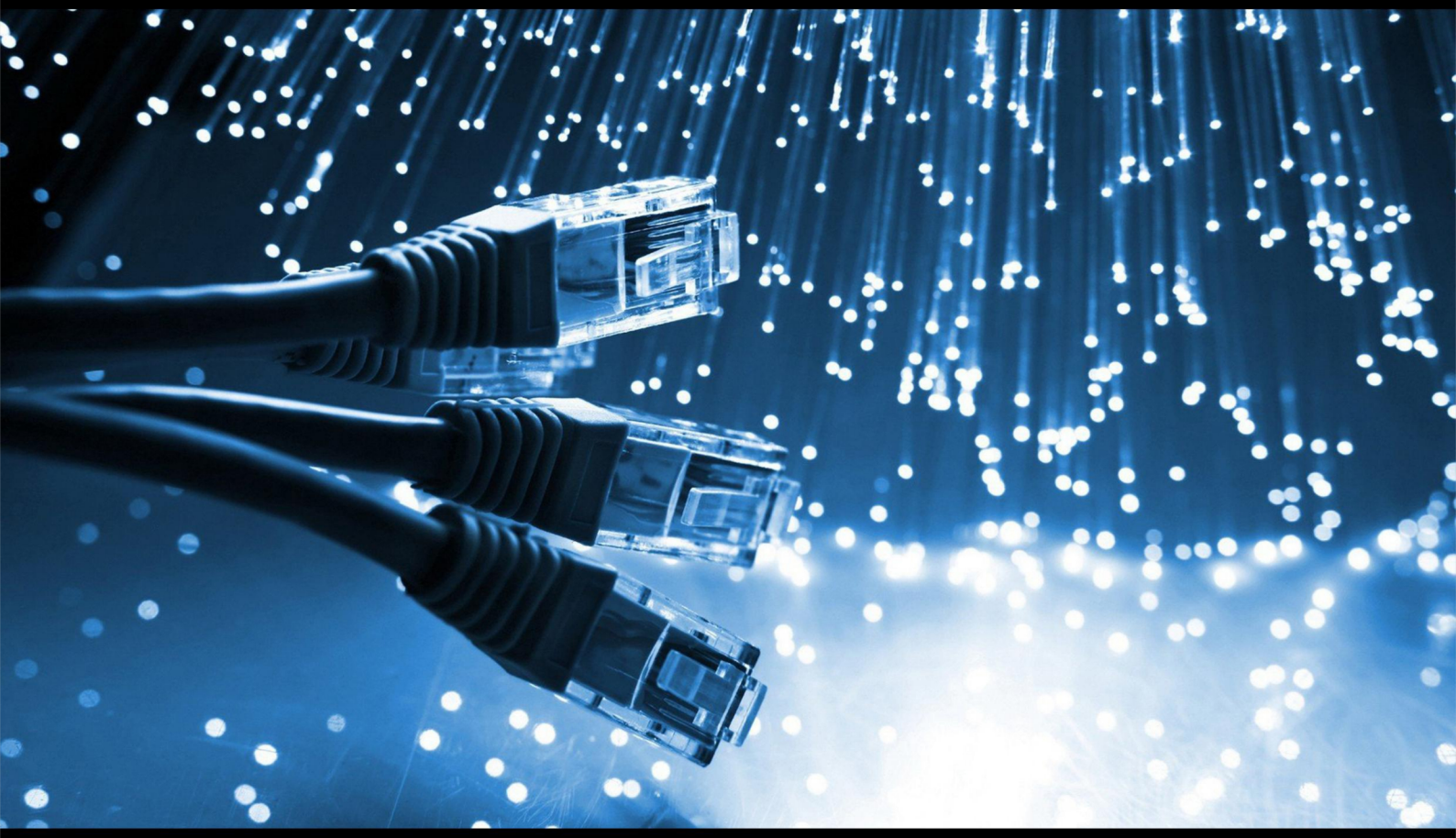
JCSI

Journal of Computer Sciences Institute

Volume 18/2021



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Katedra Informatyki
Wydział Elektrotechniki i Informatyki

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

Redaktor naczelny:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Redaktor techniczny:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Recenzenci numeru:

dr inż. Marcin Badurowicz
dr inż. Piotr Kopniak
dr inż. Sylwester Korga
dr inż. Marek Miłosz, prof. PL
dr inż. Maria Skublewska-Paszkowska
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Maciej Pańczyk

Skład komputerowy:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Projekt okładki:
Marta Zbańska

JCSI Editorial
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
Lublin University of Technology
ul. Nadbystrzycka 36 b
20-618 Lublin, Poland

Editor in Chief:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Assistant editor:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Reviewers:

Marcin Badurowicz
Piotr Kopniak
Sylwester Korga
Marek Miłosz
Maria Skublewska-Paszkowska
Małgorzata Plechawska-Wójcik
Maciej Pańczyk

Computer typesetting:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 1–6
Received: 06 November 2020
Accepted: 08 December 2020

Natural interfaces in VR - comparative analysis

Naturalne interfejsy w VR - analiza porównawcza

Adrian Madoń *, Dawid Majdanik*, Tomasz Szymczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The article presents the results of a comparative analysis of contemporary virtual reality devices. The analysis focuses on both the analysis of technical parameters of the goggles as well as comparison of natural interfaces. The following devices were tested: HTC Vive, Oculus Rift, PlayStation VR, Samsung Gear VR. The most ergonomic and user-friendly interface turned out to be Oculus Rift, while goggles Samsung Gear VR were the worst from tested devices.

Keywords: natural interfaces; virtual reality; comparative analysis

Streszczenie

W artykule zaprezentowano wyniki analizy porównawczej współczesnych urządzeń umożliwiających projekcję wirtualnego środowiska. Analiza dotyczyła zarówno analizy parametrów technicznych samych gogli jak również porównania ergonomii interfejsów. Przetestowano następujące urządzenia: HTC Vive, Oculus Rift, PlayStation VR, Samsung Gear VR. Najbardziej ergonomicznym i przyjaznym użytkownikowi interfejsem okazał się Oculus Rift, natomiast najgorzej w teście wypadły gogle Samsung Gear VR.

Słowa kluczowe: naturalne interfejsy; wirtualna rzeczywistość; analiza porównawcza

*Corresponding authors

Email address:* **adrian.madon@pollub.edu.pl** (A. Madoń), **dawid.majdanik@pollub.edu.pl** (D. Majdanik)

1. Wstęp

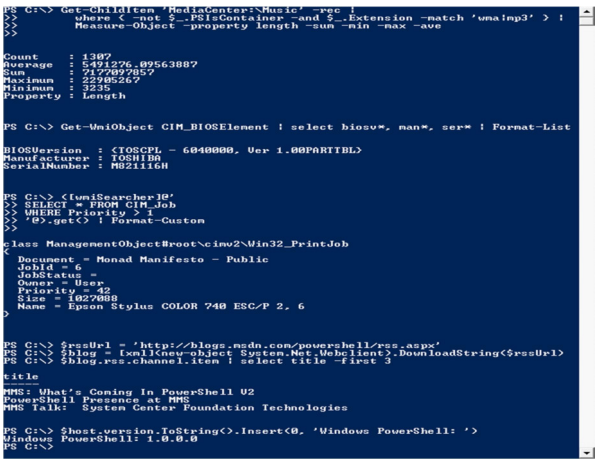
Wirtualna rzeczywistość (ang. *Virtual Reality*) to nowa i rozwijająca się technologia [1]. Jej założeniem jest imitacja wirtualnego świata i jego elementów w taki sposób, by osoba korzystająca z tej technologii mogła poczuć się jakby rzeczywiście była w tym wirtualnym miejscu [2]. Stopień w jakim użytkownik odbiera VR nazywany jest immersją [3]. Korzystanie z wirtualnej rzeczywistości dąży do uzyskania jak największego stopnia immersji.

Popularność oraz szybki rozwój przyczynia się do ogromnej sprzedaży urządzeń obsługujących VR [4]. Wirtualna rzeczywistość otwiera przed ludźmi nowe możliwości, nowe problemy oraz nowe rozwiązania. Technologia ta swoją popularność zawdzięcza wielu aspektom. Jednym z powodów takiej popularności jest powszechny dostęp do aplikacji oraz urządzeń, które obsługują wirtualną rzeczywistość. Kolejnym aspektem jest udostępniane API producentów urządzeń VR, co pomaga w rozpowszechnianiu i tworzeniu aplikacji nawet bez dużej wiedzy programistycznej. Wirtualna rzeczywistość swoje zastosowanie znajduje nie tylko w gałęzi rozrywki pod postacią gier i programów, ale także szeroko wykorzystywana jest do symulacji i nauczania ludzi w medycynie [5] oraz wojsku [6]. Tak duży zakres zastosowań przekłada się na ogromną i wciąż rosnącą liczbę użytkowników, co z kolei przekłada się na potrzebę przystosowania oraz ułatwienia korzystania z tej technologii a także ciągłe poprawki i udogodnienia.

2. Współczesne interfejsy użytkownika

Aplikacja oraz użytkownik wymieniają między sobą informacje za pomocą interfejsów. Interfejsy można podzielić na tekstowe, graficzne oraz naturalne [7].

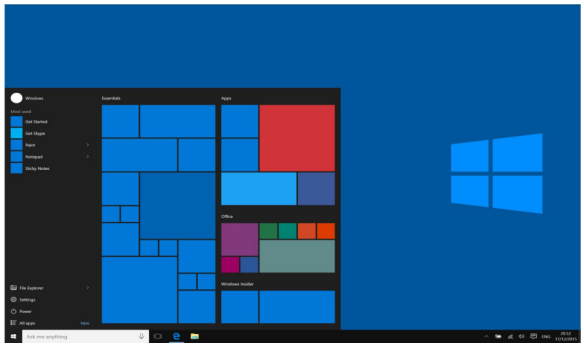
Przedstawicielem interfejsu tekstowego jest CLI (ang. *Command Line Interface*), czyli wiersz poleceń - jest to najprostszy interfejs tekstowy, w którym użytkownik wysyła sygnały do programu za pomocą klawiatury czy też specjalnych skryptów. Jest to jeden z pierwszych interfejsów, które pojawiły się w codziennym użytku. Można je spotkać do tej pory, na przykład w konsoli systemu Windows lub Linux [8].



Rysunek 1: Rysunek Powershell systemu Windows [8].

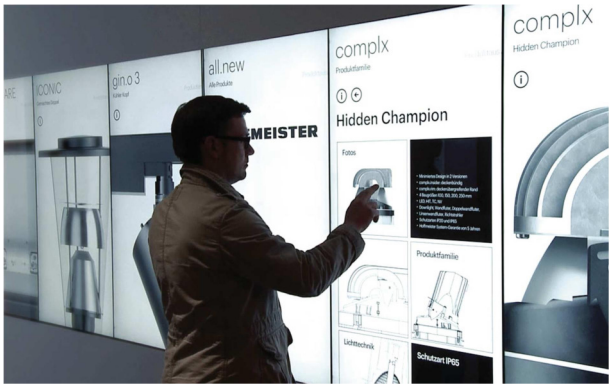
GUI (ang. *Graphical User Interface*), czyli interfejs graficzny, w którym użytkownik komunikuje się z programem za pomocą myszki lub dotyku. Jest to obecnie

jeden z najbardziej popularnych interfejsów. Używane one są w najpopularniejszych systemach operacyjnych Windows oraz na większości urządzeń przenośnych korzystających z systemów Android lub iOS [8].



Rysunek 2: Szkielet menu w systemie Windows 10 [8].

NUI (ang. *Natural User Interface*), czyli interfejs naturalny, w którym użytkownik porozumiewa się z programem za pomocą naturalnych poleceń, takich jak ruch ręką, ruch głową czy też ruch ciała. Najczęściej spotykanym interfejsem naturalnym jest GUI smartfonów czy też innych urządzeń z ekranem dotykowym [9].



Rysunek 3: Naturalny interfejs graficzny [9].

3. Środowisko testowe

Analiza porównawcza zarówno interfejsów jak i sprzętu została wykonana na następujących urządzeniach:

- HTC Vive,
- Oculus Rift,
- PlayStation VR,
- Samsung Gear VR.

W zakresie analizy porównawczej parametrów technicznej porównane zostały takie czynniki jak: waga gogli, ich cena w dniu premiery oraz generowana przez nie grafika. W zakresie analizy interfejsów porównane zostały takie czynniki jak: intuicyjność interfejsu głównego, standardowo dołączone kontrolery, dostępne dodatkowe urządzenia peryferyjne oraz występujące w nich interfejsy naturalne.

HTC Vive są to gogle stworzone przy współpracy firm HTC oraz Valve. Pierwszy model urządzenia HTC Vive został wydany na rynku światowym na początku kwietnia 2016 roku. Gogle te stosują innowacyjną technologię śledzenia ruchów głowy użytkownika i kontrolerów o nazwie „room scale". Polega ona na wykorzystaniu dwóch stacji bazowych, które tworzą 360 ° wirtualną przestrzeń, a następnie emitują światła podczerwieni do kontrolera i gogli w celu wykrycia aktualnej pozycji użytkownika.

Rysunek 4: Gogle HTC Vive z kontrolerami oraz dwiema stacjami bazowymi [10].

Poniższa tabela 1 przedstawia główne parametry techniczne występujące w poszczególnych wersjach gogli HTC Vive.

Tabela 1: Porównanie parametrów technicznych HTC Vive, HTC Vive Cosmos, HTC Vive Pro.

	HTC Vive	HTC Vive Cosmos	HTC Vive Pro
Typ Ekranu	Podwójny AMOLED 3.6 cala	Podwójny 3.4 cala	Podwójny AMOLED 3.5 cala
Rozdzielczość [px]	2160x1200	2880x1700	2880x1600
Częstotliwość odświeżania [Hz]	90	90	90
Pole widzenia [°]	110	110	110
Sensory	SteamVR tracking, G-Sensor, żyroskop, czujnik zbliżeniowy	G-Sensor, żyroskop, ustawienia komfortu oczu (IPD)	SteamVR tracking, G-Sensor, żyroskop, czujnik zbliżeniowy, ustawienia komfortu oczu (IPD)
Ergonomia	Regulacja rozstawu okularów i soczewki	Regulacja rozstawu okularów i soczewki, odchylany daszek, regulowane ustawienia komfortu oczu (IPD)	Regulacja rozstawu okularów i soczewki, odchylany daszek, regulowane ustawienia komfortu oczu (IPD)

Oculus Rift to gogle mające swoje początki na crowdfundingowym serwisie - Kickstarter, na której uzbierało niemalże 2.5 mln dolarów. Gogle te do śledzenia pozycji gogli jak i kontrolerów wykorzystuje system o nazwie „Oculus Insight". Polega on na przewidzeniu trajektorii ruchów Oculus Rift za pomocą pięciu kamer zamieszczonych na goglach oraz światła podczerwieni LED na kontrolerach.

Rysunek 5: Gogle Oculus Rift z kontrolerami [11].

