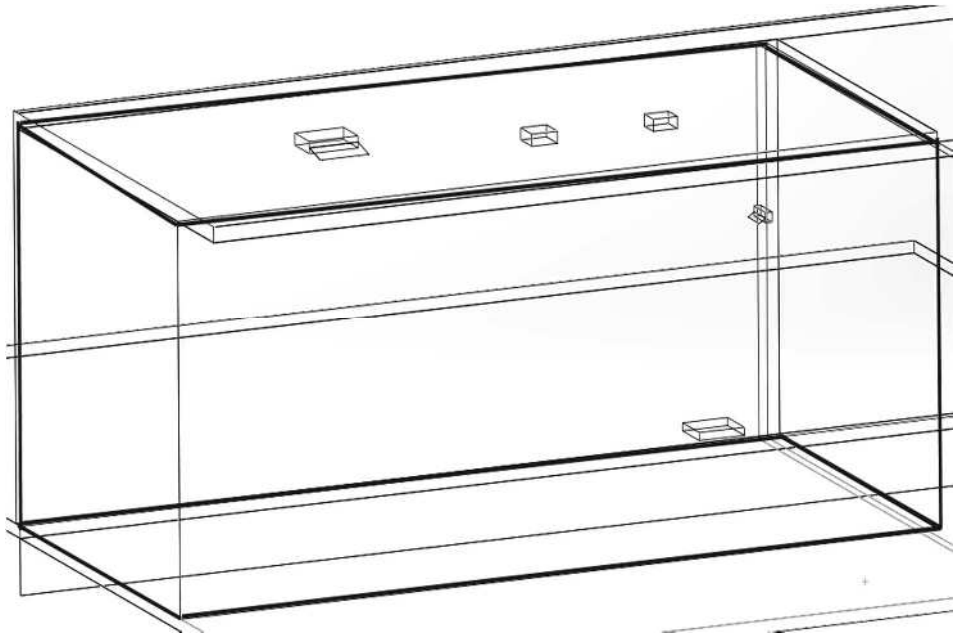


Fig. 8. The processing area

Apparently new objects had been added apart from simplifications. It is used to simulate air conditioners. It is represented by a simple geometry - a body without cavities, having two flat faces: one will imitate the input, the other - the output. It is irrelevant what is between these, therefore the profile of the bodies schematically reproduces the contours of air conditioners.

Creation of boundary conditions is one of the most important elements of the work. It is worth noting that it is necessary to set the boundary conditions for each element in the project, otherwise, the correct calculations are impossible. Following default thermodynamic parameters are used: pressure set equal to 101325 Pa,

temperature set equal to 293.2 K. Air is chosen as the fluid, humidity is not taken into account. The task is solved as an internal non-stationary.

The boundary condition of the SolidWorks "Fan" is considered as an ideal device that creates volumetric (or mass) consumption (flow rate can be set to spin) depending on the difference in static pressures at the inlet and outlet of this device. The volumetric consumption rate of the fluid through the SolidWorks "Internal Fan" is determined by the difference between the static pressures of the fluid on its inlet and outlet surfaces, that are calculated during modeling and averaged over these surfaces. Surfaces of the model on which "Fan" is set are selected. To set up "Internal Fan", "Inlet" is set as the surface through which the fluid exits the fan, and the "Outlet" is set as the surface through which the fluid enters the fan [2].

The process of movement of the airflow was modeled after determining the conditions of the calculation.

3. Interpretation of the calculation results

One of the advantages of analysis tools integrated into CAD systems is the visualization of the calculation results directly in the graphics window of the modeling system. Flow Simulation has a complete visualization toolkit, that has already become the standard for that kind of applications.

It includes the shear and surface diagrams, the distribution of the results by the fluid flow (in our case it is air), etc. [3]. Spatial trajectories of air masses in an enclosed spaces were built as a result of the calculation (Fig. 9).

Fig. 9. Spatial trajectories of air masses

Current lines contain substantial information. Apparently almost any information can be extracted. The existing situation assessment and improvements proposals can be made based on that information. Main constraint is the large computational complexity of the problem.

In addition to the research results, demonstration results were obtained that represent the animation of the movement of air masses. Spatial geometry with textures and lighting has been added to the results described above. The frame of the animation is shown in Figure 10.

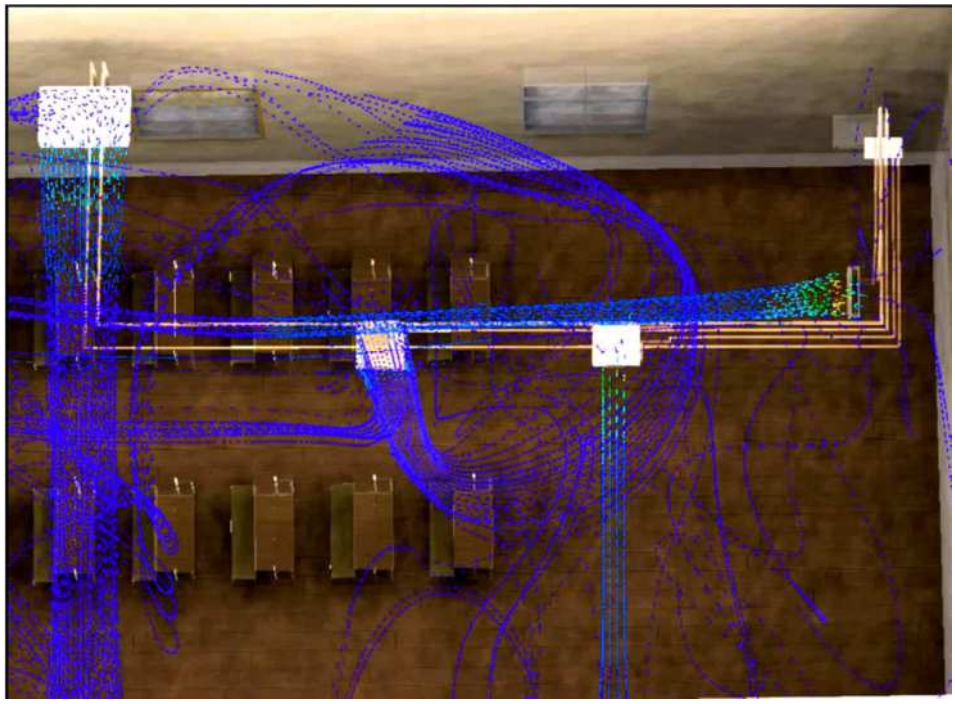


Fig. 10. An air mass animation frame

Conclusion

The paper contains overview of the results of solid modeling and assembly of the elements that composed the multi-zone air conditioning system. Generated a visual representation of results. Movement of air masses in enclosed space was studied based on assembly implementation. The results are presented in graphics and animations.

Simulation of the movement of air masses is carried out to obtain the following indicators: the direction of the air flow and the vectors of its movement; "dead zones"; airflow speed. Based on these indicators, experts will make reasonable

recommendations for improving the microclimate of the space, optimization of the air masses movement, placement and power of ventilation equipment, etc.

Computer Flow Simulation is faster than making a prototype or model, equipped with sensors, run a test cycle and get data suitable for further work. The resulting model can be used to compile and visually evaluate options for the planning of classrooms, as well as a demonstration material on the electronic resources of the school.

Literature

1. A.A. Alyamovsky. SolidWorks 2007/2008.Computer modeling in engineering practice / A.A. Alyamovsky, A.A. Sobachkin, E.V. Odintsov, A. I. Kharitonovich. –SPb .: BHV–Petersburg. 2008. – 1040 pp., Ill.
2. A. A. Alyamovsky. SolidWorks Simulation.How to solve practical problems.– SPb .: BHV– Petersburg.2012. – 448.
3. Flow Simulation 2009 Tutorial. [Electronic resource].– Access mode https://learn.ztu.edu.ua/pluginfile.php/29485/mod_resource/content/1/solidworks_flow_flow_simulation_2009_tutorial.pdf

Abstract

The technologies spatial visualization and simulation of the operation of technological equipment have become particularly relevant due to the fact that they provide an opportunity to obtain and analyze information about the operation of equipment before its installation.

The article provides an example of solid-state spatial modeling of a multizone conditioning system working in an chosen area – research laboratory. Based on the model, studies of the movement of air masses in a closed room were carried out.

Streszczenie

Technologie przestrzennej wizualizacji i symulacji działania urządzeń technologicznych stały się szczególnie istotne ze względu na fakt, że dają one możliwość uzyskania i analizy informacji o działaniu sprzętu przed jego zainstalowaniem.

Artykuł stanowi przykład modelowania przestrzennego wielostrefowego systemu kondycjonowania w przykładowym pomieszczeniu – laboratorium badawczym. W oparciu o model przeprowadzono badania ruchu mas powietrza w zamkniętym pomieszczeniu.