Poniższa tabela przedstawia główne parametry techniczne występujące w poszczególnych wersjach gogli Oculus.

Tabela 2: Porównanie parametrów technicznych Oculus Rift, Oculus Rift S, Oculus Quest, Oculus Go.

	Oculus Rift	Oculus Rift S	Oculus Quest
Rodzaj zestawu	wymaga komputera	wymaga komputera	All in one
Typ Ekranu	Podwójny OLED	Pojedynczy LCD	Podwójny OLED
Rozdzielczość [px]	2160x 1200	2560x 1440	2880x 1600
Częstotliwość odświeżania [Hz]	90	80	72
Regulacja dystansu soczewek	Tak	Nie	Tak

PlayStation VR to gogle VR stworzone przez firmę Sony wykorzystujące konsolę PS4 do generowania grafiki. Do śledzenia trajektorii ruchów użytkownika służy specjalna kamera do PS4, która śledzi emitowane niebieskie światło LED przez gogle oraz ruchy w kontrolerach.

Rysunek 6: Gogle PlayStation VR [12].

Poniższa tabela przedstawia główne parametry techniczne gogli PlayStation VR.

Gogle Samsung Gear VR zostały zaprojektowane przez Samsung Electronics z pomocą Oculus VR. Urządzenie to współpracuje jedynie z telefonami Samsunga, zaczynając od Galaxy S6, a kończąc na najnowszych Galaxy S10+ i Note 9.

Tabela 3: Opis parametrów technicznych gogli PlayStation VR.

Typ Ekranu	Pojedynczy 5.7 cala
Rozdzielczość [px]	1920x1080 pełny kolor OLED RGB (1920xRGBx1080)
Częstotliwość odświeżania [Hz]	90 lub 120 (w zależności od aplikacji)

Rysunek 7: Gogle Samsung Gear VR [13].

Poniższa tabela przedstawia główne parametry techniczne gogli Samsung Gear VR.

Tabela 4: Opis parametrów technicznych gogli Samsung Gear VR.

Typ Ekranu	Brak - ekranem jest telefon komórkowy
Rozdzielczość [px]	Taka sama jak telefonu
Częstotliwość odświeżania [Hz]	Taka sama jak telefonu
Pole widzenia [°]	101 (w przypadku starszych modeli 96)

4. Analiza porównawcza

Celem analizy porównawczej jest zdefiniowanie, które urządzenie posiada najlepsze parametry techniczne oraz najbardziej naturalny interfejs.

4.1. Kryteria analizy

Analiza podzielona jest na dwie części, pierwsza porównuje parametry techniczne urządzeń, druga natomiast skupia się na porównaniu interfejsów naturalnych. Obie analizy dokładnie opisują poszczególne cechy w skali od 0 do 5, gdzie:

- 0 oznacza brak możliwości ocenienia lub brak cechy,
- 1 oznacza nieakceptowalną cechę,
- 2 oznacza słabą akceptowalność,
- 3 oznacza dostatecznie akceptowalną cechę,
- 4 oznacza dobrą akceptowalność,
- 5 oznacza najlepszą akceptowalność.

W ogólnym zestawieniu umożliwi to na podsumowanie oraz ułożenie w kolejności wyników, które otrzymano analizując wybrane urządzenia, co z kolei pozwoli na wybranie najlepszego urządzenia pod względem technicznym, pod względem naturalnych interfejsów oraz najlepszego urządzenia łącznie.

4.2. Analiza parametrów technicznych

Poniższy rozdział zawiera szczegółową analizę parametrów technicznych poszczególnych urządzeń VR. Do analizy zostały wybrane następujące aspekty gogli:

- waga gogli,
- cena w dniu premiery,
- grafika.

Do oceny analizy wagi urządzeń VR zostało wykorzystane badanie LeClair [14] mówiące, że czym mniejsza waga gogli tym większe subiektywne uczucie komfortu u użytkowników. Najwyższą ocenę (5) otrzymały gogle posiadającą wagę poniżej 500 g a każde kolejne 100 g powyżej 500 g powoduje spadek oceny o 1.

Wprowadzone kryterium punktowe zostało ustalone empirycznie na podstawie odczucia użytkownika. Ciężkie gogle są nieprzyjemne w obcowaniu. Wspomniane 100 g to zauważalna i odczuwalna waga.

Tabela 5: Oceniona analiza wagi poszczególnych zestawów VR.

Nazwa gogli	Waga [g]	Ocena
Samsung Gear VR bez telefonu	345	5
Oculus Go	468	5
Oculus Rift	470	5
HTC Vive	470	5
Oculus Rift S	510	4
Oculus Quest	571	4
PlayStation VR	610	3
HTC Vive Cosmos	645	3
HTC Vive Pro	803	1

Kolejnym kryterium analizy technicznej urządzeń VR była ich cena w dniu premiery. Wszelkie nowoczesne gogle VR są bardzo kosztowne. Wynika to z tego, że te urządzenia zazwyczaj potrzebują wyświetlacza o wysokiej rozdzielczości, wielu czujników ruchu gogli oraz na kontrolerach i urządzenia do generowania grafiki (konsola, komputer stacjonarny lub telefon).

Cena produktu dla przeciętnych klientów jest zazwyczaj jednym z głównych wyznacznikiem zakupu produktu, więc najwyższą ocenę otrzymały urządzenia najtańsze w dniu premiery.

Tabela 6: Oceniona analiza ceny w dniu premiery poszczególnych zestawów VR.

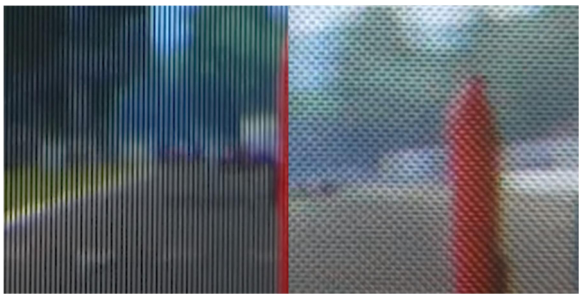
Nazwa gogli	Cena [$]	Ocena
Samsung Gear VR	99.99	5
Oculus Go	199	5
Oculus Quest	399	4
PlayStation VR	399	4
Oculus Rift S	399	4
HTC Vive	599	3
Oculus Rift	599.99	3
HTC Vive Pro	799	2
HTC Vive Cosmos	900	1

Ostatnim kryterium analizy technicznej urządzeń VR była grafika. Analizę grafiki przeprowadzonej dwoma sposobami. Pierwszym sposobem było zauważenie różnic w generowanej grafice pomiędzy poszczególnymi goglami VR, natomiast drugą metodą było badanie występowania efektu „screen-door", który polega na pojawianiu się czarnych artefaktów przy dużym przybliżeniu obrazu (Rys. 8 i Rys. 9).

a) b)

Rysunek 8: Porównanie modelu z soczewek gogli VR a) HTC Vive b) HTC Vive Pro [15].

a) b)



Rysunek 9: Porównanie „screen-door effect" dla urządzeń VR a) Oculus Rift S b) HTC Vive Pro [16].

Poniższa tabela zawiera ocenioną analizę grafiki badanych urządzeń VR.

Tabela 7: Oceniona analiza grafiki poszczególnych zestawów VR.

Nazwa gogli	Ocena grafiki
Oculus Rift S	5
HTC Vive Pro	4
HTC Vive Cosmos	4
PlayStation VR	3
Oculus Quest	3
Oculus Rift	3
HTC Vive	2
Oculus Go	1
Samsung Gear VR	1

4.3. Analiza interfejsów naturalnych

Poniższy rozdział zawiera analizę zdefiniowanych na podstawie materiałów parametrów poszczególnych urządzeń VR. Do analizy zdefiniowano następujące kategorie:

- menu główne,
- dodatkowe urządzenie peryferyjne - rękawice,
- standardowe urządzenie peryferyjne - kontrolery,
- wykrywanie naturalnych ruchów.

Pierwszym urządzeniem poddanym analizie jest HTC Vive. Menu główne zaimplementowane w urządzeniu HTC Vive posiada wiele udogodnień związanych z jego obsługą, posiada zaimplementowane wykrywanie wielu naturalnych ruchów oraz dba o odczucia wizualne i dźwiękowe użytkownika. Standardowy kontroler dołączony do urządzenia posiada rozstawione przyciski w taki sposób, że ręka układa się w naturalnej pozycji, nie jest on za ciężki co powoduje, że jego używanie jest bardzo przyjemnie i nie męczy dłoni. Do

HTC Vive dostępne jest wiele urządzeń peryferyjnych, które można podłączyć bez żadnych problemów, ponieważ posiada on wbudowaną obsługę urządzeń trzecich. Ocena wyżej opisanych parametrów przedstawiona jest w tabeli 8.

Tabela 8: Tabela przedstawiająca ocenę poszczególnych elementów analizy HTC Vive.

Element analizy	Ocena
Menu główne	5
Dodatkowe urządzenie peryferyjne - rękawice	5
Standardowe urządzenie peryferyjne - kontrolery	5
Wykrywanie naturalnych ruchów	5

Kolejne urządzenie poddane analizie to Oculus Rift. Główny interfejs graficzny tego urządzenia, podobnie jak HTC Vive, posiada zaimplementowane wiele udogodnień, porusza zmysł wzroku oraz słuchu i obsługuje wiele ruchów naturalnych. Standardowy kontroler dołączony do urządzenia jest wygodny w użytkowaniu, jednak rozstaw jego przycisków powoduje, że ręka nie do końca układa się w naturalnej pozycji, co przy dłuższym użytkowaniu powoduje zmęczenie dłoni. Dodatkowe urządzenia peryferyjne są obsługiwane przez te okulary, jednak nie wszystkie modele można z nimi połączyć, niektóre modele nie są przystosowane do współpracy z Oculus Rift. Ocena wyżej opisanych parametrów przedstawiona jest w tabeli 9.

Tabela 9: Tabela przedstawiająca ocenę poszczególnych elementów analizy Oculus Rift.

Element analizy	Ocena
Menu główne	5
Dodatkowe urządzenie peryferyjne - rękawice	4
Standardowe urządzenie peryferyjne - kontrolery	4
Wykrywanie naturalnych ruchów	5

Następne testowane urządzenie to PlayStation VR. Jego interfejs to przekonwertowany standardowy interfejs PlayStation w wersji wirtualnej, co przekłada się na gorsze odczucie podczas korzystania w wirtualnej rzeczywistości. Pomimo tej konwersji, PlayStation VR zawiera zaprogramowane naturalne ruchy oraz ich wykrywanie. Urządzenie to nie posiada możliwości podłączenia urządzeń peryferyjnych w postaci rękawic, co bardzo źle wpływa na ocenę w tym zakresie. Standardowe kontrolery PlayStation VR są wygodne, mają odpowiednio rozstawione przyciski, lecz muszą być one widoczne przez kamerę PlayStation Eye, co w niektórych momentach ogranicza rozgrywkę, na przykład gdy gracz stoi tyłem do kamery i swoim ciałem zasłania kontrolery. Ocena wyżej opisanych parametrów przedstawiona jest w tabeli 10.

Samsung Gear VR to ostatni z testowanych urządzeń. Interfejs graficzny tego urządzenia, podobnie jak w przypadku wcześniej analizowanych urządzeń, posiada zaprogramowane wykrywanie naturalnych ruchów oraz dodatki dźwiękowe i wizualne pomagające w sprawny sposób zarządzać interfejsem. Samsung Gear VR nie posiada możliwości podłączenia dodatkowych urządzeń innych niż standardowy pad, co skutkuje brakiem oceny tego elementu analizy. Standardowy kontroler dołączony do urządzenia jest mały, ułożenie jego przycisków pozostawia wiele możliwości do poprawy, lecz pomimo tych wad nie jest on uciążliwy w używaniu. Ocena wyżej opisanych parametrów przedstawiona jest w tabeli 11.

Tabela 10: Tabela przedstawiająca ocenę poszczególnych elementów analizy PlayStation VR.

Element analizy	Ocena
Menu główne	4
Dodatkowe urządzenie peryferyjne - rękawice	0
Standardowe urządzenie peryferyjne - kontrolery	4
Wykrywanie naturalnych ruchów	4

Tabela 11: Tabela przedstawiająca ocenę poszczególnych elementów analizy Samsung Gear VR.

Element analizy	Ocena
Menu główne	4
Dodatkowe urządzenie peryferyjne - rękawice	0
Standardowe urządzenie peryferyjne - kontrolery	3
Wykrywanie naturalnych ruchów	4

5. Wnioski

Analiza porównawcza wyżej wymienionych urządzeń pozwoliła na wybranie najbardziej optymalnego urządzenia pod względem zarówno technicznym jak i ergonomicznym oraz pozwoliło na zdefiniowanie które z nich posiada najlepszy według założonych kryteriów interfejs naturalny.

Tabela 12 przedstawia zbiorcze zestawienie ocen parametrów analizy porównawczej parametrów techniczną badanych urządzeń. Na podstawie tej tabeli można wywnioskować, że podczas analizy, najlepszy interfejs posiada urządzenie Oculus Rift S z wynikiem 14 punktów na 15 możliwych co stanowi 93% wszystkich możliwych do zdobycia punktów, natomiast najgorsze parametry techniczne ma HTC Vive Pro, które uzyskało zaledwie 7 punktów.

Tabela 12: Tabela przedstawiająca sumę zdobytych punktów podczas analizy technicznej gogli VR.

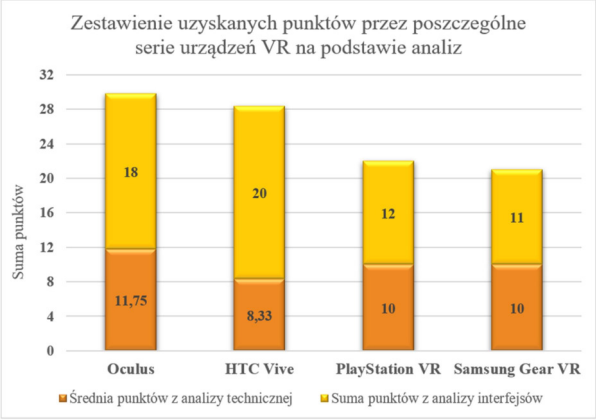
Nazwa poszczególnych urządzeń VR	Waga gogli VR	Cena w dniu premiery gogli VR	Grafika w gogli VR	Suma punktów
Oculus Rift S	5	4	5	14
Oculus Quest	4	4	3	11
Oculus Go	5	5	1	11
Oculus Rift	5	3	3	11
Samsung Gear VR	4	5	1	10
HTC Vive	5	3	2	10
PlayStation VR	3	4	3	10
HTC Vive Cosmos	3	1	4	8
HTC Vive Pro	1	2	4	7

Tabela 13 przedstawia zbiorcze zestawienie ocen parametrów analizy porównawczej interfejsów naturalnych ocenianych urządzeń. Na podstawie tej tabeli można wywnioskować, że podczas analizy naturalnych interfejsów urządzeń, najlepszy interfejs posiada urządzenie HTC Vive z wynikiem 18 punktów na 20 możliwych, natomiast najgorszy interfejs naturalny można spotkać w urządzeniu Samsung Gear VR, które uzyskało zaledwie 7 punktów.

Tabela 13: Tabela przedstawiająca sumy ocen poszczególnych elementów analizy interfejsów naturalnych wszystkich badanych urządzeń.

Element analizy	Liczba punktów zdobyta przez gogle HTC Vive	Liczba punktów zdobyta przez gogle Oculus Rift	Liczba punktów zdobyta przez gogle PlayStation VR	Liczba punktów zdobyta przez gogle Samsung Gear VR
Menu główne	5	5	4	4
Dodatkowe urządzenie peryferyjne - rękawice	5	4	0	0
Standardowe urządzenie peryferyjne - kontrolery	5	4	4	3
Wykrywanie naturalnych ruchów	5	5	4	4
Suma punktów	20	18	12	11

Rysunek 10 przedstawia średnią sumę analiz parametrów technicznych dla poszczególnych serii badanych urządzeń VR (Tabela 12) oraz interfejsów naturalnych. Na jego podstawie wywnioskować można, że najlepsze okazały się urządzenia z serii Oculus, które uzyskały łączną sumę 29.75 punktów na 35 możliwych, natomiast najgorszy wynik uzyskały okulary Samsung Gear VR, otrzymując 21 punktów.



Rysunek 10: Wykres przedstawiający sumę punktów uzyskanych przez poszczególne serie badanych urządzeń VR.

Biorąc pod uwagę przeprowadzoną powyżej analizę wszelkich aspektów urządzeń wirtualnej rzeczywistości najlepszymi goglami okazały się Oculus Rift S ze względu na ich znakomite parametry techniczne jak i także zaimplementowane dla nich interfejsy naturalne.

Z wyników badań wywnioskować można, że nowsze modele osiągały lepsze wyniki niż ich poprzednicy, co spowodowane jest faktem, że wirtualna rzeczywistość jest nową i rozwijającą się technologią. Wywnioskować z tego można również to, że kolejne wersje urządzeń jak i nowe urządzenia będą miały coraz to lepsze osiągi pod względem parametrów jak i naturalności zaimplementowanych interfejsów.

Przeprowadzona analiza wybranych gogli pozwoliła wybrać najlepsze urządzenie pod względem technicznym, najlepsze urządzenie pod względem naturalności interfejsów oraz najlepsze urządzenie sumując uzyskane wyniki, co przełożyło się na osiągnięcie celu pracy.

Literatura

[1] S. M. LaValle, Virtual Reality, Cambridge University Press, 2019.

[2] M. Magnor, A. Sorkine-Hornung, Real VR – Immersive Digital Reality, Springer Nature 13 (2020) 301-306.

[3] Definicja immersji, https://pl.wikipedia.org/wiki/Immersyjno%C5%9B%C4%87 , [14.08.2020].

[4] Statystyki dotyczące liczby użytkowników VR i AR, https://www.emarketer.com/content/virtual-and-augmented-reality-users-2019 , [14.08.2020].

[5] R. Riener, M. Harders, Virtual Reality in Medicine, Springer Science & Business Media, 2012.

[6] A. Lele, Virtual reality and its military utility, Institute for Defence Studies and Analyses 4(1) (2013) 17-26.

[7] Interfejsy graficzne, https://en.wikibooks.org/wiki/A-level_Computing/CIE/Computer_systems,_communications_and_software/System_software/User_interfaces , [30.09.2020].

[8] Interfejs użytkownika, https://pl.wikipedia.org/wiki/Interfejs_u%C5%BCytkownika , [30.09.2020].

[9] Naturalny interfejs użytkownika, https://tylersmcmillan.home.blog/2019/02/18/blog-6/ , [30.09.2020].

[10] Debiut rynkowy HTC Vive https://www.theverge.com/2016/2/21/11081462/htc-vive-consumer-edition-price-release-date-mwc-2016 , [30.10.2020].

[11] Opis parametrów technicznych urządzeń Oculus, http://oculus.com , [30.10.2020].

[12] Opis parametrów technicznych urządzeń PlayStation VR https://www.playstation.com/en-au/explore/playstation-vr , [30.10.2020].

[13] Opis parametrów technicznych urządzeń Samsung Gear VR, https://www.roadtovr.com/samsung-gear-vr-2015-model-now-60/ , [30.10.2020].

[14] B. LeClair, P. O'Connor, S. Podrucky, W. B. Lievers, Measuring the mass and center of gravity of helmet systems for underground workers, International Journal of Industrial Ergonomics 64 (2018) 23-30.

[15] Przegląd i porównanie HTC Vive Pro, https://www.youtube.com/watch?v=jmZjJIyz-xA , [30.10.2020].

[16] Porównanie Oculus RIFT S i HTC Vive Pro, www.youtube.com/watch?v=6NlDaam6izM , [30.10.2020].

JOURNAL OF COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 7–14
Received: 26 October 2020
Accepted: 23 November 2020

Comparative analysis of solutions used in automated testing of Internet applications

Analiza porównawcza rozwiązań wykorzystywanych w testowaniu automatycznym aplikacji internetowych

Magdalena Psujek* , Aleksandra Radzik*, Grzegorz Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This article is devoted to the comparison of solutions used in automatic testing of web applications. In order to carry out the analysis, test scenarios were created and tests were carried out using each of the tested programming frameworks (Selenium WebDriver, Cucumber, Ranorex, Robot Framework, Cypress, Unified Functional Testing, TestComplete and Katalon Studio). The study showed that there is no single best tool that meets all the requirements. Taking into account the analyzed aspects of effectiveness, the TestComplete program was the best, however, when choosing a solution, the team's skills and project specification should be taken into account.

Keywords: automated testing; testing frameworks

Streszczenie
Niniejszy artykuł został poświęcony porównaniu rozwiązań wykorzystywanych w testowaniu automatycznym aplikacji internetowych. W celu przeprowadzenia analizy stworzono scenariusze testowe oraz przeprowadzono testy z użyciem każdego z badanych narzędzi (Selenium WebDriver, Cucumber, Ranorex, Robot Framework, Cypress, Unified Functional Testing, TestComplete oraz Katalon Studio). W pracy wykazano, że nie istnieje jedno najlepsze rozwiązanie, spełniające wszystkie wymagania. Biorąc pod uwagę analizowane aspekty efektywności program TestComplete wypadł najlepiej, jednak przy wyborze rozwiązania należy wziąć pod uwagę umiejętności zespołu oraz specyfikację projektu.

Słowa kluczowe: testowanie automatyczne; frameworki testujące

*Corresponding author
Email addresses: m.e.psujek@gmail.com (M. Psujek), aradzik96@gmail.com (A. Radzik)

1. Wstęp

Oprogramowanie zawierające błędy nie powinno być wprowadzane do użytku komercyjnego. Nieprzetestowany produkt może spowodować straty finansowe a nawet być przyczyną choroby lub śmierci wielu osób [1].

W niniejszej pracy porównywane zostały narzędzia służące do tworzenia testów automatycznych. Takie testy wykonywane są gdy konieczne jest wielokrotne powtarzanie danego przypadku lub podczas symulowania dużego obciążenia. Automatyzacji testów najczęściej poddawane są systemy, które nie ulegają zmianom lub takie, w których wprowadzane zmiany są niewielkie i nie występują często. Testy automatyczne dają wyniki znacznie szybciej niż testy manualne, które przeprowadzone muszą być ręcznie przez pracownika [2]. W przeciwieństwie do maszyny, człowiek może popełnić błędy przy wprowadzaniu danych lub wykonać niepoprawne operacje systemowe [3]. Test przeprowadzany manualnie za każdym razem będzie dawał inny wynik chociażby, ze względu na czas jego wykonania. Zaletą testów automatycznych jest to, że wykonywane są zawsze z tą samą precyzją.

2. Analiza literatury

Powstało wiele dzieł podejmujących temat testowania automatycznego [4], których podejście zostanie przedstawione w niniejszym rozdziale. Na początku jednak należy wyjaśnić czym jest testowanie oprogramowania w jakim celu się stosuje oraz jakie są jego rodzaje.

Zgodnie z “Sylabusem poziomu podstawowego ISTQB®” [5] testowanie oprogramowania daje możliwość oceny jego jakości a także umożliwia zmniejszenie ryzyka wystąpienia niechcianych incydentów czy awarii w trakcie użytkowania systemu. W zależności od poziomu testu celem może być zlokalizowanie jak największej liczby defektów lub sprawdzenie czy system spełnia oczekiwania i został stworzony zgodnie z wymaganiami.

Testowanie oprogramowania można podzielić na dwie główne kategorie: manualne oraz automatyczne [6]. W artykule “Introduction to Software Testing” [7] autorstwa Durgesh Raghuvanshi, poruszony został temat testowania automatycznego. Zauważono w nim, że testowanie oprogramowania pochłania 50% kosztów jego wytwarzania. Zwrócono uwagę, że wprowadzenie automatyzacji pozwala na zmniejszenie zapotrzebowania na zasoby ludzkie, zwiększa dokładność, co za tym idzie zmniejsza ryzyko popełnienia błędu w trakcie przeprowadzania testu oraz znacznie skraca czas potrzebny na wykonanie testu.

Niniejsza praca traktuje o testach automatycznych funkcjonalnych. Testy funkcjonalne to proces, w ramach którego QA Specialist (Quality Assurance

Specialist- ang. Specjalista ds. Zapewniania Jakości) określa, czy oprogramowanie działa zgodnie z wcześniej ustalonymi wymaganiami. Wykorzystuje techniki testowania czarnej skrzynki, w których tester nie ma wiedzy o wewnętrznej logice systemu. Testy funkcjonalne dotyczą wyłącznie sprawdzania poprawności, czy system działa zgodnie z przeznaczeniem [5]. Niepowodzenie przeprowadzenia testu (uzyskanie niepoprawnego wyniku) może być spowodowane błędami w implementacji kontrolera czy tez modelu. Dlatego też analizowanie rodzajów błędów jakie są w stanie wykryć testy funkcjonalne mija się z celem. Tu sprawdza się czy wszystkie elementy systemu jako całość poprawnie współpracują i pozwalają na uzyskanie właściwego efektu działania aplikacji obsługiwanej przez interfejs graficzny dostępny przez przeglądarkę. Termin testy funkcjonalne występuje również w "Sylabusie poziomu podstawowego ISTQB®" [8], w którym wyjaśniono, że testy funkcjonalne to proces uwzględniający zachowanie systemu. W sylabusie przedstawiono zalety i ryzyka automatyzacji testów. Ze względu na to, że temat zalet został już poruszony w poprzednich akapitach rozważone zostaną zagrożenia takiego rozwiązania. Zgodnie z materiałem źródłowym należą do nich:

- oczekiwanie, że wybrane narzędzie będzie proste w obsłudze oraz spełni nierealistyczne wymagania;
- złe oszacowanie liczby godzin i kosztów wdrożenia narzędzia;
- złe oszacowanie liczby koniecznych działań, które miałoby zapewnić trwałe i znaczące korzyści z automatyzacji;
- całkowite zastąpienie testów manualnych automatycznymi;
- wycofanie narzędzia z rynku lub sprzedaż innemu dostawcy;
- brak współpracy z nowymi technologiami.

W zasobach portalu www.researchgate.net znaleziono artykuł "A Study of Automated Software Testing:Automation Tools and Frameworks" [9] autorstwa Mubarak Albarka Umar oraz Chen Zhanfang. Autorzy porównują w nim cztery narzędzia analizowane również w niniejszej pracy. Należą do nich: Katalon Studio, Unified Functional Testing, Selenium oraz TestComplete. Wymieniono wady i zalety każdego z rozwiązań. Jednak autorzy zaznaczają, że nie ma dobrego lub złego narzędzia do automatyzacji. Każde ma inne przeznaczenie i wymaga innego nakładu pracy. Wybór odpowiedniego narzędzia powinien zależeć od charakteru oprogramowania.

Następnym artykułem traktującym o testowaniu automatycznym jest "Analiza porównawcza rozwiązań wykorzystywanych w testowaniu automatycznym" [10] autorstwa: Agnieszka Dorota Wac, Tomasz Kamil Watras, Grzegorz Kozieł. W pracy zostały porównane SeleniumIDE, TestComplete, SahiPro oraz Katalon Studio. Twórcy analizowani wybrane narzędzia między innymi pod kątem procesu tworzenia testów oraz prędkości ich wykonywania. Na podstawie przeprowadzonych badań nie wybrano najlepszego i uniwersalnego programu służącego do wykonywania automatycznych testów.

Szybkość wykonywania testów automatycznych jest jednym z kluczowych aspektów testowania. Testy automatyczne są tworzone aby tester w dowolnym momencie pracy mógł w szybki sposób zweryfikować poprawność swojej pracy i znaleźć błędy. Źródła wskazują, że testowanie powinno się odbywać jak najwcześniej w celu zredukowania kosztów naprawy błędów [11]. Wiąże się to z tym, że programista powinien testować kod wielokrotnie podczas jego tworzenia. Czas wykonania testów jest szczególnie istotny w przypadku testów regresyjnych, gdzie badany jest wpływ wprowadzonych zmian na wiele modułów kodu - tu szczególnie istotne jest szybkie wykonywanie testów pozwalające na uzyskanie wyników w rozsądnym czasie [12]. Ze względu na przytoczone argumenty w artykule skupiono się na aspektach czasowych wykonania testów.

3. Badane rozwiązania

Technologie internetowe, tak jak i inne dziedziny informatyki, rozwijają się w bardzo szybkim tempie [13]. Z tego względu zwiększa się konieczność tworzenia coraz lepszych i wydajniejszych sposobów ich testowania. Obecnie udostępniono wiele narzędzi służących do tworzenia automatycznych testów funkcjonalnych. W niniejszej pracy przebadano jedne z najpopularniejszych używanych przez testerów. Należą do nich: Selenium WebDriver, Cucumber, Ranorex, Robot Framework, Cypress, Unified Functional Testing (UFT), TestComplete oraz Katalon Studio[14-15].

4. Plan badań

Aby uzyskać jak najbardziej rzetelne porównanie, zdecydowano się na przeprowadzenie badań w formie testów z użyciem każdego z narzędzi wymienionych w punkcie trzecim niniejszego artykułu.

4.1. Metoda badań

Przygotowane zostały cztery scenariusze testowe. Po dwa dla każdej z testowanych stron. Scenariusze miały różną długość, aby sprawdzić które narzędzie lepiej radzi sobie z danym rodzajem testu. Dodatkowo sprawdzono czy testy wykrywają awarię na stronie. Awaria była symulowana na stworzonej na potrzeby tej pracy platformie. Za pomocą każdego z narzędzi został przeprowadzony ten sam zestaw testów w takich samych warunkach, z użyciem tej samej przeglądarki, platformy, dodatkowego oprogramowania, komputera.