Słowa kluczowe: SOLIDWORKS CAD, modelowanie przestrzenne, symulacja przepływu powietrza

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 89-103

Ewa Kaczmar,
Józef Matuszek,
Dorota Więcek
Wydział Budowy Maszyn i Informatyki
Akademia Techniczno-Humanistyczna w Bielsku-Białej
43-309 Bielsko-Biała, ul. Willowa 2
ekaczmar@ath.bielsko.pl,
jmatuszek@ath.bielsko.pl,
wiecekd@ath.bielsko.pl

Model kalkulacji kosztów własnych wyrobów w warunkach zróżnicowanej wielkości produkcji

Słowa kluczowe: koszty własne wyrobu, rachunek kosztów

1. Wstęp

Współcześnie mimo funkcjonowania wielu algorytmów kalkulacji kosztów produkcji wyrobów, wciąż istnieje potrzeba tworzenia nowych, coraz bardziej dokładniejszych modeli, które znajdą zastosowanie w elastycznych systemach produkcji, gdzie niejednokrotnie produkowane są różne wyroby, charakteryzujące się niejednorodną wielkością produkcji. Wiąże się to z koniecznością ciągłego szukania oszczędności przez przedsiębiorstwa w kosztach produkcji, zachowując przy tym jakość wyrobów pożądaną przez rynek. Zwykle przedsiębiorstwa stosują jeden algorytm kalkulacji dla wszystkich wyrobów, niezależnie od wielkości produkcji czy złożoności wyrobu, co nie zawsze jest dobrym rozwiązaniem, gdyż może się to wiązać z problemem niedoszacowania lub przeszacowania kosztów dla poszczególnych produktów.

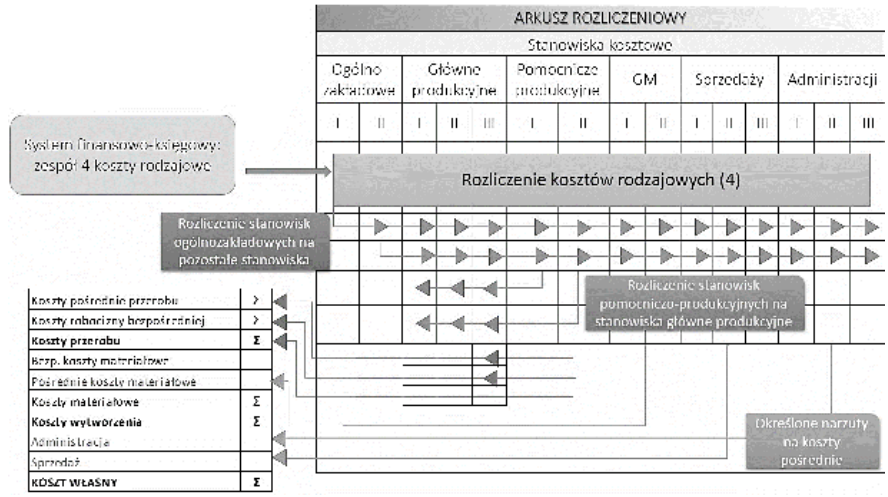
2. Określanie kosztów produkcji

2.1. Arkusz Rozliczeniowy

Jednym z powszechnie stosowanych narzędzi analizy kosztów jest Arkusz Rozliczeniowy (AR). Arkusz ma postać tablicy, w której kolumny stanowią stanowiska kosztowe, natomiast w wierszach wyszczególnione są koszty według rodzaju. Sporządzanie arkusza polega na ewidencjonowaniu poszczególnych kosztów rodzajowych i przyporządkowania ich do miejsc ich powstawania. Przez miejsca powstawania kosztów rozumieć można elementy działalności przedsiębiorstwa takie, jak na przykład komórki organizacyjne czy stanowiska pracy [3]. Schemat budowy Arkusza Rozliczeniowego przedstawia rys. 1.

Dzięki stworzeniu Arkusza Rozliczeniowego w dalszym etapie możliwe jest dokonanie rachunku kosztów według nośników kosztów. Za nośnik kosztów przyjąć można zlecenia, partie produkcyjne, pojedyncze wyroby czy grupy wyrobów. Podstawową rolę w rachunku kosztów na nośniki odgrywa kalkulacja, której zadaniem jest wskazanie wartości jednostkowego kosztu wytworzenia przedmiotu kalkulacji wraz ze wskazaniem jego elementów składowych [3].

Oprócz rozliczenia kosztów na miejsca powstawania, arkusz AR umożliwia także rozliczenie na stanowiska główne kosztów stanowisk pomocniczych oraz obliczenie stawek doliczeniowych (narzutów) kosztów pośrednich [5].



Rys. 1. Budowa Arkusza Rozliczeniowego [3]

Arkusz Rozliczeniowy może dotyczyć różnych okresów rozliczeniowych w zależności od potrzeb przedsiębiorstwa – miesiąca, kwartału, roku itp. Porównywanie kosztów i wartości narzutów z Arkuszów Rozliczeniowych dla

różnych okresów może dostarczyć danych o wysokości kosztów poniesionych przez stanowiska kosztowe na przestrzeni czasu, co w wyniku dalszej analizy może wykazać tendencję kształtowania się kosztów i ich przyczyny.

2.2. Metody kalkulacji kosztów własnych produkcji

Metody kalkulacji kosztów własnych produkcji można podzielić na trzy grupy biorąc pod uwagę kryterium budowy algorytmu obliczeniowego [3]:

- kalkulacje podziałowe,
- kalkulacje doliczeniowe,
- kalkulacje kosztów działań.

W kalkulacji podziałowej dzielona jest suma kosztów poniesionych w danym okresie przez wytworzoną w tym okresie ogólną liczbę jednostek produkcji. Dzięki temu możliwe jest ustalenie kosztu przeciętnego jednostki w danym okresie. Metoda ta znajduje zastosowanie głównie w przypadku produkcji wielkoseryjnej i masowej, w przypadku produkcji jednego wyrobu lub wyrobów podobnych [5].

Kalkulacja doliczeniowa sprowadza się do przyporządkowania kosztów bezpośrednich do nośnika kosztów na podstawie dokumentacji źródłowej, natomiast pośrednie koszty rozliczane są przy pomocy kluczy rozliczeniowych, a w dalszej kolejności na podstawie narzutów doliczane są do kosztów bezpośrednich. Kalkulacja ta ma zastosowanie głównie w produkcji seryjnej [5].

Rachunek kosztów działań *(Activity-BasedCosting)* stanowi alternatywę dla tradycyjnych metod rozliczania kosztów tj. kalkulacji podziałowej i doliczeniowej, gdyż według tej metody koszty pośrednie rozliczane są na wyroby według różnych podstaw rozliczenia, nieproporcjonalnych do wielkości produkcji. Kalkulacja kosztów działań koszty muszą zostać ujęte w przekroju działań, a potem w przekroju obiektów kosztowych. Metoda ta obejmuje 4 etapy:

1. etap – Identyfikacja występujących w przedsiębiorstwie istotnych działań
2. etap – Zdefiniowanie jednostki miary wielkości danego działania,
3. etap – Ustalenie kosztów wyszczególnionych działań,
4. etap – Rozliczenie pośrednich kosztów działań na produkty[5].

Wybór określonej metody zależy od rozmiaru i rodzaju produkcji, charakteru procesu wytwarzania, jego stopnia automatyzacji i mechanizacji [3].

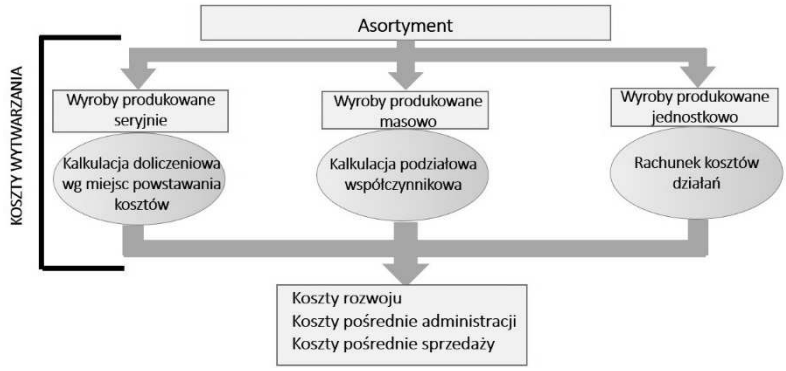
3. Model kalkulacji wyrobów

W niniejszym rozdziale przedstawiony zostanie model kalkulacji kosztów dla przedsiębiorstwa, w którym produkowane są różnorodne wyroby, a wielkość ich produkcji znacznie się różni między sobą. W przedsiębiorstwie produkowane są 3 rodzaje wyrobów:

- brzeszczoty wytwarzane masowo,
- piły i noże taśmowe wytwarzane seryjnie,
- sprzęt sportowy wytwarzany jednostkowo.

Wymienione wyroby różnią się między sobą nie tylko wielkością produkcji, ale także zróżnicowanym popytem w skali roku.