4.2. Strona Moodle Pollub

Dla strony moodle1.pollub.pl utworzone zostały dwa scenariusze. Jednym z nich jest scenariusz zmieniający język na stronie i sprawdzający czy zmiana przebiegła poprawnie. Drugi test sprawdza czy wiadomości wysyłane do innych użytkowników są przez nich odbierane, następnie czy wiadomość jest usuwana. Test zakończony zostaje pozytywnie jeśli każda z tych czynności zostanie wykonana a użytkownik po dokonaniu zmian wylogowany.

4.3. Własna strona

Strona stworzona na potrzeby tej pracy to prosty sklep internetowy. Umożliwia rejestrację i logowanie użytkowników, dodanie do koszyka oraz zakup produktów po wcześniejszym zalogowaniu. Po zebraniu odpowiedniej liczby wyników testów, na stronie zasymulowano awarię.

5. Scenariusze testowe

Przygotowano cztery zestawy testowe:

- scenariusz testowy nr 1: Test zmiany języka Moodle Pollub, przypadek testowy 1.1: Test zmiany języka Moodle Pollub na język angielski,
- scenariusz testowy nr 2: Wysłanie wiadomości Moodle Pollub, przypadek testowy 2.1: Wysłanie wiadomości Moodle Pollub z konta o loginie 83821, na konto o loginie 83818,
- scenariusz testowy nr 3: Przegląd dostępnych produktów, przypadek testowy 3.1: Sprawdzenie dostępności wybranego produktu,
- scenariusz testowy nr 4: Sprawdzenie funkcjonalności koszyka, przypadek testowy 4.1: Dodawanie i usuwanie produktów z koszyka.

Przykładowo przedstawiono jeden z nich. Scenariusz przedstawiony w tabeli 1 oraz przypadek testowy przedstawiony w tabeli 2 zostały wykorzystane dwukrotnie: w pierwszej kolejności na sprawnej stronie oraz podczas symulacji awarii, która polegała na dezaktywowaniu przycisku w menu strony.

Tabela 1: Scenariusz testowy nr 4.

Nazwa	Sprawdzenie funkcjonalności koszyka
Cel	Celem przeprowadzenia scenariusza testowego jest weryfikacja funkcjonalności dotyczącej działania koszyka: dodawanie i usuwanie produktu a także rejestracja nowego użytkownika oraz usunięcie konta
Typ	Test funkcjonalny
Czynności przygotowawcze	Sprawdzenie posiadania przeglądarki internetowej, sprawdzenie posiadania aktualnej aplikacji. Uruchomienie serwera bazy danych
Czynności końcowe	Wyłączenie przeglądarki. Zapis wyników do bazy danych

Tabela 2: Przypadek testowy 4.1

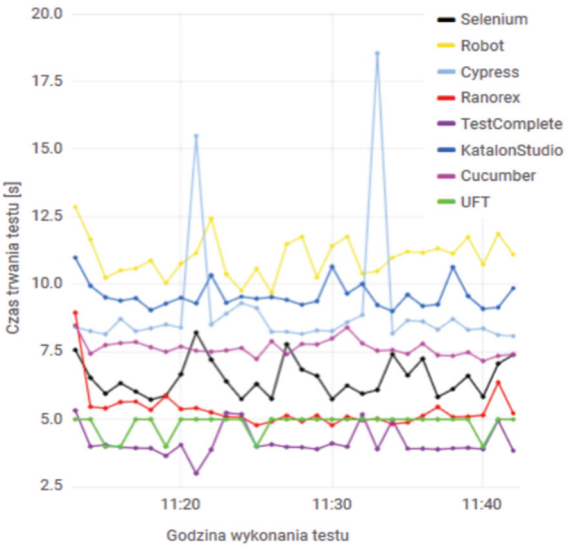
Nazwa	Dodawanie i usuwanie produktów z koszyka
Środowisko	Przeglądarka: Google Chrome Wersja 83.0.4103.116
Warunek wstępny	Włączona przeglądarka
Kroki	1. Otwórz stronę sklepu umieszczonego na localhost 2. Kliknij zarejestruj się 3. Wpisz imię “Test” 4. Wpisz nazwisko “Test” 5. Wpisz login “test1” 6. Wpisz miasto “Lublin” 7. Wpisz kod pocztowy „20-240” 8. Wpisz email “test@test.pl” 9. Wpisz hasło “testtest1” 10. Zaloguj się do sklepu internetowego 11. Przejdź do zakładki produkty 12. Wyszukaj produkt “Berberys” 13. Dodaj produkt do koszyka 14. Usuń produkt z koszyka 15. Kliknij wyświetl profil 16. Usuń konto
Oczekiwany rezultat	Poprawne dodanie do koszyka przez nowo zarejestrowanego klienta
Warunki końcowe	Produkt usunięty z koszyka, brak użytkownika w bazie

6. Rezultaty testów

W tym rozdziale przedstawione zostaną rezultaty przeprowadzonych testów. Dla każdego przypadku oraz narzędzia zrealizowano 30 testów. Przyjęto, że ze względu na powtarzalność czasów taka liczba powtórzeń wystarczy aby dać rzetelny średni wynik dla poszczególnych rozwiązań.

6.1. Test zmiany języka strony

Test zmiany języka strony został stworzony na podstawie scenariusza testowego nr 1 przypadku testowego 1.1 o nazwie "Test zmiany języka Moodle Pollub na język angielski".



Rysunek 1: Wykres czasu trwania wszystkich testów, dla przypadku testowego 1.1 .

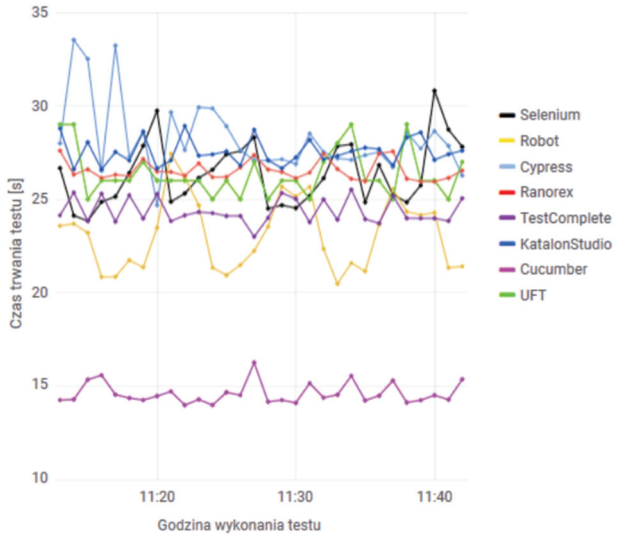
Rysunek 1 przedstawia wykres zbiorczy czasu trwania wszystkich testów, przeprowadzonych na podstawie przypadku 1.1 dla wykorzystanych narzędzi do testowania automatycznego. Z przedstawionego wykresu można odczytać, że najlepiej wypadło narzędzie TestComplete uzyskując najniższe czasy wykonania pojedynczego testu. Największy czas trwania odnotowano w przypadku framewoka Robot. Na wykresie można zauważyć również, że narzędzie Cypress posiada dwa duże odchyły od normy spowodowane przez niepoprawny wynik testu. Średni oraz całkowity czas trwania testów przedstawiono w tabeli 3.

Tabela 3: Wykaz najszybszych narzędzi.

Pozycja	Nazwa	Średni czas (s)	Całkowity czas (s)
1	TestComplete	4.15	124.8
2	UFT	4.83	145.2
3	Ranorex	5.35	160.8
4	Selenium	6.53	195.6
5	Cucumber	7.64	229.2
6	Cypress	9.03	271.2
7	Katalon	9.58	297.0
8	Robot	11.01	330.0

6.2. Test wysłania wiadomości strony

Test wysłania wiadomości dotyczy przypadku testowego 2.1 o nazwie "Wysłanie wiadomości Moodle Pollub z konta o loginie 83821, na konto o loginie 83818".



Rysunek 2: Wykres czasu trwania wszystkich testów, dla przypadku testowego 2.1.

Na rysunku 2 przedstawiony został zbiorczy wykres z wynikami długości trwania testów, przeprowadzonych na podstawie przypadku 2.1, każdego z narzędzi. Na wykresie od razu można dostrzec jeden wyróżniający się framework. Jest to Cucumber, uzyskał on najniższe czasy spośród wszystkich testowanych narzędzi. Jako drugie i trzecie - Robot Framework oraz TestComplete. Narzędziem, które najwolniej przeprowadziło test okazał się Cypress. Średni oraz całkowity czas trwania testów przedstawiono w tabeli 4.

Tabela 4: Wykaz najszybszych narzędzi.

Pozycja	Nazwa	Średni czas (s)	Całkowity czas (s)
1	Cucumber	14.61	438.6
2	Robot	23.11	693.6
3	TestComplete	24.36	730.8
4	Selenium	26.35	790.8
5	UFT	26.37	790.8
6	Ranorex	26.54	796.2
7	Katalon Studio	27.54	826.2
8	Cypress	28.24	847.2

6.3. Test sprawdzenia dostępności produktu

Kolejnym przypadkiem testowym na podstawie którego, analizowano czasy działania różnych narzędzi jest przypadek 3.1 o nazwie "Sprawdzenie dostępności wybranego produktu". Rysunek 3 przestawia wykres czasu trwania wszystkich testów, przeprowadzonych na podstawie przypadku 3.1, dla każdego z narzędzi.

Rysunek 3: Wykres czasu trwania wszystkich testów, dla przypadku testowego 3.1.

Widocznym jest, iż narzędzie Selenium uzyskało najlepszy czasu trwania testu. Zbliżone czasy do najszybszego z narzędzi odnotowano dla programów TestComplete oraz Cypress. Programem, który wykonał testy najwolniej okazał się Katalon. Średni oraz całkowity czas trwania testów przedstawiono w tabeli 5.

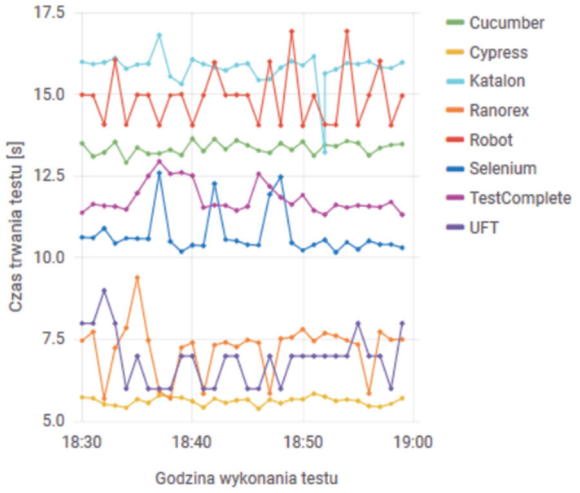
Tabela 5: Wykaz najszybszych narzędzi.

Pozycja	Nazwa	Średni czas (s)	Całkowity czas (s)
1	Selenium	1.7	49.8
2	TestComplete	1.7	51.0
3	Cypress	1.8	55,2
4	UFT	3.1	94.2
5	Ranorex	3.4	103.2
6	Cucumber	5.8	174.0
7	Robot	6.1	180.0
8	Katalon	7.4	222.0

6.4. Test funkcjonalności koszyka

Ostatnim przygotowanym przypadkiem testowym jest przypadek 4.1 o nazwie "Dodawanie i usuwanie produktów z koszyka". Przypadek ten przetestowano po trzydzieści razy dla każdego narzędzia.

Wyniki przeprowadzonych testów przedstawione zostały zaprezentowanie na rysunku 4. Średni oraz całkowity czas trwania testów przedstawiono w tabeli 6.



Rysunek 4: Wykres czasu trwania wszystkich testów, dla przypadku testowego 4.1.

Tabela 6: Wykaz najszybszych narzędzi.

Pozycja	Nazwa	Średni czas (s)	Całkowity czas (s)
1	Cypress	5.6	168.0
2	UFT	6.9	210.0
3	Ranorex	7.2	216,0
4	Selenium	11.0	324.0
5	TestComplete	11.8	354.0
6	Cucumber	13.0	402.0
7	Robot	15.0	450.0
8	Katalon	16.0	492.0

6.5. Symulacja awarii

Jako ostatni etap testowania narzędzi zasymulowano awarię poprzez wyłączenie funkcjonalności jednego z odnośnika na pasku menu. W ten sposób sprawdzono jak szybko narzędzia radzą sobie z wykryciem niepoprawności zaistniałej na stronie. Rysunek 5 przedstawia czasy trwania wszystkich testów. Najgorzej w tej próbie wypadł program Ranorex, każdy z testów wynosił 60s, jest to związane z tym, że maksymalny czas szukania elementu na stronie wynosi minutę, jest to czas ustawiony systemowo.

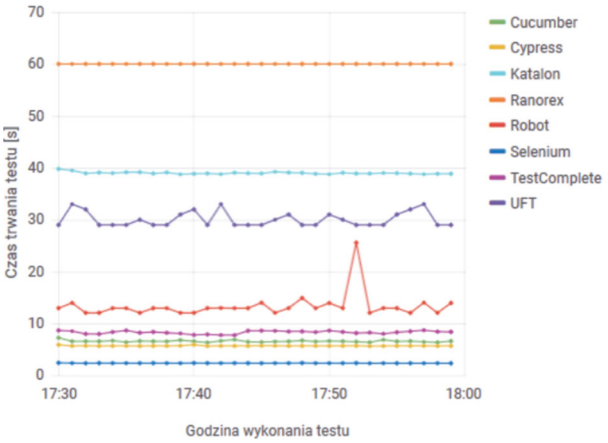
Średni oraz całkowity czas trwania testów przedstawiono w tabeli 7.

Tabela 7: Wykaz najszybszych narzędzi.

Pozycja	Nazwa	Średni czas (s)	Całkowity czas (min)
1	Selenium	2.4	70.8
2	Cypress	5.7	174.0
3	Cucumber	6.6	198.0
4	TestComplete	8.3	252.0
5	Robot	14.0	426.0
6	UFT	30.0	900.0
7	Katalon	39.0	1170.0
8	Ranorex	60.0	1800.0

7. Porównanie narzędzi

Tematem pracy było porównanie efektywności rozwiązań wykorzystywanych w testowaniu automatycznym aplikacji internetowych. Wybrano następujące kategorie, w których postanowiono przeanalizować rozwiązania zawarte w pracy: dostępność dokumentacji i wsparcie społeczności, poziom trudności konfiguracji oraz obsługi narzędzia, szybkość wykonania testu oraz prezentacja wyników.



Rysunek 5: Wykres czasu trwania wszystkich testów w przypadku awarii.

7.1. Dostępność dokumentacji

Ważnym elementem w trakcie korzystania z danego narzędzia jest dobrze i zrozumiale przygotowana dokumentacja. Postanowiono przeanalizować dokumentację rozwiązań pod kątem:

1. Dostępności (0-1),
2. Przejrzystości (0-2),
3. Łatwości w kontakcie i możliwości wsparcia społeczności tworzącej i korzystającej z danego narzędzia (0-4),
4. Częstotliwości aktualizacji (0-2),
5. Łatwości wyszukiwania (0-2).