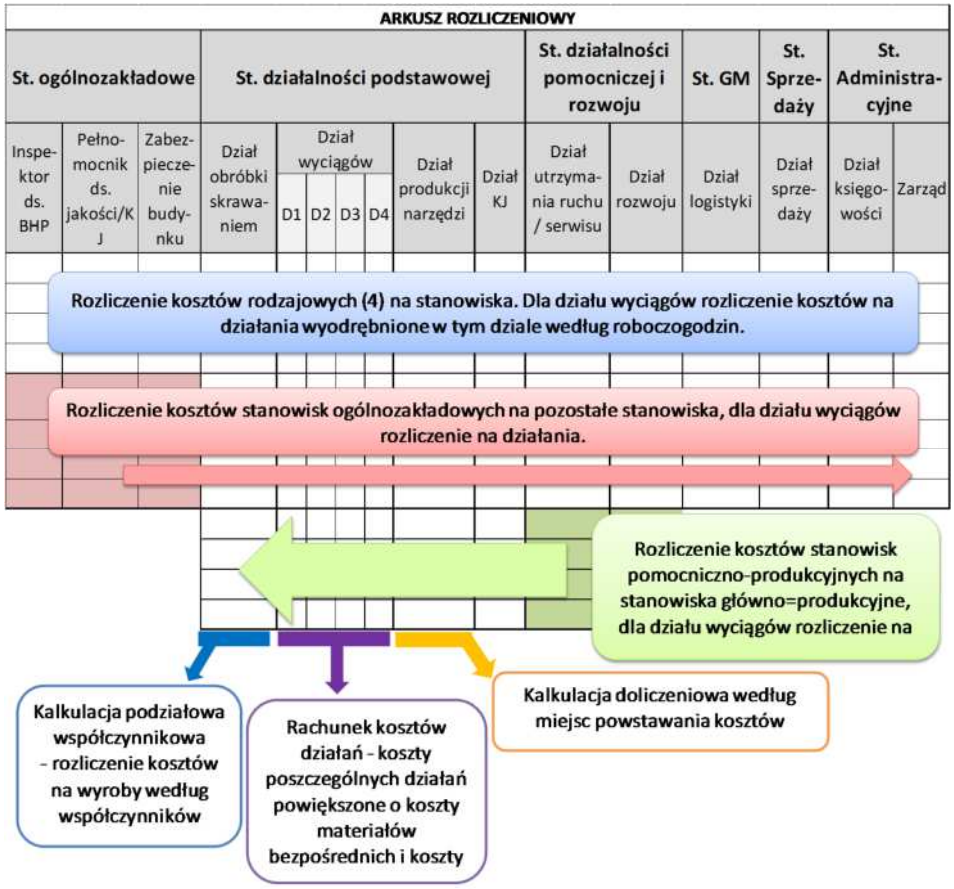
W przedsiębiorstwach tego typu, w których asortyment jest mocno zróżnicowany zarówno pod względem technologicznym, jak również wielkości produkcji, dobranie odpowiedniej metody kalkulacji kosztów własnych wyrobu może okazać się bardzo problematyczne. W związku z tym opracowano algorytm rachunku kosztów, który łączy w ramach jednego przedsiębiorstwa różne metody kalkulacji. Opracowany model zastosowania kalkulacji do różnych wielkości produkcji został przedstawiony na rysunku 2.



Rys. 2. Zastosowanie algorytmów kalkulacji kosztów dla różnych wielkości produkcji

Koszty dla poszczególnych wyrobów zostaną pozyskane ze zmodyfikowanego arkusza rozliczeniowego (rys. 3).

Arkusz stworzony został według ogólnego schematu budowy arkusza rozliczeniowego. Natomiast w związku z tym, że dla sprzętu sportowego produkowanego w dziale wyciągów, zgodnie z opracowanym modelem kalkulacji hybrydowej, zastosowany ma zostać rachunek kosztów według kosztów działań, na arkuszu rozliczeniowym w ramach tego działu koszty rozliczane będą nie na konkretne stanowiska, lecz na działania.



Rys. 3. Arkusz rozliczeniowy dla omawianego przedsiębiorstwa

3.1. Wyrób wytwarzany w warunkach produkcji masowej

W przedsiębiorstwie masowo wytwarzane są brzeszczoty. Produkcja obejmuje wytwarzanie kilku rodzajów brzeszczotów. Wyroby są technologicznie podobne, składają się z małej liczby elementów, różnicą między wyrobami jest ich wielkość, kształt oraz masa. Dla produkcji masowej zastosowana zostanie kalkulacja podziałowa jednostopniowa współczynnikowa. Według założeń tej kalkulacji koszt własny liczony jest początkowo dla wyrobu jako iloraz wszystkich kosztów rodzajowych związanych z realizacją produkcji w rozpatrywanym okresie K_r przez iloczyn liczby wyrobów *pb* produkowanych w badanym okresie N_{pb} i współczynnika kosztochłonności *p*-tego wyrobu w_{pb}. Dla wyrobu bazowego współczynnik kosztochłonności wynosi 1,0. Algorytm kalkulacji przedstawiony został poniżej:

$$k_{wł_{pb}} = \frac{\sum_i k_{ri}}{\sum_p N_p \cdot w_p} \quad (1)$$

$$k_{wł_p} = w_p \cdot k_{wł_{pb}} \quad (2)$$

gdzie:

$k_{wł_{pb}}$ - koszt własny produkcji *p*-tego wyrobu przyjętego jako produkt bazowy,

$k_{wł_p}$ - koszt własny produkcji *p*-tego wyrobu,

$\sum_i k_{ri}$ - suma kosztów rodzajowych poniesionych w badanym okresie,

w_p - współczynnik kosztochłonności *p*-tego wyrobu,

N_p - liczba sztuk produkcji *p*-tego wyrobu w badanym okresie [3].

Algorytm zakłada kalkulację kosztu własnego. Jednakże dla zastosowanego modelu poniżej obliczony został koszt wytworzenia brzeszczotu przyjętego za produkt bazowy *(pb)* i drugiego *(p)*, którego współczynnik kosztochłonności wynosi 1,2. Współczynniki kosztochłonności dla brzeszczotów determinowane są wagą danego wyrobu. Dzięki znajomości miesięcznej wielkości produkcji (N_{pb}*=100000szt/miesiąc* i N_p*=100000szt/miesiąc*) i sumy kosztów rodzajowych dla działu obróbki skrawaniem, w którym realizowana jest produkcja brzeszczotów *pb*i *p*(*Kr=44132,64zł/miesiąc)* możliwe jest obliczenie kosztu wytworzenia jednostki wyrobu:

$$k_{w_{pb}} = \frac{44032{,}64}{1\,00\,000 \cdot 1{,}0 + 1\,00\,000 \cdot 1{,}2} = 0{,}20 zł/szt \quad (3)$$

Gdzie $k_{w_{pb}}$ stanowi jednostkowy koszt wytworzenia produkcji brzeszczotu przyjętego jako produkt bazowy. Koszt produkcji brzeszczotu *p* wynosi:

$$k_{w_p} = 1{,}2 \cdot 0{,}20 = 0{,}24 zł/szt \quad (4)$$

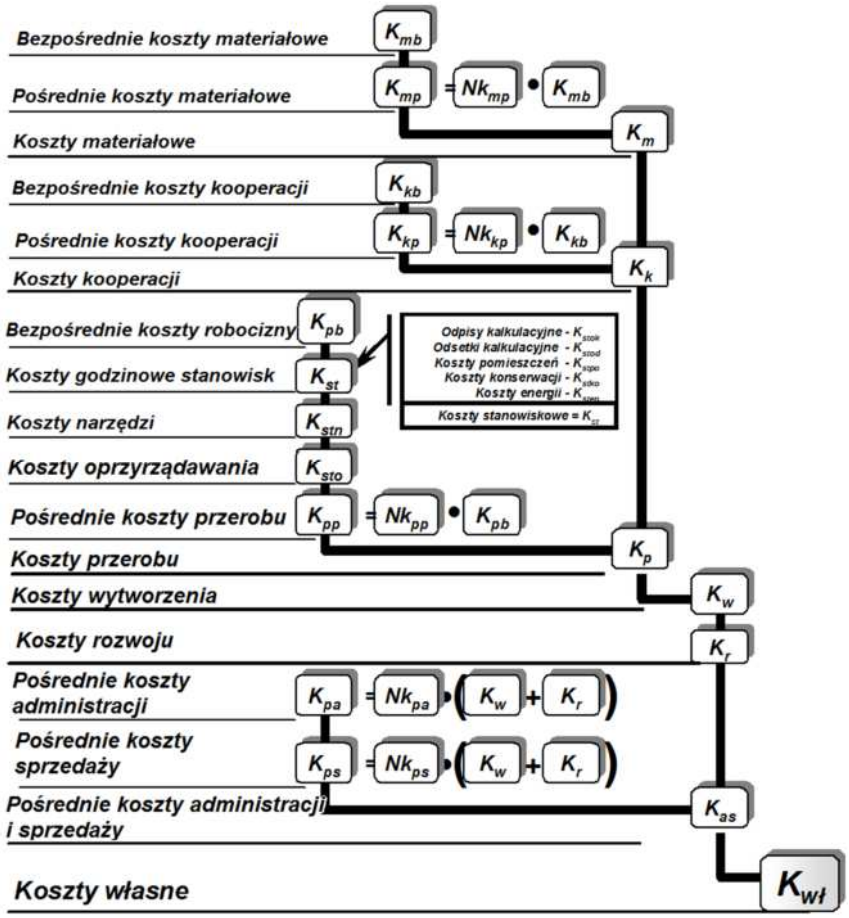
Analogicznie kalkulowane są pozostałe rodzaje brzeszczotów w przedsiębiorstwie na podstawie znajomości kosztów rodzajowych oraz wielkości współczynników kosztochłonności. W omawianym przypadku za koszty rodzajowe przyjmujemy wszystkie koszty w danym okresie pozyskane z arkusza AR.

3.2. Wyrób wytwarzany w warunkach produkcji seryjnej

W przypadku produkcji seryjnej zalecane jest stosowanie kalkulacji doliczeniowej, dlatego w przypadku pił taśmowych i noży taśmowych produkowanych w przedsiębiorstwie zastosowana zostanie kalkulacja doliczeniowa według miejsc powstawania kosztów. Algorytm rozliczania kosztów na nośniki przedstawiony został na rysunku 4.

Kalkulacja doliczeniowa sprowadza się do rozliczenia kosztów bezpośrednich (np. materiały bezpośrednie, płace bezpośrednie, koszty stanowiskowe) na poszczególne wyroby gotowe na podstawie dokumentów źródłowych, koszty

stanowiskowe wg norm czasu maszynowego związanych z miejscami powstawania kosztów, a pozostałe koszty pośrednie (wydziałowe i zarządu, itp.) dolicza się przy pomocy narzutów obliczonych przy pomocy arkusza rozliczeniowego [1]. Zastosowanie kalkulacji doliczeniowej według miejsc powstawania kosztów daje także informacje o strukturze kosztów, co może zostać poddane dalszej analizie w przedsiębiorstwie.



Rys. 4. Algorytm kalkulacji doliczeniowej według miejsc powstawania kosztów [1]

W analizowanym przypadku dzięki danych o normie zużycia materiałów można wyznaczyć koszty materiałowe bezpośrednie *Kmb* oraz koszty materiałowe pośrednie *Kmp,* które stanowią iloczyn kosztów bezpośrednich i narzutu pośrednich kosztów materiałowych *Nkmp* obliczonego przy pomocy arkusza AR:

$$Kmb = 5{,}00 zł/mb \tag{5}$$

$$Kmp = Nkmp \cdot Kmb = 60{,}19\% \cdot 10{,}00 = 3{,}01 zł/mb \quad (6)$$

Zatem koszty materiałowe wynoszą:

$$Km = 5{,}00 + 3{,}01 = 8{,}01 zł/mb \quad (7)$$

Kolejnym krokiem jest wyznaczenie kosztów robocizny bezpośredniej *Krb* na podstawie czasu jednostkowego *tj* i czasu przygotowawczo-zakończeniowego *tpz*:

$$Krb = 3{,}00 zł/mb \quad (8)$$

Koszty stanowiskowe na wydziale produkcyjnym związane z danym miejscem powstawania kosztów składają się z [4]:

– kosztów odpisów amortyzacyjnych *Kstoa* (*Ws* - wartość stanowiska, *l.lateksp*l. – szacowana liczba lat eksploatacji stanowiska)

$$Kstoa = \frac{W_s}{l.lat\ ekspl.} \quad (9)$$

– kosztów odsetek kalkulacyjnych *Kstok (st.%* – stopa odsetek)

$$Kstok = \frac{W_s}{2} \cdot st.\% \quad (10)$$

– kosztów utrzymania powierzchni stanowisk *Kstpo* (*S* - powierzchnia stanowiska, *Kpoj* - koszt utrzymania 1 m^2 powierzchni)

$$Kstpo = S \cdot Kpo_j \quad (11)$$

– kosztów planowych konserwacji *Kstko* (*wsp.ko* – roczny współczynnik konserwacji)

$$Kstko = wsp.ko \cdot W_s$$

lub przez oszacowanie rocznej kwota konserwacji $Kstko$ (12)

– kosztów energii technologicznej *Ksten* (*P* - zainstalowana moc na stanowisku, *wsp. mocy* - współczynnik wykorzystania mocy, *Kenj* - koszt 1 KWh)

$$Ksten = P \cdot wsp.\ mocy \cdot Ken_j \quad (13)$$

Po przeprowadzeniu kalkulacji dla noża taśmowego jednostkowe koszty stanowiskowe wynoszą:

$$Kst = 7{,}50\ zł/mb \quad (14)$$

Koszty specjalnych narzędzi i oprzyrządowania wynoszą 0 zł/mb.

Pośrednie koszty przerobu *Kpp* wyznaczono jako iloczyn narzutu pośrednich kosztów przerobu *Nkpp* wyznaczonego z AR (*Nkpp=140,32%)* i bezpośrednich kosztów pracy *Krb*:

$$Kpp = 140{,}32\% \cdot 3{,}00 = 4{,}21 zł/mb \quad (15)$$

Koszty przerobu *Kp* są sumą kosztów pracy bezpośredniej, kosztów stanowiskowych i pośrednich kosztów przerobu:

$$Kp = Krb + Kst + Kpp = 3{,}00 + 7{,}50 + 4{,}21 = 14{,}71 zł/mb \quad (16)$$

Koszty wytworzenia *Kw* są sumą kosztów materiałowych i kosztów przerobu *Kp*:

$$Kw = Km + Kp = 8{,}01 + 14{,}71 = 22{,}72 zł/mb \quad (17)$$

W dalszej części algorytm kalkulacji doliczeniowej według miejsc powstawania kosztów zakłada doliczanie kosztów rozwoju, pośrednich kosztów administracji i kosztów sprzedaży do kosztu wytworzenia celem otrzymania kosztu własnego. Jednakże, w związku z zastosowaniem modelu hybrydowego koszty te zostaną dodawane w dalszej części artykułu, wspólnie dla wszystkich rodzajów wyrobów.

3.3. Wyrób wytwarzany jednostkowo

W przedsiębiorstwie jednostkowo wytwarzany jest sprzęt sportowy. W takim przypadku rozliczanie kosztów pośrednich tradycyjnymi metodami – tzn. za pomocą narzutów może stać się niewystarczające i powodować małą dokładność obliczeń. W związku z tym dobrym rozwiązaniem dla tego przypadku jest rozliczanie kosztów pośrednich do konkretnych działań, co zgodne jest z założeniami kalkulacji według rachunku kosztów działań.

Koncepcja opiera się na założeniu, że przyczynami powstawania kosztów są działania, a nie bezpośrednio produkty. Metoda rachunku kosztów działań lepiej oddaje powiązanie kosztów z przyczynami ich powstawania. Dodatkowo metoda ta może dostarczyć danych o pełnych kosztach produktów, jak i danych przydatnych do podejmowania decyzji. Koszty działań wyznaczane są według zużycia poszczególnych zasobów do wykonania tychże działań [5].

Dla przykładu przedstawiono obliczenia kosztu wytworzenia na przykładzie wyciągu narciarskiego. Wyciąg narciarski składa się z następujących zespołów przedstawionych w tabeli 1. W tabeli pokazano również uproszczone działania wraz z miarami wielkości danych działań.

Tabela 1. Działania wyodrębnione w produkcji wyciągu narciarskiego

Zespół		**Działanie**		**Miara wielkości działania**
Z1	Zespół stacji początkowej	D1	Produkcja zespołu Z1	Liczba rbg
Z2	Zespół napędowy z liną	D2	Produkcja zespołu Z2	Liczba rbg
Z3	Zespół podpór	D3	Produkcja zespołu Z3	Liczba rbg
Z4	Zespół stacji końcowej	D4	Produkcja zespołu Z4	Liczba rbg

Dla każdego z działań przedstawionych w tabeli odpowiednio rozliczono koszty w arkuszu rozliczeniowym na podstawie liczby roboczogodzin przeznaczonych na produkcję danych zespołów.

Zestawienie kosztów rodzajowych w rozpatrywanym okresie czasu dla każdego z działań w skali miesiąca przedstawiono w tabeli 2.

Tabela 2. Zestawienie kosztów rodzajowych dla poszczególnych działań

Działanie		**Suma kosztów rodzajowych**
D1	Produkcja zespołu Z1	3 046,63 zł
D2	Produkcja zespołu Z2	5 331,60 zł
D3	Produkcja zespołu Z3	3 046,63 zł
D4	Produkcja zespołu Z4	3 808,29 zł
Suma		15 233,14 zł

Koszty przedstawione w tabeli nie zawierają kosztów materiałów bezpośrednich i bezpośrednich kosztów pracy. W związku z tym kalkulacja wyrobu gotowego zawiera wartość kosztów działań zgromadzoną w arkuszu rozliczeniowym powiększoną o *kmb* oraz *krb* dla rozpatrywanego okresu czasu:

$$kw = K_{działań} + kmb + krb \tag{18}$$

$$kw = 15233{,}14 + 900\,000 + 16\,800 = 932033{,}14zł \tag{19}$$

W rozpatrywanym okresie wytwarzania produkowane są jednocześnie 3 wyciągi. Zatem otrzymany koszt wytwarzania należy pomniejszyć trzykrotnie:

$$kw = 932033{,}14 \div 3 = 310677{,}71zł/szt \tag{20}$$

Koszt wytworzenia jednego wyciągu narciarskiego w rozpatrywanym okresie czasu wynosi 310677,71 zł/szt.

3.4. Kalkulacja kosztów pośrednich

W przyjętym algorytmie kalkulacji kosztów uwzględniono wszystkie koszty rodzajowe dla każdego wyrobu, jednakże bez uwzględnienia kosztów administracji, sprzedaży oraz kosztów rozwoju. Doliczenie tych kosztów pośrednich zostanie przeprowadzone wspólnie dla wszystkich wyrobów.