Aby wybrać najlepsze narzędzie pod wskazanym aspektem postanowiono nadać odpowiednio punkty, przedstawione w nawiasach, dotyczące danej kategorii, ocena 0 oznacza brak, zaś najwyższa- możliwe najlepsze rozwiązanie w danej kategorii (tabela 8). Punkty przyznane w tabeli 8 zostały przydzielone na podstawie subiektywnych opinii autorów pracy dotyczących prezentacji i łatwości w korzystaniu z dokumentacji, ilości wątków w serwisie https://stackoverflow.com/, wprowadzanych zmian dotyczących konkretnego narzędzia w serwisie https://github.com/.

Tabela 8: Dostępność dokumentacji oraz wsparcie dostępności.

Nazwa	Numer kategorii					Suma
	1	2	3	4	5	
Cucumber	1	2	3	2	2	10
Cypress	1	2	2	2	2	9
Katalon	1	2	1	2	2	8
Ranorex	1	1	0	2	1	5
Robot Framework	1	1	3	2	1	8
Selenium	1	2	4	2	2	11
TestComplete	1	2	0	2	2	7
UFT	1	1	2	0	2	6

Na podstawie powstałej tabeli, można zauważyć, że większość z firm produkujących narzędzia służące do testowania automatycznego dba o dobrze dostępną oraz zrozumiałą dokumentację. Udostępnia również wsparcie poprzez utworzone grupy oraz fora internetowe.

7.2. Poziom trudności konfiguracji oraz obsługi narzędzia

Na podstawie rozdziału szóstego niniejszej pracy postanowiono podsumować poziom skomplikowania każdego z narzędzi.

Tabela 9: Charakterystyka poszczególnych narzędzi.

Nazwa	Instalacja	Dodatkowe oprogramowanie	Interfejs
Cucumber	dołączenie biblioteki	kompilator wybranego języka	brak
Cypress	instalator	nie jest konieczne	prosty
Katalon	instalator	nie jest konieczne	zaawansowany
Ranorex	instalator	nie jest konieczne	zaawansowany
Robot Framework	instalacja z poziomu konsoli lub instalacja z kodu źródłowego	interpreter	brak
Selenium	dołączenie biblioteki	kompilator wybranego języka	brak
TestComplete	instalator	nie jest konieczne	zaawansowany
UFT	instalator	przydatne ale niekonieczne	zaawansowany

Tabela 10: Charakterystyka poszczególnych narzędzi.

Nazwa	Możliwość personalizacji	Wymagane umiejętności	Sposób tworzenia testów
Cucumber	wysoka	dobra znajomość Java, C#, Ruby, JavaScript, R lub Python	skrypt
Cypress	średnia	podstawowa znajomość programowania	skrypt
Katalon	wysoka	pomocna znajomość Groovy	skrypt, wykonanie nagrywania testu, ręczne wybieranie
Ranorex	wysoka	pomocna znajomość C#	wykonanie nagrywania testu, ręczne wybieranie
Robot Framework	wysoka	znajomość podstaw języka Python	skrypt
Selenium	wysoka	dobra znajomość Java, C#, Ruby, JavaScript, R lub Python	skrypt
TestComplete	wysoka	pomocna znajomość JavaScript, JScript, Python, VBScript, DelphiScript, C#Script, C++Script	wykonanie nagrywania testu, ręczne wybieranie
UFT	wysoka	pomocna znajomość VBS	wykonanie nagrywania testu manualnego lub skrypt

Każde z rozwiązań oceniono w zależności od sposobu instalacji, wymaganego dodatkowego oprogramowania, dostępności i zaawansowania interfejsu, możliwości personalizacji narzędzia, wymaganej wiedzy przed rozpoczęciem tworzenia testów oraz sposobu tworzenia testów. Charakterystykę poszczególnych narzędzi przedstawiono w tabeli 9. oraz 10.

Tabela 11. zawiera ocenę w skali 1-3 odpowiednio od najłatwiejszego do najtrudniejszego rozwiązania w każdej z kategorii. Najłatwiejsze rozwiązanie to takie, które uzyskało najmniejszą sumę punktów. Następującym numerom w tabeli 11 przyporządkowano kategorie: 1- Instalacja, 2- Dodatkowe oprogramowanie, 3- Interfejs, 4- Możliwość personalizacji, 5- Wymagane umiejętności, 6- Sposób tworzenia testów.

Tabela 11: Trudności konfiguracji oraz obsługi narzędzia.

Nazwa	Numer kategorii					
	1	2	3	4	5	6
Cucumber	3	2	3	1	3	3
Cypress	1	1	1	2	2	3
Katalon	1	1	2	1	1	1
Ranorex	1	1	2	1	1	1
Robot Framework	2	2	3	1	2	1
Selenium	3	2	3	1	3	3
TestComplete	1	1	2	1	1	1
UFT	1	2	2	1	1	1

7.3. Szybkość wykonania testu

Aby określić który z wybranych frameworków najlepiej wypadł w testach biorąc pod uwagę szybkość wykonywania testu postanowiono nadać odpowiednio punkty w zależności od zajętego miejsca w tabelach od 5 do 9, przy czym najniższa liczba punktów ustanawia narzędzie najlepszym.

Tabela 12: Ocena narzędzi w zależności od szybkości wykonania testu.

Nazwa	Pozycja w tabelach dotyczących szybkości o przypadkach nr					Suma
	1.1	2.1	3.1	4.1	4.1-Awaria	
Selenium	4	4	1	4	1	14
Cucumber	5	1	6	6	3	21
Robot	8	2	7	7	5	29
Cypress	6	8	3	1	2	20
UFT	2	5	4	2	6	19
Ranorex	3	6	5	3	8	25
TestComplete	1	3	2	5	4	15
Katalon	7	7	8	8	7	37

W tabeli 12. zebrano punkty jakie otrzymały narzędzia analizowane w niniejszej pracy.

Najlepszym narzędziem do testowania automatycznego pod względem szybkości okazało się Selenium, bardzo zbliżony wynik, bo różniący się tylko o jeden punkt uzyskał program TestComplete. Najwolniej testy wykonywało narzędzie Katalon Studio. Pozostałe narzędzia zebrały podobną liczbę punktów.

7.4. Prezentacja wyników

W celu analizy sporządzonych testów, większość z narzędzi tworzy raporty zawierające między innymi czas trwania testu i informacje o powodzeniu próby. Tabela 13. przedstawia zbiór informacji dotyczących raportowania w wykorzystywanych narzędziach.

Tabela 13: Ocena narzędzi w zależności od sposobu prezentacji wyników.

Nazwa	Typ plików generowanych po zakończeniu testu	Czy automatycznie generowany jest raport?
Cucumber	json, html, JUnit	nie
Cypress	raport jest dołączany do projektu, brak możliwości otworzenia go bez użycia programu, możliwość eks-portu do json, html	tak
Katalon	raport jest dołączany do projektu, brak możliwości otworzenia go bez użycia programu, możliwość eks-portu do PDF, CSV, html lub JUnit	tak
Ranorex	rxlog- specjalny plik generowany przez program Ranorex (brak możliwości otworzenie pliku bez posiadania aplikacji Ranorex)	tak
Robot Framewrok	html, xml, jpg (zrzuty ekranu w wypadku wyniku negatywnego)	tak
Selenium	narzędzie samo w sobie nie generuje plików	nie
TestComplete	raport jest dołączany do projektu, brak możliwości otworzenia go bez użycia programu, możliwość eksportu do JUnit reports, MHT, HTML, XML, PDF, tcLogX	tak
UFT	raport jest dołączany do projektu, brak możliwości otworzenia go bez użycia programu. Możliwość eksportu do html	tak

Aby ocenić i wybrać najlepsze narzędzie postanowiono przyznać odpowiednio przyznać punkty. Sposób oceniania:

- zero punktów za brak generowania raportu,
- dwa punkty w przypadku generowania jednego typu pliku zawierającego raport,
- przyznanie trzech punktów jeśli generowane są minimum dwa typy plików zawierających raport,
- odjęcie jednego punktu za brak możliwości otworzenia raportu bez użycia narzędzia,

dodatkowo:

- przyznanie trzech punktów za automatyczne generowanie raportu.

Po zsumowaniu punktów otrzymujemy następujące zestawienie przedstawione w tabeli 14. W tym przypadku im więcej punktów zdobyło narzędzie tym okazuje się lepszym rozwiązaniem. Najlepszym narzędziem w tym wypadku okazał się Robot Framework

Tabela 14: Ocena podsumowująca narzędzi w zależności od sposobu prezentacji wyników.

Nazwa	Suma punktów
Cucumber	4
Cypress	6
Katalon	6
Ranorex	4
Robot Framework	7
Selenium	0
TestComplete	6
UFT	4

7.5. Podsumowanie

Wybrane rozwiązania przetestowano oraz przeanalizowano pod różnym kątem. W podrozdziałach 7.1-7.4 zostały przedstawione różne aspekty dotyczące efektywności oraz oceniono każde z narzędzi. W niniejszym podrozdziale postanowiono nadać punkty na podstawie miejsca jakie zdobyły w wybranych kategoriach w celu przekonania się o tym, które z narzędzi jest najlepsze (tabela 15.). Numery porządkowe w tabeli 15 odpowiadają kolejno: 1- dokumentacja, 2- konfiguracja, 3- szybkość, 4- raporty.

Tabela 15: Podsumowanie wszystkich analizowanych aspektów.

Nazwa	Numer porządkowy				Suma
	1	2	3	4	
Cucumber	2	5	5	3	15
Cypress	3	3	4	2	12
Katalon	4	1	8	2	15
Ranorex	7	1	6	3	17
Robot Framework	4	4	7	1	16
Selenium	1	5	1	4	11
TestComplete	5	1	2	2	10
UFT	6	2	3	3	14

8. Wnioski

Celem pracy było porównanie wybranych rozwiązań wykorzystywanych w testowaniu automatycznym aplikacji internetowych. Przeanalizowano ogólnodostępną literaturę w tym artykuły udostępnione w czasopismach naukowych. Dostarczone informacje nie pozwoliły jednak na jednoznaczne określenie, który z programów do testowania jest najlepszym rozwiązaniem.

Podsumowując badania przedstawione w tabelach, najlepszym narzędziem biorąc pod uwagę czas wykonania testu okazało się Selenium. Zbliżony wynik uzyskał program TestComplete, jako trzecie narzędzie uzyskujące najlepszą liczbę punktów okazało się UFT. Programy TestComplete oraz UFT są to aplikacje płatne, o zamkniętym oprogramowaniu w przeciwieństwie do Selenium. Dzięki temu można stwierdzić, że nie tylko płatne rozwiązania są dobrym wyborem.

Kolejnym krokiem przybliżającym do uzyskania celu niniejszej pracy było porównanie wybranych narzędzi pod względem dostępności dokumentacji oraz wsparcia społeczności, poziomu trudności konfiguracji środowisk, generowanych raportów z testów oraz szybkości wykonywania testów. W każdym z wymienionych aspektów oceniono narzędzia, nadając im subiektywne noty, oraz obiektywne w przypadku oceniania szybkości wykonania testu. Jako najlepsze narzędzie uznano program TestComplete, drugim w kolejności Selenium zaś trzecim program Cypress. Płatne narzędzie okazało się być najwygodniejszym i najłatwiejszym w obsłudze, jednak rozwiązania darmowe również nie odbiegają znacząco od rozwiązań z kosztowną licencją.

Podczas wyboru narzędzia do testowania automatycznego aplikacji internetowych należy zastanowić się jakie funkcjonalności są najbardziej pożądane. Nie należy kierować się zawsze ścisłymi liczbami oraz rankingami, ponieważ istnieje prawdopodobieństwo narażenia się na zakup narzędzia, które nie jest potrzebne. Warto wziąć również pod uwagę aspekt ludzki, czy osoba używająca narzędzia będzie potrafiła go w pełni obsłużyć.

Literatura

[1] R. Dahiya, A. Shahid, Importance of manual and automation testing 2019.

[2] P. Kunte, D. Mane, Automation Testing of Web based application with Selenium and HP UFT (QTP). International Research Journal of Engineering and Technology 6 (2017) 2579-2583.

[3] Automatyzacja testów musi być przemyślana, https://www.computerworld.pl/news/Automatyzacja-testow-musi-byc-przemyslana,381609.html, [17.03.2020].

[4] Stowarzyszenie Jakości Systemów Informatycznych, Certyfikowany tester, Sylabus poziomu podstawowego certyfikatu ISTQB 2018, Wersja 1.0.1. 2019.

[5] H. V. Gamido, M. V. Gamido, Comparative review of the features of automated software testing tools, International Journal of Electrical and Computer Engineering (IJECE) 9(5) (2019) 4473.

[6] A. S. Gadwal, L. Prasad, Comparative review of the literature of automated testing tools 2020.

[7] D. Raghuvanshi, Introduction to Software Testing, International Journal of Trend in Scientific Research and Development (IJTSRD) 2020.

[8] D. Chelimsky, D. Astels, Z. Dennis, A. Hellesøy, B. Helmkamp, D. North, The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber and Friends, The Pragmatic Bookshelf 2012.

[9] M. A. Umar, C. Zhanfang, A Study of Automated Software Testing: Automation Tools and Frameworks. International Journal of Computer Science Engineering 6 (2019) 217-225.

[10] A. D. Wac, T. K. Watras, G. Kozieł, Analiza porównawcza rozwiązań wykorzystywanych w testowaniu automatycznym. Journal of Computer Sciences Institute 15 (2020) 156-163.

[11] H. Q. Jaleel, Testing Web Applications, SSRG International Journal of Computer Science and Engineering 6(12) (2019) 1-9.

[12] Koszty i wartość automatyzacji. https://testerzy.pl/baza-wiedzy/koszty-i-wartosc-automatyzacji, [20.06.2020].

[13] P. Marek, Weryfikacja i automatyzacja procesu testowania oprogramowania. C0RE Magazine 2011.

[14] 10 najlepszych narzędzi automatyzacji testów. https://testerzy.pl/narzedzia/10-najlepszych-narzedzi-automatyzacji-testow, [20.06.2020].