Na podstawie danych z arkusza rozliczeniowego AR obliczono narzuty kosztów pośrednich:

- Koszty rozwoju *Nkpr=21,05%,*
- Pośrednie koszty administracji *Nkap=10,00%,*
- Pośrednie koszty sprzedaży *Nkas=10,00%.*

Wartości narzutów zostaną w dalszej części wykorzystane w kalkulacji kosztów własnych wyrobów:

a. Kalkulacja kosztów wyrobów produkowanych masowo

Koszt wytworzenia brzeszczotu przyjętego za produkt bazowy wyniósł 0,20 zł/szt. Koszt ten zostanie powiększony o koszty pośrednie:

- Koszty rozwoju:

$$Kr = Kw \cdot Nkpr = 0{,}20 \cdot 21{,}05\% = 0{,}04\ zł/szt \quad (21)$$

- Koszty pośrednie administracji:

$$Kpa = Kw \cdot Nkpa = 0{,}20 \cdot 10{,}00\% = 0{,}02\ zł/szt \quad (22)$$

- Koszty pośrednie sprzedaży:

$$Kps = Kw \cdot Nkps = 0{,}20 \cdot 10{,}00\% = 0{,}02\ zł/szt \quad (23)$$

Zatem koszt własny produkcji brzeszczotu dla celów kalkulacji przyjętego jako produkt bazowy wynosi:

$$k_{włb_{pb}} = Kw + Kr + Kpa + Kps \quad (24)$$

$$k_{włb_{pb}} = 0{,}20 + 0{,}04 + 0{,}02 + 0{,}02 = 0{,}28zł/szt \quad (25)$$

b. Kalkulacja kosztów wyrobów produkowanych seryjnie

Koszt wytworzenia noża taśmowego wyniósł 22,72 zł/mb. Koszt ten zostanie powiększony o koszty pośrednie:

- Koszty rozwoju:

$$Kr = Kw \cdot Nkpr = 22{,}72 \cdot 21{,}05\% = 4{,}72\ zł/mb \quad (26)$$

- Koszty pośrednie administracji:

$$Kpa = Kw \cdot Nkpa = 22{,}72 \cdot 10{,}00\% = 2{,}27\ zł/mb \quad (27)$$

- Koszty pośrednie sprzedaży:

$$Kps = Kw \cdot Nkps = 22{,}72 \cdot 10{,}00\% = 2{,}27\ zł/mb \quad (28)$$

Zatem koszt własny produkcji noża taśmowego wynosi:

$$k_{włn} = Kw + Kr + Kpa + Kps \quad (29)$$

$$k_{włn} = 22{,}72 + 4{,}72 + 2{,}27 + 2{,}27 = 31{,}98zł/mb \quad (30)$$

c. Kalkulacja kosztów wyrobów produkowanych jednostkowo

Koszt wytworzenia wyciągu wyniósł 310677,71 zł/szt. Koszt ten zostanie powiększony o koszty pośrednie:

- Koszty rozwoju:

$$Kr = Kw \cdot Nkpr = 310677{,}71 \cdot 21{,}05\% = 65397{,}66\ zł/szt \quad (31)$$

- Koszty pośrednie administracji:

$$Kpa = Kw \cdot Nkpa = 310677{,}71 \cdot 10{,}00\% = 31067{,}77 zł/szt \quad (32)$$

- Koszty pośrednie sprzedaży:

$$Kps = Kw \cdot Nkps = 310677{,}71 \cdot 10{,}00\% = 31067{,}77\ zł/szt \quad (33)$$

Zatem koszt własny produkcji wyciągu narciarskiego wynosi:

$$k_{włn} = Kw + Kr + Kpa + Kps \quad (34)$$

$$k_{włn} = 310677{,}71 + 65397{,}66 + 31067{,}77 + 31067{,}77 = \\ = 438210{,}91 zł/szt \quad (35)$$

Zaproponowany model kalkulacji kosztów pozwala na wyznaczenie kosztów własnych wyrobów produkowanych w różnych wielkościach produkcji i proporcjonalne doliczenie kosztów pośrednich na każdy z wyrobów. Model kalkulacji hybrydowej pomógł lepiej oszacować koszty każdego z wyrobów niż dotychczas stosowany jeden algorytm kalkulacji dla całego przedsiębiorstwa.

4. Koncepcja wsparcia informatycznego rachunku kosztów

Na uwagę zasługuje fakt, że ewidencja kosztów w przekroju działań i procesów wymaga bardziej zaawansowanych rozwiązań informatycznych niż systemy finansowo-księgowe, które znajdują zastosowanie w tradycyjnych metodach kalkulacji kosztów. Informacje o kosztach gromadzone są w tych systemach na kontach syntetycznych i analitycznych i mogą stanowić podstawowe źródło danych o wielkości kosztów. Kalkulacja kosztów z wykorzystaniem różnych metod kalkulacji w tym rachunku kosztów działań powinna rozpoczynać się od gromadzenia informacji nie tylko o wielkości kosztów, ale także o ich nośniach – działaniach, procesach i stanowiskach. Dla tego celu stosowne wydaje się zastosowanie arkusza kalkulacyjnego Excel. Arkusz pozwala na prezentacje nośników kosztów i wielkości tych kosztów w przedsiębiorstwach [2].

Prawidłowy podział kosztów na koszty bezpośrednie i pośrednie w rachunku kosztów wraz z przypisaniem odpowiednio dobranych kluczy podziałowych stanowi podstawowe zagadnienie w prawidłowym zarządzaniu w kalkulacji kosztów. Wsparciem dla tego procesu może być zastosowanie rozwiązań informatycznych. Taki system może nie tylko gromadzić koszty według ich rodzajów, ale także przypisywać je do miejsc ich powstawania na podstawie kluczy podziałowych czy do działań lub procesów za pomocą określonych miar wielkości działań. O ile przypisywanie kosztów do miejsc ich powstawania nie stanowi problemu znając

wielkości kosztów oraz klucze podziałowe, tak przypisanie kosztów do odpowiednich działań wymaga raportów o stawkach nośników kosztów działań. Bieżącą kontrolę codziennych wydatków na działania i procesy umożliwia natomiast system klasy ERP. Dzięki niemu nie trzeba czekać na miesięczny raport o rzeczywistych wielkościach kosztów działań, dzięki czemu możliwe jest szybsze reagowanie na potencjalne niekorzystne zjawiska [2].

Wykorzystywanie systemu ERP oznacza efektywne zarządzanie przedsiębiorstwem. Operacje wykonywane przez użytkowników systemu są rejestrowane we wspólnej bazie danych, co jest powiązane z pełną integracją poszczególnych modułów wchodzących w skład systemu. Do korzyści związanych z implementacją programu zaliczyć można między innymi możliwość sprawowania globalnej kontroli nad przedsiębiorstwem oraz scentralizowanego zarządzania. Dodatkową zaletą jest szybki przepływ informacji między komórkami organizacyjnymi firmy, dzięki jednokrotnemu wprowadzaniu danych do systemu, a także spójność informacji finansowych, dzięki całkowitej autoryzacji procesów księgowych. Zastosowanie w programie technologii internetowych umożliwia zdalną pracę przedsiębiorstwom, które mają wiele oddziałów przy zastosowaniu jednej bazy danych i jednego systemu znajdującej się w centrali. W wyniku tego zarząd firmy otrzymuje pełny obraz przedsiębiorstwa [2].

Arkusze rozliczeniowe prezentują funkcjonalną stronę rozliczania kosztów, natomiast bazy danych obsługują SQL-ową strukturę danych służącą do gromadzenia danych kosztowych oraz danych niezbędnych do rozliczania kosztów na nośniki. Celem kontroli kosztów, bezpośrednie i pośrednie koszty powinny być gromadzone w długim okresie czasowym. Zatem klasyczna baza SQL-owa powinna w tym przypadku zostać zmodyfikowana, tak by być zorientowana na gromadzenie danych w dłuższej perspektywie czasu. Bazy danych wspomagające rachunek kosztów muszą być stworzone w postaci dwuwymiarowych tabeli, pomiędzy którymi występują określone relacje (powiązania). W takiej bazie danych powinny znaleźć się tabele odnoszące się do planu kont funkcjonującego w przedsiębiorstwie wraz z obrotami oraz bilansu otwarcia. Bazy danych znajdują również zastosowanie w technologii OLAP. System informatyczny OLAP *(Online Analitycal Processing)* to technologia, która umożliwia analityczny i szybki dostęp do zgromadzanych w bazach danych informacji. Dane w tego typu systemie zgromadzone są w tzw. kostkach olapowych (*multidimensionalcubs*). Systemy typu OLAP znajdują zastosowanie zwłaszcza we wsparciu rachunku kosztów działań. Znaczna część modeli rachunku kosztów działań skupia się na charakterystyce struktury modelowej uwzględniającej następujące moduły: obiekty kosztowe, działania oraz zasoby wraz z przypisanymi kosztami zasobów do działań, a następnie kosztów działań do obiektów kalkulacyjnych. W dzisiejszej, złożonej rzeczywistości dane finansowe i niefinansowe mogą być analizowane w wielu płaszczyznach. Wielowymiarowa analiza danych biznesowych opartych na kalkulacji rachunku

kosztów działań w przekroju produktów, klientów lub kanałów dystrybucji daje przedsiębiorstwom niemal nieograniczone możliwości analityczne.