[15] Top 10 Automation Testing Tools in 2020. https://www.netsolutions.com/insights/top-10-automation-testing-tools/, [20.06.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 15–21
Received: 19 November 2020
Accepted: 08 December 2020

Evaluation of the Kinect controller precision

Ocena precyzji działania kontrolera Kinect

Piotr Mieszawski*, Tomasz Szymczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The subject of this work is to evaluate the precision of the Kinect controller operation. Research study was performed and a measuring stand have been prepared. Then an application was created that captures the user's throw gesture and simulates the flight of a virtual ball. Based on this, measurements were made to determine the difference between hand movement and it's detection by the application, and differences among throw made in real life and in VR. The analysis of these results allowed the accuracy of the controller to be assessed.
Keywords: comparative analysis; Kinect; virtual reality

Streszczenie
Tematem tej pracy jest ocena precyzji działania kontrolera Kinect. Na potrzeby tego został opracowana metodyka badawcza oraz przygotowane stanowisko pomiarowe. Następnie została stworzona aplikacja, która wychwytuje gest rzutu użytkownika oraz symuluje lot wirtualnej piłki. Na podstawie tego wykonano pomiary, polegające na określeniu różnicy kątów pomiędzy tym jak poruszała się dłoń, a jak to zostało wykryte przez aplikacje oraz określeniu różnicy pomiędzy odległością rzutu wykonanego w rzeczywistości a w VR (ang. Virtual Reality). Analiza tych wyników pozwoliła określić precyzje kontrolera.
Słowa kluczowe: Kinect; wirtualna rzeczywistość; analiza porównawcza

*Corresponding author
Email address: **piotr.mieszawski@pollub.edu.pl**

1. Wstęp

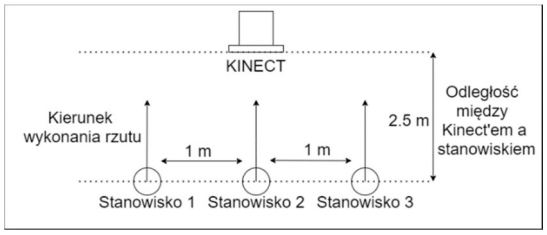
Wirtualna rzeczywistość aktualnie jest znana jako sztuczny, trójwymiarowy obraz wytworzony przy wykorzystaniu najnowszej technologii. Może zawierać w sobie różnego rodzaju przedmioty, obiekty a nawet całe zdarzenia. Może się opierać zarówno na elementach świata fikcyjnego jak i realnego.

Założeniami tej technologii jest wywołanie u ludzi prawdziwych i naturalnych odczuć. Nie jest to łatwe, ponieważ ludzki mózg jest tak skonstruowany, że na podstawie najmniejszych detali może wyczuć, że widziany obraz nie jest prawdziwy. Dlatego aby stworzyć świat wirtualny jak najbardziej przypominający realny trzeba zrozumieć fizjonomie człowieka. Każdy element jest poznawany za pomocą zmysłów. Z tego powodu trzeba było wykorzystać możliwości, które pozwoliłyby je oszukać. Jednym z najpopularniejszych takich rozwiązań, są okulary VR, które odcinają użytkownika od prawdziwego świata i pokazują tylko ten wirtualny. Dodatkowo trzeba było odpowiednio oddać dźwięki otoczenia, tak aby odczucia wzorkowe i słuchowe były spójne. Aby poruszać się w takim świecie, potrzebny jest także kontroler. Istnieje aktualnie wiele rozwiązań, gdzie jednymi z najpopularniejszych są VR joystick'i, które trzymane w dłoniach przekazują ich położenie do VR. Jednak jeszcze inne możliwości tworzą interfejsy bezdotykowe. Takim przykładem jest Kinect od firmy Microsoft. Wykorzystuje on zestaw czujników oraz kamer, aby w przestrzeni 3D przenieść ruchy ciała do wirtualnej rzeczywistości. Wtedy kontrolerem staje się sam użytkownik. W tej pracy zostały przeprowadzone badania jak precyzyjnie działają takie kontrolery. Do tego został wykorzystany bezdotykowy interfejs Kinect 360 [1, 2].

2. Opis badania

Na potrzeby artykułu zostało stworzone oprogramowanie pozwalające na zbadanie precyzji działania kontrolera Kinect. Zostało ono wykorzystane do wykonania dwóch badań.

Pierwszym z nich było sprawdzenie, jak dokładnie wychwytywany jest kąt prowadzenia ręki oraz czy jest różnica w wynikach w zależności od ustawienia użytkownika względem Kinect'a. Biorąc pod uwagę, że zasięg działania czujników, który wynosi od 0,8 m do 4,5 m badanie zostało przeprowadzone na dystansie 2,5 m. Dodatkowo wykonano je z trzech punktów oddalonych od siebie o 1 m. Miejsca rozmieszczenia stanowisk zostały pokazane na rysunku 1.



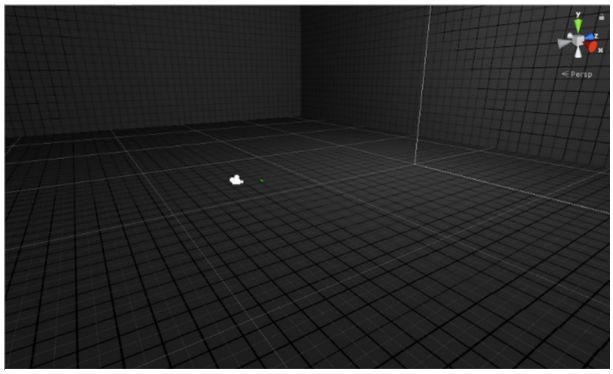
Rysunek 1: Poglądowe miejsca stanowisk wykonania rzutu, podczas badania pierwszego.

Były badane kąty 30 i 45 stopni. W celu jak najdokładniejszego ich oddania, została wykorzystana linka,

ustawiona pod wybranym kątem. Dzięki temu rozwiązaniu ręka poruszając się wzdłuż niej wykonywała ruch pod kątem jak najbardziej zbliżonym do badanego.

Kolejnym badaniem było porównanie odległości uzyskanych po wykonaniu rzutów w rzeczywistości oraz wykonując gest przed Kinect'em. Do rzutu rzeczywistego wykorzystano piłkę tenisową o wadze 60 g oraz średnicy 6,5 cm. Ten rzut był wykonany na świeżym powietrzu w warunkach możliwie bezwietrznych, aby jak najbardziej ograniczyć wpływ warunków atmosferycznych na wyniki. Z kolei rzut w VR został wykonany tak, aby osoba badana widziała zachowanie piłki przed rzutem oraz tor lotu piłki po wykonaniu odpowiedniego gestu. Na rysunku 2 została pokazana plansza, która została stworzona na potrzeby aplikacji.

3. Implementacja aplikacji



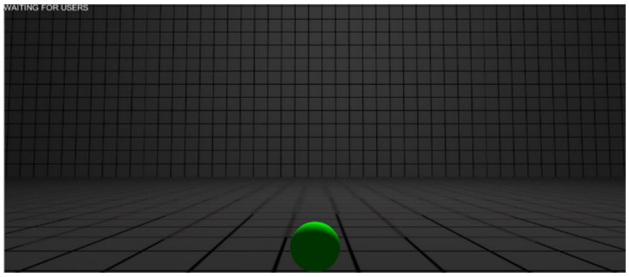
Rysunek 2: Stworzone środowisko w wirtualnej rzeczywistości.

Rysunek 3: Miejsce badawcze z którego zostały wykonywane rzuty do badania nr 2.

Aplikacja została napisana w języku C# przy użyciu środowiska Unity [3,4]. Zastosowano bibliotekę Kinect with MS-SDK, która zawiera w sobie narzędzia, które znacząco przyśpieszyły tworzenie aplikacji [5]. Kinect sprawdza położenie 20 punktów ciała z częstotliwością 30 razy na sekundę [6]. Algorytm, aby wyliczyć moment rzutu, bierze pod uwagę tylko miejsce położenia prawej dłoni. Gest rzutu został podzielony na 4 fazy:

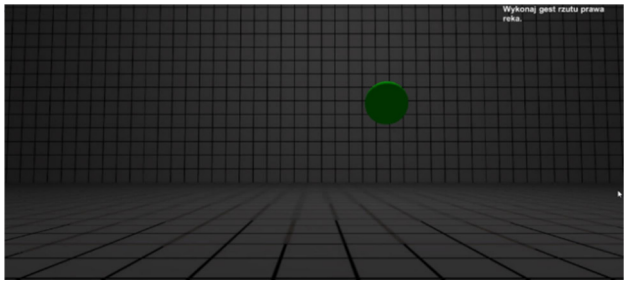
- W pierwszej fazie jest sprawdzane, czy ręka jest na odpowiedniej wysokości względem reszty ciała.
- W drugiej fazie, ręka jest w ruchu, wykonując gest rzutu. Wtedy na podstawie położenie ręki obliczana jest prędkość oraz kąt z jakimi porusza się ręka.
- W trzeciej fazie, ręka zatrzymuje się i Kinect wyłapuje moment, kiedy prędkość była największa. Te informacje zostają wykorzystane, aby nadać prędkość wirtualnej piłce i rozpoczyna się symulacja lotu piłki.
- W czwartej fazie, piłka styka się z podłożem i zostaje wyhamowana. Wtedy też zostają wyświetlone statystyki rzutu, takie jak: prędkość rzutu, kąt wyrzutu oraz odległość rzutu.

Na rysunkach od 4 do 6 pokazano widok aplikacji w trzech kolejnych momentach działania.



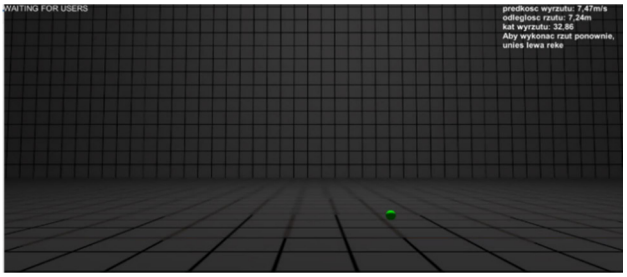
Rysunek 4: Widok aplikacji przed wykryciem użytkownika.

Na rysunku powyższym (Rys. 4) zademonstrowano widok aplikacji w fazie uruchomiania, kiedy użytkownik jeszcze nie jest wykrywany przez kontroler Kinecta.



Rysunek 5: Widok aplikacji po wykryciu użytkownika.

Widok aplikacji, kiedy użytkownik zostanie wykryty (Rys 5). Piłka zaczyna poruszać się tak jak ręka w rzeczywistości. Zostaje wyświetlony komunikat, aby wykonać gest rzutu. Następuje iteracyjna weryfikacja, czy zostały spełnione warunki konieczny, aby ruch ręki zaliczyć jako rzut.

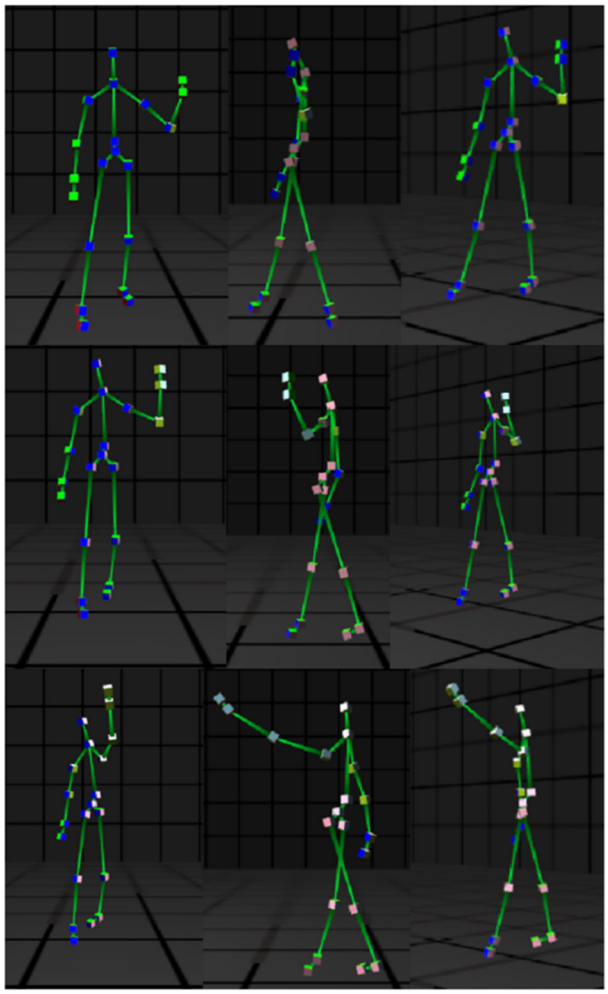


Rysunek 6: Widok aplikacji po wykonanym geście rzutu i skończonej symulacji lotu piłki.

Na rysunku powyższym (Rys. 6) przedstawiono widok aplikacji, kiedy został poprawnie wykryty gest rzutu. Zostaje wykonana symulacja lotu piłki oraz wyświetlona statystyki lotu, którymi są: prędkość wyrzutu, odległość i kąt rzutu. Po zakończonej symulacji jest

informacja, że w celu wykonania ponownie rzutu należy podnieść lewą rękę.

Na rysunku 7 pokazano widok wirtualnego szkieletu podczas wykonywania gestu rzutu.



Rysunek 7: Widok wirtualnego szkieletu użytkownika podczas 3 faz rzutu.

Podczas wykonywania rzutu zbadano koordynaty x, y i z prawej dłoni, którą jest wykonywany rzut.
Koordynat x to jest przesunięcie prawo – lewo względem Kinect'a. Koordynat y to jest przesunięcie góra – dół względem Kinect'a. Koordynat z to jest przesunięcie przód – tył względem Kinect'a.

Dla pierwszej fazy rzutu, gdzie dłoń jest przy barku koordynaty wynosiły:
x = 0,438, y = 0,571, z = 0,132.
Dla drugiej fazy rzutu, gdzie ręka jest w ruchu koordynaty wynosiły:
x = 0,355, y = 0,634, z = -0,112.
Dla trzeciej fazy rzutu, gdzie ręka już wykonała rzut, koordynaty wynosiły:
x = 0,315, y = 0,793, z = -0,552

4. Wyniki badań

Tabele od 1 do 3 pokazują wyniki badania wychwytywanego kata rzutu przed Kinect'em. Poniżej zaprezentowano uzyskane wyniki w trzech tabelach z trzech stanowisk wspomnianych na rysunku 1.
Błąd względny był wyliczany na podstawie wzoru:

$$\Delta x = |x - x_0| \quad (1)$$

Błąd bezwzględny był wyliczany na podstawie wzoru:

$$\delta = \frac{\Delta x}{x} * 100\% = \frac{|x - x_0|}{x} * 100\% \quad (2)$$

Tabela 1: Wyniki ze stanowiska nr 1.

	Sprawdzany kąt 30			Sprawdzany kąt 45		
Numer próby	Kąt zmierzony	Błąd bezwzględny	Błąd względny	Kąt zmierzony	Błąd bezwzględny	Błąd względny
1	38°	8°	26,7%	45°	0°	0,0%
2	30°	0°	0,0%	45°	0°	0,0%
3	32°	2°	6,7%	42°	3°	6,7%
4	35°	5°	16,7%	43°	2°	4,4%
5	32°	2°	6,7%	39°	6°	13,3%
6	25°	5°	16,7%	44°	1°	2,2%
7	30°	0°	0,0%	47°	2°	4,4%
8	31°	1°	3,3%	42°	3°	6,7%
9	33°	3°	10,0%	47°	2°	4,4%
10	35°	5°	16,7%	42°	3°	6,7%

Tabela 2: Wyniki ze stanowiska nr 2.