5. Podsumowanie

Przedsiębiorstwa wytwarzające wyroby w różnych typach produkcji niejednokrotnie borykają się z problemem kalkulacji kosztów wyrobów gotowych. Często nie jest możliwe zastosowanie jednego algorytmu kalkulacji dla całego przedsiębiorstwa. Wyzwanie stanowi kalkulacja kosztów rodzajowych wyrobów, ale także proporcjonalne doliczenie kosztów pośrednich takich, jak koszty administracji, sprzedaży czy rozwoju. W tego typu systemach produkcyjnych najlepszym rozwiązaniem wydaje się zastosowanie kalkulacji hybrydowej. Kalkulacja hybrydowa łączy w sobie elementy różnych rodzajów kalkulacji kosztów własnych, dzięki czemu zmniejsza ryzyko niedoszacowania lub przeszacowania kosztów wyrobów.

Bibliografia

1. Drury C. (2009) *Management Accounting for Business*. Cengage Learning EMEA
2. Konstantyn K. (2012) *Wykorzystanie systemów informatycznych w budżetowaniu na podstawie rachunku kosztów działań w przedsiębiorstwach komunikacji samochodowej,* Zeszyty Naukowe Uniwersytetu Szczecińskiego nr 58, s. 151-162
3. Matuszek J., Kołosowski M., Krokosz-Krynke Z. (2011) *Rachunek kosztów dla inżynierów*, Warszawa, Polskie Wydawnictwo Ekonomiczne
4. Matuszek J., Więcek D., Kaczmar E. (2018) T*endencje rozwoju w procesach określania kosztów własnych produkcji wyrobów*, Innowacje w zarządzaniu i inżynierii produkcji T. 1, Opole, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją
5. Więcek D., Więcek D. (2015) *Wybrane zagadnienia z rachunku kosztów dla inżynierów*, Bielsko-Biała, Wydawnictwo Naukowe Akademii Techniczno-Humanistycznej w Bielsku-Białej

Streszczenie

Obecnie kładziony jest coraz większy nacisk na kalkulacje kosztów własnych produkowanych wyrobów. Sporządzenie kalkulacji wyrobów może stanowić wyzwanie, zwłaszcza w warunkach wytwarzania produktów o różnej wielkości produkcji. Niniejszy artykuł przedstawia model kalkulacji produktów

w przedsiębiorstwie o niejednorodnym asortymencie i różnym charakterze produkcji.

Summary

Nowadays the validity of product costing is growing. Calculation of costs is challenging, especially in diversified production volume conditions. This article presents a model of products cost calculating in an enterprise with a various assortment and different character of manufacture.

Keywords: product costing, cost accounting

Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN:1897-7421
Nr 13, Listopad 2018
Str: 105-114

Maxim Bushinsky
Nina Tereshko
Olga Mantytskaya
Vera Fedotova
Roman Lanovsky
Scientific-Practical Materials Research Centre of NAS of Belarus
Minsk, Belarus

Magnetoresistance effect in layered cobaltite $Sr_{0.9}Y_{0.1}CoO_{2.63}$

Keywords: crystal structure; magnetic structure; magnetization; magnetoresistance

1. Introduction

Complex oxides of cobalt with a perovskite structure have attracted great interest due to different spin states of the Co^{3+} ion, the existence of interplay between the magnetic and transport properties [1] and the effect of colossal magnetoresistance [2]. The character of the magnetic interactions in cobaltites depends on the spin state of Co^{3+} ions, which can be found in the low spin (t_{2g}^6 , S = 0), intermediate spin ($t_{2g}^5 e_g$, S = 1), and high spin ($t_{2g}^4 e_g^2$, S = 2) states. In the parent compound $LaCoO_3$, a gradual semiconductor-metal transition is associated with a change in the spin state of Co^{3+} ions. On the surface of crystallites in powders and inside epitaxial $LaCoO_3$ films, ferromagnetism with T_C~85 K was observed [3]. The replacement of La ions by Sr ions in the $La_{1-x}Sr_xCoO_3$ system leads to ferromagnetism with the Curie temperature gradually increasing up to 305 K ($SrCoO_3$) [4,5]. The end compound $SrCoO_{3-\gamma}$ can have various structural distortions depending on the conditions of synthesis and oxygen content [4]. The decrease of the oxygen content leads to a transition from the ferromagnetic state with $T_C \approx 305$ K ($\gamma \approx 0$) to the antiferromagnetic state with $T_N \approx 537$ K ($\gamma \approx 0.5$) [5]. It was shown in [6] that a small substitution of Sr ions by rare-earth ions (near 5%) can stabilize the cubic phase $Sr_{0.95}Y_{0.05}CoO_{3-\gamma}$ under air synthesis conditions, whereas the

cubic phase $SrCoO_{3-\gamma}$ is obtained only under high oxygen pressure [4]. Depending on the oxygen content, the sample$Sr_{0.95}Y_{0.05}CoO_{3-\gamma}$ can also be tetragonal ($a_p \times a_p \times 2a_p$, a_p - primitive unit cell parameter, space group *P4/mmm*).

Relatively recently anion-deficient layered cobaltites $Sr_3YCo_4O_{10.5+\delta}$ (the reduced chemical formula is $Sr_{0.75}Y_{0.25}CoO_{3-\gamma}$) were obtained in which rare-earth ions can be ordered [6, 7]. These compounds are predominantly antiferromagnetic with a Neel temperature well above room temperature [7, 8]. A relatively small ferromagnetic component occurs simultaneously with the antiferromagnetic ordering [7, 8]. The basic crystal structure of layered cobaltites $Sr_3LnCo_4O_{10.5+\delta}$(Ln - lanthanide) consists of alternating anion-deficient layers $CoO_{4+\delta}$ and layers formed by CoO_6 octahedra touching the apexes [6-9]. The oxygen vacancies are ordered in the $CoO_{4+\delta}$ layers [6].

A study of layered cobaltite $Sr_3YCo_4O_{10.5}$ by X-ray spectroscopy was carried out in [15], which indicated the presence of ordering of the some 3d orbitals of cobalt ions below the Neel point. So it was suggested that the ferrimagnetic component of this compound was explained by the ordering of the 3d orbitals of Co^{3+} ions in the intermediate spin state in the layers having a stoichiometric content of oxygen ions. However, the ordering of the 3d orbitals of cobalt ions may be associated with anion-deficient layers. Since there are different points of view on the nature of the ferromagnetic component in the layered cobaltites $Sr_{3-x}Ln_xCo_4O_{10.5+\delta}$, we have carefully studied the structure, magnetic and magnetotransport properties of cobaltite $Sr_{0.9}Y_{0.1}CoO_{3-\gamma}$, in which the yttrium and oxygen content is intermediate between oxygen vacancies ordered $SrCoO_{2.5}$ and $Sr_3YCo_4O_{10.5+\delta}$.

2. Experimental procedures

Polycrystalline samples of the sample$Sr_{1-x}Y_xCoO_{2.65}$ ($0.05 \le x \le 0.3$) were obtained by conventional ceramic technology in air. The stoichiometric amounts of Y_2O_3 and Co_3O_4 oxides and $SrCO_3$ carbonate of high purity were thoroughly ground in a planetary ball mill from RETSCH PM-100 for 30 min at a rate of 250 rpm. The Y_2O_3 oxide was heated to 1000°C to remove moisture prior to weighing. The synthesis of the samples was carried out in two stages. Preliminary synthesis was performed at a temperature of 1000 °C. The final synthesis was performed at 1150 – 1200 °C for 7 – 8 h. Then the sample was cooled for 12 h to 300°C. Scanning electron microscope (SEM) LEO 1455 VP was used to study the microstructure. To determine the elemental composition of the samples X-ray microanalysis was conducted using energy dispersive X-ray spectroscopy microanalyzer Aztec Energy Advanced X-Max 80 (EDX). Synchrotron X-ray diffraction studies ($95 \text{ K} \le T \le 420 \text{ K}$) were carried out using the powder diffraction station of the Materials Sciences beamline on Swiss Light Source at the Paul Scherrer Institut (Willigen, Switzerland). Neutron diffraction studies in the temperature range from

10 to 420 K were performed on the D2B high-resolution diffractometer at the Laue-Langevin Institute (Grenoble, France). The FullProf software package was used for the refinement of the crystal and magnetic structure by the Rietveld method [18]. Magnetic and magnetotransport measurements were performed using physical property measurement system (Cryogenic Ltd.) in magnetic fields up to 14 T in the temperature range 5 – 315 K. Conductivity measurements were performed using a four-contact method with indium contacts deposited by ultrasound.

3. Results and discussion

The shape and size of the particles of ceramic samples $Sr_{1-x}Y_xCoO_{2.65}$ ($0.05 \leq x \leq 0.3$) were examined using scanning electron microscope. The investigation of the composition of synthesized samples by registration of element distribution along any given line on the surface has shown a good accordance with prescribed parameters (Fig.1). There is an insignificant variation of the observed of Y, Sr, Co content in different crystallites so it can regard the sample as practically homogeneous on the composition.

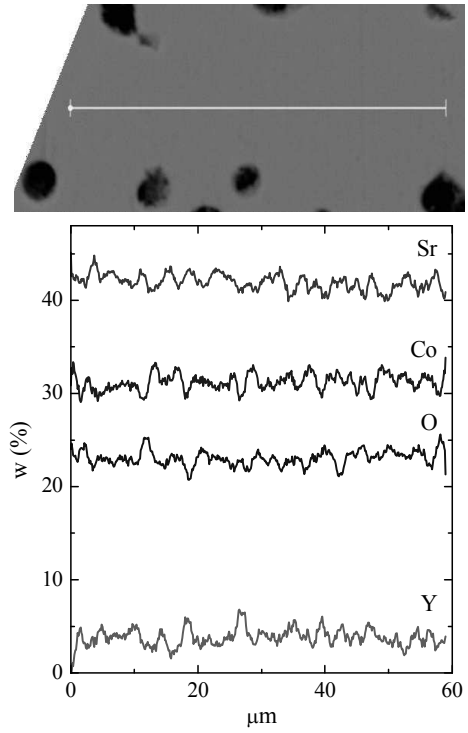


Fig. 1. SEM image and EDX line scan of the surface area of the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$

Synchrotron X-ray diffraction studies of $Sr_{0.9}Y_{0.1}CoO_{2.63}$ composition were carried out in the temperature range 95-420 K. The crystal unit cell parameters of the main structural phase of $Sr_{0.9}Y_{0.1}CoO_{2.63}$ were calculated in the tetragonal space group *I4/mmm* with a superstructure of the type $2a_p \times 2a_p \times 4a_p$ (a_p is the parameter of the primitive cubic cell). It can be seen from figure 2 that the parameters and volume of the unit cell increase monotonically with increasing temperature. The gradual increase of the unit cell volume indicates the absence of any crystal structure or spin transition.

With the increase of the yttrium ion content to 20%, additional diffraction peaks with *hkl* indices 111 and $\bar{1}11$ appear on the X-ray patterns, which disappear at temperatures above 350 K. These peaks can be described within the framework of the monoclinic space group *A2/m* (superstructure $4\sqrt{2}a_p \times 2\sqrt{2}a_p \times 4a_p$) (Fig. 3). The appearance of these peaks has been associated with the spin transition of Co ions or 3d orbital ordering in CoO_6 layers [8-9]. With an increase in temperature above 350 K, a first-order phase transition is observed, accompanied by a crystal structure transition to a cell of the type $2\sqrt{2}a_p \times 2\sqrt{2}a_p \times 4a_p$ (space group *A2/m*).

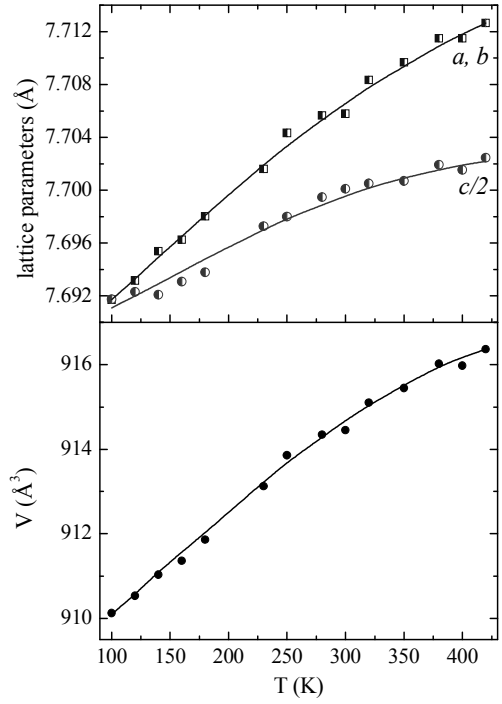


Fig. 2. Temperature dependences of lattice parameters and unit cell volume V for the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ (space group *I4/mmm*)

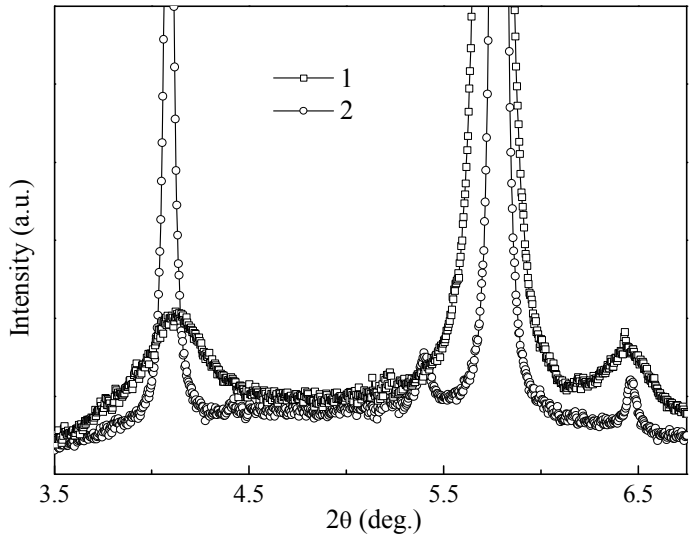


Fig. 3. X-ray powder diffraction patterns for the samples $Sr_{0.9}Y_{0.1}CoO_{2.63}$ (space group *I4/mmm*+ *A2/m*) (1) and $Sr_{0.8}Y_{0.2}CoO_{2.65}$ (*A2/m)* (2) at 100 K over the range of angles 3.5-6.5

Neutron diffraction studies of $Sr_{0.9}Y_{0.1}CoO_{3-\gamma}$ were carried out at 10 K, 260 K, and 400 K. The calculations of the crystal and magnetic structure of the major phase of the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ at 10 K and 400 K were carried out within the framework of the space group *I4/mmm* and unit cells of the type $2a_p \times 2a_p \times 4a_p$. According to the obtained data, the crystal structure consists of layers alternating along the *c* axis of CoO_6 octahedra and anion-deficient $CoO_{4+\gamma}$ layers, like what has been previously shown for other layered cobaltites $Sr_{0.75}Y_{0.25}CoO_{3-\gamma}$ [7]. At 400 K the monoclinically distorted phase is absent. Refined from the neutron diffraction data, the oxygen content in the sample is close to 2.63, which means that the cobalt ions are in the trivalent state. Analysis of the neutron diffraction data obtained at 10 K showed that the basic magnetic structure can be described by an antiferromagnetic ordering of the G type with a magnetic cell of the type $2a_p \times 2a_p \times 4a_p$ with two different magnetic positions in the layers of CoO_6 octahedra and anion-deficient layers. The magnetic moments in anion-deficient layers and in layers of CoO_6 octahedra at 10 K are equal to $\sim 2\ \mu_B$ and $\sim 1.5\ \mu_B$, respectively. The magnetic contribution to the diffraction pattern is no longer present at 400 K.

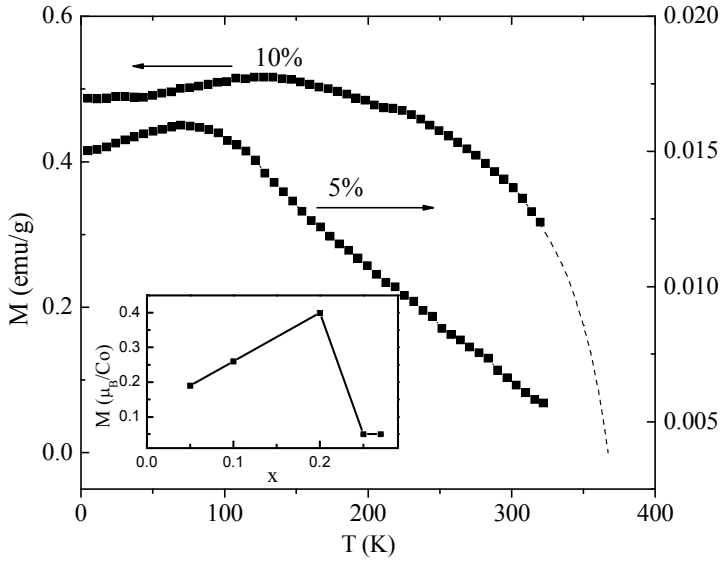


Fig. 4. Temperature dependences of the magnetization for the samples $Sr_{0.95}Y_{0.05}CoO_{2.63}$ and $Sr_{0.9}Y_{0.1}CoO_{2.63}$. The Inset shows the concentration dependence of the magnetization for B = 14 T at 5 K for $Sr_{1-x}Y_xCoO_{3-\gamma}$

From the magnetic measurements, the Néel temperature was roughly estimated to about $T_N \approx 380$ K using M(T) approximation by dashed curve in figure 4. As the temperature decreases from 320 K to 5 K, first the magnetization increases, reaching a maximum value of 0.5 emu/g at T ≈ 130 K, and then decreases slightly. The residual magnetization is ~ 0.022 μ_B per cobalt ion at 5 K and slightly decreases to 0.017 μ_B per cobalt ion at 220 K. The magnetization is not saturated in magnetic fields up to 14 T. The sample $Sr_{0.95}Y_{0.05}CoO_{2.63}$ exhibits much smaller magnetization. The magnetization as function of the Y content for B = 14 T at 5 K is plotted in the Inset of the figure 4 and shows a growth up to x=0.2 and then drops (x=0.25) apparently due to the crystal structure transition.