	Sprawdzany kąt 30			Sprawdzany kąt 45		
Numer próby	Kąt zmierzony	Błąd bezwzględny	Błąd względny	Kąt zmierzony	Błąd bezwzględny	Błąd względny
1	33°	3°	10,0%	39°	6°	13,3%
2	30°	0°	0,0%	39°	6°	13,3%
3	37°	7°	23,3%	42°	3°	6,7%
4	29°	1°	3,3%	35°	10°	22,2%
5	32°	2°	6,7%	39°	6°	13,3%
6	28°	2°	6,7%	41°	4°	8,9%
7	37°	7°	23,3%	44°	1°	2,2%
8	26°	4°	13,3%	45°	0°	0,0%
9	25°	5°	16,7%	42°	3°	6,7%
10	32°	2°	6,7%	43°	2°	4,4%

Tabela 3: Wyniki ze stanowiska nr 3.

	Sprawdzany kąt 30°			Sprawdzany kąt 45°		
Numer próby	Kąt zmierzony	Błąd bezwzględny	Błąd względny	Kąt zmierzony	Błąd bezwzględny	Błąd względny
1	35°	5°	16,7%	42°	3°	6,7%
2	33°	3°	10,0%	40°	5°	11,1%
3	30°	0°	0,0%	48°	3°	6,7%
4	32°	2°	6,7%	42°	3°	6,7%
5	37°	7°	23,3%	35°	10°	22,2%
6	32°	2°	6,7%	44°	1°	2,2%
7	28°	2°	6,7%	40°	5°	11,1%
8	26°	4°	13,3%	43°	2°	4,4%
9	29°	1°	3,3%	42°	3°	6,7%
10	29°	1°	3,3%	45°	0°	0,0%

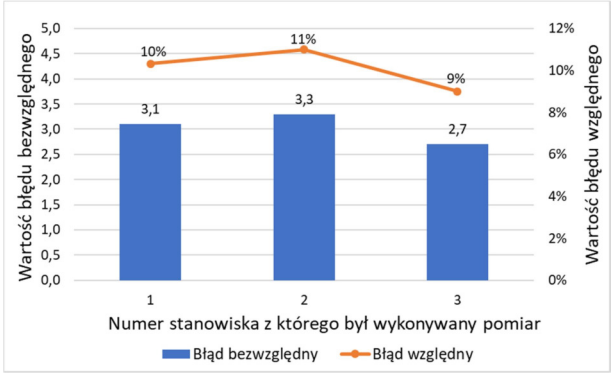
Dla stanowiska nr 1 dla kąta 30° średnia z prób wynosi 32.10° mediana 32° a odchylenie standardowe 3,54°. Dla kąta 45° średnia wynosi 43,60° mediana 43,50° a odchylenie standardowe 2,50°.

Dla stanowiska nr 2 dla kąta 30° średnia z prób wynosi 30,90° mediana 31° a odchylenie standardowe 4,12°. Dla kąta 45° średnia wynosi 40,90° mediana 41,50° a odchylenie standardowe 2,96°.

Dla stanowiska nr 3 dla kąta 30° średnia z prób wynosi 31.10° mediana 31° a odchylenie standardowe 3,35°. Dla kąta 45° średnia wynosi 42,10° mediana 42° a odchylenie standardowe 3,45°.

Po przeprowadzonych pomiarach dla kąta 30° widać, że między stanowiskami nie ma dużych różnic, jeżeli chodzi o błąd względny i błąd bezwzględny. Dla trzech stanowisk błąd względny wyniósł 3° a błąd bezwzględny wyniósł 10%.

Po przeprowadzonych pomiarach dla kąta 45° widać, że między stanowiskami nie ma dużych różnic, jeżeli chodzi o błąd względny i błąd bezwzględny. Dla trzech stanowisk błąd względny wyniósł 3° a błąd bezwzględny wyniósł 7%.

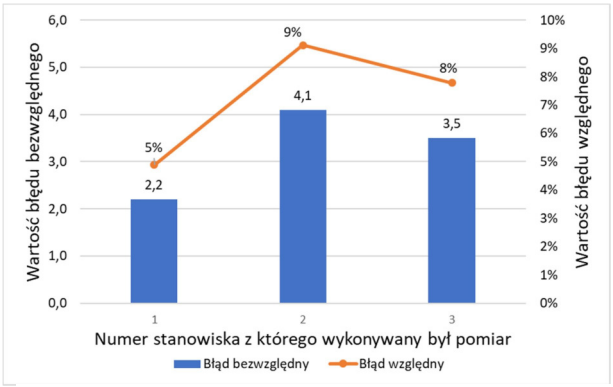


Rysunek 8: Wykres porównujący błąd względny i bezwzględny dla badanego kąta 30° między stanowiskami.

W tym badaniu dla każdego stanowiska było wykonanych po 10 pomiarów. Dla badanego kąta 30° najbliższa średnia wniosła 30.90°, kiedy rzut był wykonywany na stanowisku 2. W przypadku stanowiska 1 średnia wynosiła 32.10° a stanowiska 3 - 31.10°. Pokazuje to, że różnica pomiędzy wynosi 1.20° między najbliższym a najdalszym wynikiem. Dodatkowo różnica pomiędzy kątami uzyskanymi a pożądanymi wynosi od 0.9° to 2.1°. W przypadku badanego kąta 45° najbliższą średnią było 43.60°, kiedy badanie było wykonane na stanowisku 1. Gest przeprowadzony na stanowisku 2 dał wynik 40.90° a stanowisku 3 - 42.10°. Daje to 3.70° różnicy pomiędzy najmniejszą a największą średnią. Patrząc na różnice od kąta 45° daje od 1.40° do 4.10°.

Patrząc natomiast na oddalenie wszystkich wartości od średniej w przypadku badanego kąta 30° najmniejszą wartością jest 3.35° przy badaniu ze stanowiska 3, 3.54° przy badaniu na stanowisku 2, a największą 4.12° przy badaniu ze stanowiska 3. Przy kącie 45° najmniejszą wartością jest 2.50° dla badania na stanowisku 1, 2.96° przy badaniu ze stanowiska 2 oraz największą wartością jest 3.45° w przypadku stanowiska 3.

W tabeli 4 przedstawiono wyniki badań rzutów przed Kinect'em i w rzeczywistości. Rzut rzeczywisty został wykonany na świeżym powietrzu w warunkach możliwie bezwietrznych, aby jak najdokładniej wykonać pomiary.



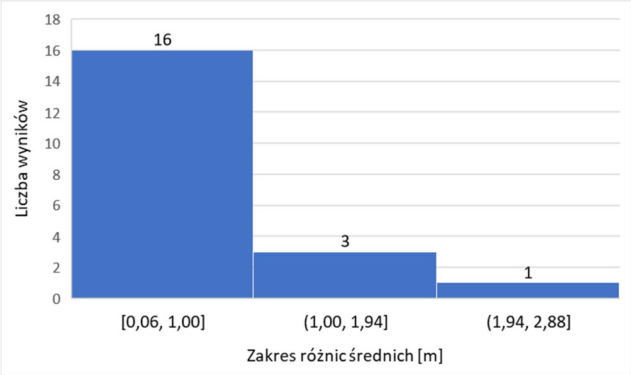
Rysunek 9: Wykres porównujący błąd względny i bezwzględny dla badanego kąta 45° między stanowiskami.

Przy obliczaniu wartości błędu względnego i bezwzględnego wartością uznaną za rzeczywistą była średnia z rzutów rzeczywistych w danym badaniu.

Tabela 4: Wyniki badania 2.

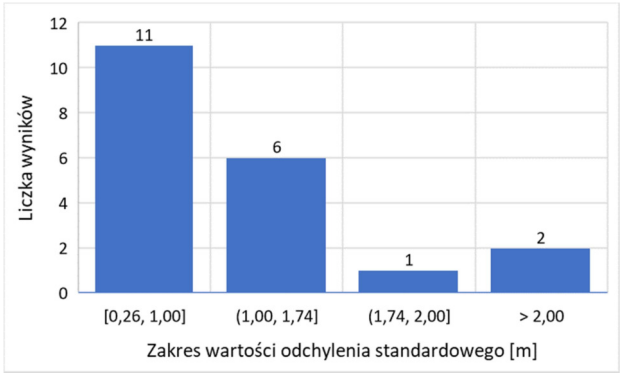
Badanie nr 1					Badanie nr 2					Badanie nr 3					Badanie nr 4				
l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]
1	6,50	4,40	1,60	27%	1	8,20	8,20	0,10	1%	1	11,00	10,50	0,48	4%	1	12,90	11,00	2,94	21%
2	5,00	10,00	4,00	67%	2	8,30	5,00	3,10	38%	2	10,60	6,00	4,98	45%	2	12,40	12,40	1,54	11%
3	6,30	3,00	3,00	50%	3	7,70	8,20	0,10	1%	3	11,20	15,00	4,02	37%	3	17,70	13,00	0,94	7%
4	6,40	6,20	0,20	3%	4	8,00	12,00	3,90	48%	4	11,70	11,00	0,02	0%	4	12,60	7,00	6,94	50%
5	5,80	5,80	0,20	3%	5	8,30	7,50	0,60	7%	5	10,40	9,00	1,98	18%	5	14,10	13,30	0,64	5%
średnia	6,00	5,88	1,80	30%	średnia	8,10	8,18	1,56	19%	średnia	10,98	10,30	2,30	21%	średnia	13,94	11,34	2,60	19%
odchylenie standardowe	1,54	27,6			odchylenie standardowe	0,26	25,2			odchylenie standardowe	1,05	42,8			odchylenie standardowe	19,4	26,7		
Badanie nr 5					Badanie nr 6					Badanie nr 7					Badanie nr 8				
l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]
1	6,10	7,20	1,06	21%	1	8,00	7,00	1,26	15%	1	11,60	11,20	0,68	6%	1	15,00	14,50	0,52	3%
2	6,00	6,80	0,66	13%	2	7,90	10,00	1,74	21%	2	11,70	10,80	1,08	9%	2	14,40	13,80	1,22	8%
3	5,80	8,00	1,86	36%	3	8,10	8,00	0,26	3%	3	12,60	8,90	2,98	25%	3	15,50	17,00	1,98	13%
4	6,40	6,00	0,14	3%	4	8,90	8,50	0,24	3%	4	11,50	12,00	0,12	1%	4	15,00	20,00	4,98	33%
5	6,40	10,00	3,86	75%	5	8,40	6,00	2,26	27%	5	12,00	13,00	1,12	9%	5	15,20	15,00	0,02	0%
średnia	6,14	7,60	1,52	29%	średnia	8,26	7,90	1,15	14%	średnia	11,88	11,18	1,20	10%	średnia	15,02	16,06	1,74	12%
odchylenie standardowe	0,27	9,28			odchylenie standardowe	0,65	9,2			odchylenie standardowe	0,79	9,33			odchylenie standardowe	0,65	25,1		
Badanie nr 9					Badanie nr 10					Badanie nr 11					Badanie nr 12				
l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]
1	8,00	8,50	0,02	0%	1	10,50	9,40	1,18	11%	1	12,40	8,00	3,90	33%	1	14,00	9,00	4,80	35%
2	9,00	5,10	3,42	40%	2	11,30	7,30	3,28	31%	2	11,80	12,30	0,40	3%	2	13,20	11,00	2,80	20%
3	8,10	8,00	0,52	6%	3	10,60	10,30	0,28	3%	3	11,70	11,70	0,20	2%	3	14,30	14,00	0,20	1%
4	9,20	9,90	1,38	16%	4	10,10	11,00	0,42	4%	4	11,60	12,00	0,10	1%	4	13,10	13,10	0,70	5%
5	8,30	8,10	0,42	5%	5	10,40	14,60	4,02	38%	5	12,00	11,90	0,00	0%	5	14,40	17,00	3,20	23%
średnia	8,52	7,92	1,15	14%	średnia	10,58	10,52	1,84	17%	średnia	11,90	11,18	0,92	8%	średnia	13,80	12,82	2,34	17%
odchylenie standardowe	1,19	12,25			odchylenie standardowe	0,79	28,55			odchylenie standardowe	0,40	12,83			odchylenie standardowe	1,50	36,85		
Badanie nr 13					Badanie nr 14					Badanie nr 15					Badanie nr 16				
l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]
1	7,50	6,80	0,88	11%	1	10,20	7,00	3,34	32%	1	11,00	6,00	5,46	48%	1	13,50	17,00	3,50	26%
2	7,70	7,50	0,18	2%	2	10,90	9,40	0,94	9%	2	11,50	9,00	2,46	21%	2	14,00	13,00	0,50	4%
3	8,00	10,00	2,32	30%	3	11,00	14,00	3,66	35%	3	11,60	12,00	0,54	5%	3	14,30	14,20	0,70	5%
4	8,00	8,00	0,32	4%	4	9,90	10,30	0,04	0%	4	11,00	11,50	0,04	0%	4	13,20	10,00	3,50	26%
5	7,20	4,00	3,68	48%	5	9,70	10,00	0,34	3%	5	12,20	11,00	0,46	4%	5	12,50	12,70	0,80	6%
średnia	7,68	7,26	1,48	19%	średnia	10,34	10,14	1,66	16%	średnia	11,46	9,90	1,79	16%	średnia	13,50	13,38	1,80	13%
odchylenie standardowe	0,47	18,95			odchylenie standardowe	1,37	25,35			odchylenie standardowe	0,99	24,20			odchylenie standardowe	1,98	25,81		
Badanie nr 17					Badanie nr 18					Badanie nr 19					Badanie nr 20				
l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]	l.p.	Rzut rzeczywisty [m]	Rzmut w VR [m]	Błąd względny [m]	Błąd bezwzględny [m]
1	4,80	3,00	1,84	38%	1	9,60	8,00	2,70	25%	1	15,00	12,00	3,30	22%	1	18,00	17,00	1,56	8%
2	4,20	4,90	0,06	1%	2	10,20	9,40	1,30	12%	2	15,30	16,50	1,20	8%	2	19,30	18,50	0,06	0%
3	5,30	5,00	0,16	3%	3	12,00	11,90	1,20	11%	3	16,10	18,00	2,70	18%	3	18,50	19,00	0,44	2%
4	5,00	8,00	3,16	65%	4	11,10	10,30	0,40	4%	4	14,90	15,00	0,30	2%	4	19,10	25,00	6,44	35%
5	4,90	4,40	0,44	9%	5	10,60	11,20	0,50	5%	5	15,20	16,20	0,90	6%	5	17,90	17,90	0,66	4%
średnia	4,84	5,06	1,13	23%	średnia	10,70	10,16	1,22	11%	średnia	15,30	15,54	1,68	11%	średnia	18,56	19,48	1,83	10%
odchylenie standardowe	0,65	13,35			odchylenie standardowe	3,32	9,37			odchylenie standardowe	0,90	20,23			odchylenie standardowe	1,59	40,31		

Rysunki od 10 do 13 pokazują graficzną interpretacje wyników przeprowadzonego badania, w którym mierzono różnice pomiędzy pomiarem rzutu w VR a w rzeczywistości



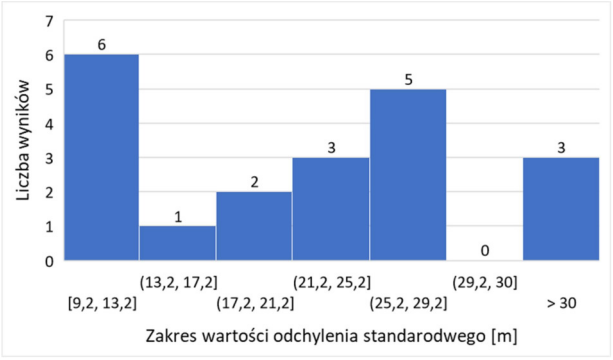
Rysunek 10: Histogram różnic pomiędzy średnimi z rzutu rzeczywistego i rzutu w wirtualnej rzeczywistości.