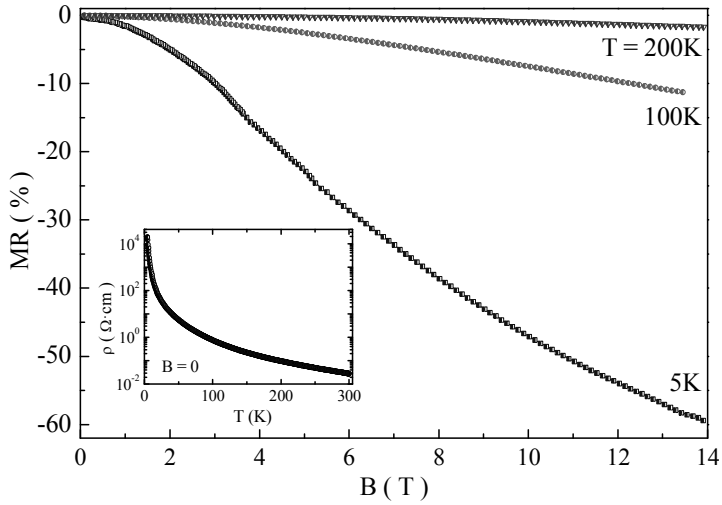


Fig. 5. Dependence of the magnetoresistance for the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ at 5K, 100K and 200K (the Inset shows temperature dependence of electrical resistivity B = 0)

The measurement of the electric transport properties of the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ shows that the temperature dependence of the electrical conductivity in this temperature range has a semiconductor character (see the inset of Fig. 5). The magnitude of the resistivity increases from ~ 0.02 Ω·cm at 300 K to ~ 20 kΩ·cm at helium temperature. Anomalies of the temperature dependence of the electrical resistivity were not found. The negative magnetoresistance ($MR = (\rho(0) - \rho(H))/\rho(0) \bullet 100\%$) reaches 58% at 5 K in a magnetic field B=14 T (Fig.5). With increasing temperature, the magnetoresistance strongly decreases to 1.6% in a field of 14 T at 200 K.

4. Conclusions

The obtained data indicate that the crystal structure changes from tetragonal *P4/mmm* ($a_p \times a_p \times 2a_p$) to monoclinic *A2/m* through the intermediate tetragonal phase *I4/mmm* ($2a_p \times 2a_p \times 4a_p$) in the system $Sr_{1-x}Y_xCoO_{3+\gamma}$ (the γ value corresponds to finding the majority of cobalt ions in the oxidation state 3+) with increasing yttrium content. The simultaneous disappearance of the monoclinic phase and of the spontaneous magnetization evidence that the type of crystal structure determines the occurrence of the ferromagnetic component.

Acknowledgments

This work is supported by the Belarusian Republican Foundation for Fundamental Research (Project F18R-159).

References

1. N.B. Ivanova, S.G. Ovchinnikov, M.M. Korshunov, I.M. Eremin, N.V. Kazak, Specific features of spin, charge, and orbital ordering in cobaltites, Phys. Usp. 52(2009) 789–810.
2. A. Maignan, C. Martin, D. Pelloquin, N. Nguyen, B.Raveau, Structural and Magnetic Studies of Ordered Oxygen-Deficient Perovskites $LnBaCo_2O_{5+\delta}$, Closely Related to the "112" Structure, J. Solid State Chem. 142 (1999) 247 – 260.
3. D. Fuchs, C. Pinta, T. Schwarz, P. Schweiss, P. Nagel, S. Schuppler, R. Schneider, M. Merz, G. Roth, H.v. Löhneysen, Ferromagnetic order in epitaxially strained $LaCoO_3$ thin films, Phys. Rev. B 75 (2007) 144402.
4. Y. Long, Y. Kaneko, Sh. Ishiwata, Y. Taguchi, Y. Tokura, Synthesis of cubic $SrCoO_3$ single crystal and its anisotropic magnetic and transport properties, J. Phys.: Condens. Matter 23 (2011) 245601.
5. A. Muñoz, C. de la Calle, J.A. Alonso, P.M. Botta, V. Pardo, D. Baldomir, J. Rivas, Crystallographic and magnetic structure of $SrCoO_{2.5}$ brownmillerite: Neutron study coupled with band-structure calculations, Phys. Rev. B 78 (2008) 054404.
6. M. James, D. Cassidy, K.F.Wilson, J. Horvat, R.L. Withers, Oxygen vacancy ordering and magnetism in the rare earth stabilised perovskite form of "$SrCoO_{3-\delta}$", Solid State Sciences 6 (2004) 655 – 662.
7. I.O. Troyanchuk, D.V. Karpinsky, V.M. Dobryanskiĭ, A.N. Chobot, G.M. Chobot, A.P. Sazonov Magnetic transformations in the $Sr_{0.78}Y_{0.22}Co_{1-x}Fe_xO_{3-\gamma}$ system with a perovskite structure, JETP 108 (2009) 428–434.
8. I.O. Troyanchuk, D.V. Karpinsky, A.P. Sazonov, V. Sikolenko, V. Efimov, A. Senyshyn Effect of iron doping on magnetic properties of $Sr_{0.78}Y_{0.22}CoO_{2.625+\delta}$-layered perovskite, J. Mater. Sci. 44(2009) 5900–5908.
9. D.V. Sheptyakov, V.Yu. Pomjakushin, O.A. Drozhzhin, S.Ya. Istomin, E.V. Antipov, I.A. Bobrikov, A. M. Balagurov Correlation of chemical coordination and magnetic ordering in $Sr_3YCo_4O_{10.5+\delta}$ (δ=0.02 and 0.26), Phys. Rev. B 80 (2009) 024409.

10. H. Nakao, T. Murata, D. Bizen, Y. Murakami, K. Ohoyama, K. Yamada, S. Ishiwata, W. Kobayashi, I. Terasaki, Orbital Ordering of Intermediate-Spin State of Co^{3+} in $Sr_3YCo_4O_{10.5}$, J. Phys. Soc. Jpn. 80(2011) 023711.
11. T. Roisnel, J. Rodríquez-Carvajal, WinPLOTR: A Windows Tool for Powder Diffraction Pattern Analysis, Mater. Sci. Forum 378-381 (2001) 118 – 123.

Abstract

The structure and the magnetic and magnetotransport properties of the perovskite sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ have been studied using different diffraction methods and magnetization and conductivity measurements. Synchrotron X-ray diffraction shows that the sample is structurally two-phase. The majority phase has a tetragonally distorted unit cell and is described by the space group *I4/mmm*. A very strongly broadened superstructure peak observed at small angles in X-ray diffraction patterns at temperatures below 400 K are explained by the existence of a monoclinic phase with large unit cell whose phase fraction is much smaller than that of the tetragonal phase, but which is dominant in the sample $Sr_{0.8}Y_{0.2}CoO_{2.65}$. The spontaneous magnetization strongly increases with increasing the Y content up to 20% due to formation of the monoclinic phase. The magnetic structure is predominantly antiferromagnetic G-type with magnetic moments 1.5 μ_B in the layers of CoO_6 octahedra and 2 μ_B in the anion-deficient $CoO_{4+\gamma}$ layers. The electrical conductivity of the sample $Sr_{0.9}Y_{0.1}CoO_{2.63}$ has semiconducting character. The magnetoresistance reaches 58% for the field B = 14 T at 5 K and decreases strongly with the increasing temperature and Y content.

Streszczenie

Struktura i właściwości magnetyczne i magnetotransportowe perowskitu $Sr_{0.9}Y_{0.1}CoO_{2.63}$ zostały zbadane przy użyciu różnych metod dyfrakcyjnych oraz pomiarów namagnesowania i przewodnictwa. Dyfrakcja rentgenowska mierzona na synchrotronie pokazuje, że próbka ma strukturę dwufazową. Główna faza ma tetragonalnie zniekształconą komórkę elementarną i jest opisana przez grupę przestrzenną I4 /mmm.

Pik o bardzo mocno poszerzonej superstrukturze obserwowano pod niewielkimi kątami w dyfraktogramach rentgenowskich w temperaturach poniżej 400 K i jest związany z istnieniem fazy monoklinowej o dużej komórce elementarnej, której frakcja fazowa jest znacznie mniejsza niż faza tetragonalna, ale która jest dominujący w próbce $Sr_{0.8}Y_{0.2}CoO_{2.65}$. Spontaniczne namagnesowanie silnie wzrasta wraz ze wzrostem zawartości Y do 20% z powodu tworzenia się fazy monoklinowej.

Struktura magnetyczna jest w przeważającej mierze antyferromagnetyczną typu G z momentami magnetycznymi 1,5 μ_B w warstwach oktaedrycznych $CoO_{4+\gamma}$ i 2 μ_B w warstwach $CoO_{4+\gamma}$ z niedoborem anionów. Przewodność elektryczna próbki $Sr_{0.9}Y_{0.1}CoO_{2.63}$ ma charakter półprzewodnikowy. Magnetooporność osiąga 58% dla pola B = 14 T przy 5 K i silnie się zmniejsza wraz ze wzrostem temperatury i zawartością Y.

Słowa kluczowe: struktura krystaliczna; struktura magnetyczna; magnetyzacja; magnetooporność