Badanie pokazujące różnice pomiędzy rzutem rzeczywistym a w wirtualnej rzeczywistości pokazują, że różnica średnich pomiędzy dwoma rodzajami rzutów nie jest większa niż 1,50 m. Największą wartością jest 2,60 m. W 16 z 20 badań ta różnica wynosiła poniżej 1 m.



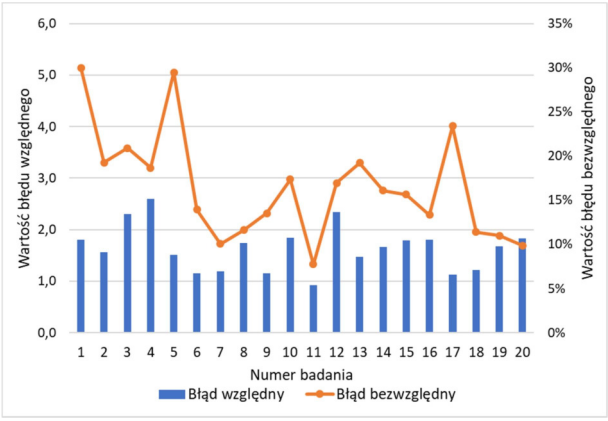
Rysunek 11: Histogram wyników odchyleń standardowych dla rzutu rzeczywistego.

Inaczej wygląda analiza odchyleń standardowych dwóch typów rzutów. W przypadku rzutu rzeczywistego, 11 z 20 rzutów miało odchylenie standardowe na poziomie do 1 m. Kolejne 6 z 20 do 1,74 m a tylko 3 rzuty powyżej 1,74 m. Pokazuje to dosyć niewielkie rozproszenie wyników rzutów.



Rysunek 12: Histogram wyników odchyleń standardowych dla rzutu wirtualnego.

W przypadku rzutu w wirtualnej rzeczywistości najmniejszą wartością odchylenia standardowego było 9,20 m. Wyniki 6 z 20 badań pokazało odchylenie standardowe na poziomie od 9,20 m do 13,20 m. Kolejne 11 z 20 badań dało wyniki na poziomie od 13,20 m do 29,20 m a 3 z 30 badań dało wyniki powyżej 30 m.



Rysunek 13: Wartości błędu względnego i błędu bezwzględnego w badaniu nr 2.

Analiza błędów bezwzględnych i błędów względnych pokazuje, że dokładność pomiarów nie jest idealna. Dla 20 przebadanych osób, z których każda wykonała po 5 pomiarów rzeczywistych i wirtualnych wychodzi, że błąd bezwzględny średnio wynosi 16,47%, a błąd względny wynosi średnio 1,64 m.

5. Wnioski

Na samym początku artykułu została przedstawiona technologia wirtualnej rzeczywistości. Zostały opracowane badania, które pozwolą sprawdzić precyzje działania Kinect'a. Następnie została opracowana i napisana aplikacja, za pomocą której można było przeprowadzić badania.

Wyniki przeprowadzonych badań pokazują, że technologia Kinect (pierwszej generacji) działa stosunkowo precyzyjnie. Jednak analiza wychwytywania kątów wykazała, że próby wykonywane przez człowieka nie były idealne. Świadczą o tym wyliczenia, które pokazują, że błąd bezwzględny wyniósł 10% dla kata 30 stopni oraz 7% dla 45 stopni. Błąd względny wyniósł 3 stopnie dla obu badanych kątów. Należy zwrócić uwagę, na to, że pomiary były wykonywane ręcznie, a nie za pomocą np. mechanicznego ramienia. Takie ramie dałoby odpowiedni kąt i zostałby wtedy wykluczony błąd spowodowany nierównym ruchem przez człowieka. Dodać do tego należy, że wykorzystywana była najstarsza wersja Kinect'a, wyprodukowana w 2010 r. Jest ona o wiele mniej precyzyjna od nowszej wersji „Kinect v2 for Windows". Na podstawie filmów testowych załączonych w Internecie [7-10] można byłoby wywnioskować, że nowsza wersja dałaby dokładniejsze wyniki.

Następnym badaniem było porównanie odległości uzyskanych w rucie wirtualnym i rzeczywistym. Błąd bezwzględny wyniósł ponad 16% a względny 1,64 m. Różnice średnich pomiędzy oboma typami rzutów w zdecydowanej większości są niskie, wynoszą do 1 m. Badając jednak odchylenie standardowe widać o wiele

większe różnice. W przypadku rzutu rzeczywistego wartości 17 z 20 badań dochodzą do 1,74 m. Natomiast patrząc na wyniki z rzutu wirtualnego wartości zaczynają się od 9.2 m. Wyniki 17 z 20 wykazują odchylenie standardowe do 29 m. Jest to duża różnica. Taki wynik może powodować kilka problemów. Jednym z nich jest stosunkowo niewielka liczba rejestrowanych klatek obrazu na sekundę. Jest ich 30, co oznacza, że pozycja ciała jest rejestrowana co ok. 33 ms. Ruch ręki wykonującej rzut trwa średnio 0.1-0.15 s, co oznacza, że łapanych jest od 3 do 5 pomiarów pozycji ciała. Jest to niską wartością i utrudnia precyzyjne wyliczenie siły rzutu. Dodatkowo przy sprawdzaniu działania wykonywanego programu, czasy pojedynczych pomiarów wynosiły pomiędzy 20 ms a 40 ms, co przy przyjętej metodzie komplikowało wyliczanie prędkości piłki. Zastosowany algorytm, wyliczał prędkość dłoni pomiędzy każdym z wychwytywanych punktów, po czym największą prędkość wybierał jako moment oddania rzutu. Zdarzały się jednak bardzo często sytuacje, że koniec ruchu ręki, gdzie jej prędkość była największa następował w połowie pomiaru pozycji ciała, przez co algorytm wychwytywał mniejszą prędkość niż przy poprzednim pomiarze. Zostało to skorygowane, poprzez mnożenie ostatniej prędkości razy dwa, ponieważ taka wartość zazwyczaj była bliska prawdziwej.

W następnych wersjach aplikacji można zmienić model wykorzystywanego Kinect'a. Mogłoby zmniejszyć wagę niepoprawnych pomiarów oraz ustabilizować czasy pomiędzy pojedynczymi pomiarami. Dodatkowo można by napisać dokładniejszy algorytm wyliczający dokładną prędkość dłoni, który by rozwiązywał problem, który pojawia się, gdy ruch ręki kończy się przed ostatnim pomiarem. Wtedy rzut wirtualną piłką będzie maksymalnie zbliżony do tego wykonywanego w rzeczywistości.

Literatura

[1] Ł. Pełka, Ł. Podstawka, T. Szymczyk, Analiza porównawcza gogli do VR, Journal of Computer Sciences Institute 10 (2019) 34-44.

[2] Wirtualna Rzeczywistość, https://systel.pl/virtual-reality/, [14.10.2020].

[3] A. Kempa, T. Staś, Wstęp do programowania w C#: Łatwy podręcznik dla, Wydawnictwo Helion, 2014

[4] Unity, https://unity.com/, [14.10.2020].

[5] Kinect with MS-SDK, https://assetstore.unity.com/packages/tools/kinect-with-ms-sdk-7747, [14.10.2020].

[6] Kinect, https://en.wikipedia.org/wiki/Kinect, [14.10.2020].

[7] Kinect v1 vs Kinect v2, https://www.youtube.com/watch?v=Zx2E19IV2zs, [22.10.2020].

[8] Kinect v1 vs Kinect v2, https://www.youtube.com/watch?v=eNIP9nFo9n4, [22.10.2020].

[9] Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision, https://www.youtube.com/watch?v=FznorYFODCY, [22.10.2020].

[10] Kinect for Windows v1 vs v2 - Skeleton Tracking, https://www.youtube.com/watch?v=0swQqPqdd2w, [22.10.2020].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 22–29
Received: 22 November 2020
Accepted: 26 January 2021

Analysis of modern human-computer interfaces

Analiza współczesnych interfejsów człowiek-komputer

Michał Cioczek*, Tomasz Czarnota, Tomasz Szymczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
Until recently, computers were devices intended for research and military processes, and today they are an integral part of human life. Such a rapid development of technology and new possibilities means that the science that deals with human-computer relations (HCI) has a lot of unexplored or insufficiently studied areas. Based on the available HCI research methods, this article is to show that the currently highly popularized computer mouse, which is a tool that enables HCI, will not always be the best solution for every user. Comparing a computer mouse with other commercially available controllers and testing them in a specially prepared test environment by a selected research group will be the best way to reliably state that the use of a computer mouse may not always be the best choice.

Keywords: Human Computer Interaction; ergonomics; interface; controller

Streszczenie
Jeszcze do niedawna komputery były urządzeniami przeznaczonymi dla procesów badawczych i militarnych, a dziś stanowią integralną część życia człowieka. Tak szybki rozwój technologii i nowych możliwości sprawia, że nauka, która zajmuje się relacjami człowieka z komputerem (HCI) ma pełno niezbadanych lub niewystarczająco przebadanych obszarów. Niniejszy artykuł na podstawie dostępnych metod badawczych HCI ma pokazać, że obecnie mocno spopularyzowana mysz komputerowa, która jest narzędziem umożliwiającym HCI, nie zawsze będzie najlepszym rozwiązaniem dla każdego użytkownika. Zestawienie myszy komputerowej z innymi dostępnymi na rynku kontrolerami i przebadanie ich w specjalnie przygotowanym środowisku testowym przez wybraną grupę badawczą, będzie najlepszym sposobem do rzetelnego stwierdzenia, że użycie myszy komputerowa nie zawsze może okazać się najlepszym wyborem.

Słowa kluczowe: Interakcja Człowiek-Komputer; ergonomia; interfejs; kontroler

*Corresponding author
E-mail address: **michal.cioczek@pollub.edu.pl** (M. Cioczek)

1. Wstęp

Współcześnie komputery oraz smartfony stały się nie tylko narzędziami pomocnymi człowiekowi, ale wręcz koniecznymi do funkcjonowania w życiu codziennym. Aspektem nauki, który stara się nie tylko zbadać relację między człowiekiem, a komputerem, ale i pomóc odkryć lub ulepszyć dostępne technologie umożliwiające taką zależność jest tzw. interakcja człowiek-komputer (ang. *HCI, czyli human-computer interaction*). Początki HCI sięgają połowy XX wieku, kiedy komputery zaczęły być wykorzystywane na szeroką skalę przez wojsko. Rozkwit tej nauki nastąpił natomiast w latach 80. XX wieku, kiedy komputery zaczęły pojawiać się w domach zwykłych ludzi [3].

Interakcja człowieka z komputerem jest niezwykle ciekawym i szerokim tematem, a że i niedostatecznie zbadanym, więc warto się w niego zagłębić. Szczególnie interesujące wydaje się wykorzystanie kontrolerów ruchowych umożliwiających wspomnianą interakcję. Na rynku są obecne różne kontrolery mające odmienną budowę i specyfikę. Wiele z nich dąży jednak do wspólnego celu, jakim jest przechwycenie obiektu, którym użytkownik będzie sterować i wykonywać nim akcje w przeznaczonym do tego celu środowisku (aplikacja). Ze względu, że najbardziej popularnym obiektem tego typu jest kursor ekranowy, jego przechwycenie przez wybrane kontrolery ruchowe wydaje się być najlepszym wyborem do badania HCI.

Skąd pomysł, aby zbadać to zjawisko? Otóż obecnie ogrom ludzi korzysta z komputera i może się okazać, że szeroko spopularyzowana w ubiegłym wieku mysz komputerowa nie zawsze będzie najlepszym wyborem dla każdego użytkownika. W artykule opisano i wykorzystano metody badawcze HCI, które to sprawdzą [4].

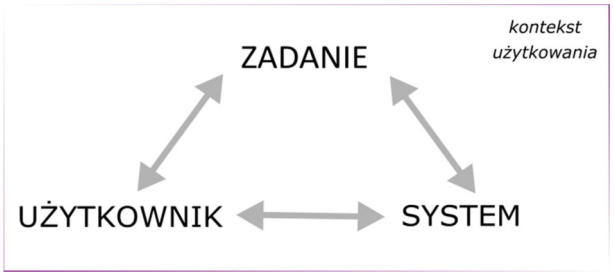
Aby umożliwić przeprowadzenie takiego badania autorzy wybrali do porównania kilka z dostępnych na rynku kontrolerów będących urządzeniami wejścia, przygotowali środowisko testowe pozwalające na zebranie i przetworzenie wyników oraz wyznaczyli grupę badawczą.

Ponadto autorzy wybrali metody badawcze, które pozwolą zbadać interakcję człowieka z komputerem w sposób liczbowy, ale i metody pozwalające określić aspekt odczuć człowieka. Według badań w zakresie HCI okazuje się, że jedynie zestawienie wyników liczbowych wraz z wywiadem z grupą badawczą, pozwalają pokazać pełny obraz badanego zjawiska i umożliwić wyciągnięcie wniosków [4].

2. Metoda badań

Interakcja człowiek-komputer (HCI) jest dziedziną nauki, która posiada wiele metod badawczych pozwala-

jących na opisanie relacji użytkownika z systemem. Taka relacja została przedstawiona w modelu człowiek-praca-środowisko, który ma swoje podłoże w tzw. ergonomii, czyli dyscyplinie naukowej zajmującej się przystosowaniem narzędzi używanych przez użytkownika do jego anatomicznych i psychofizycznych potrzeb [3].



Rysunek 1: Model człowiek-praca-środowisko [9]

Do opisania wspomnianej relacji jako metodę czy właściwie metody ogólne, należy podać metody jakościowe, na których analizie opierają się wszystkie metody związane z HCI. Do metod jakościowych należą:

- jakość konstrukcyjna (techniczna), która skupia się na budowie oraz wydajności interfejsu, czyli głównie na aspekcie wartości, które da się opisać liczbowo;
- jakość ergonomiczna, która skupia się na odczuciach użytkownika podczas korzystania z interfejsu (ergonomia);
- jakość użytkowa (użyteczność), która skupia się na satysfakcji użytkownika (czyli ukazuje zadowolenie użytkownika z interfejsu, którego używał) [3], [8].

2.1. Metoda obliczeniowa

Jako pierwszą z metod należy wskazać model KLM (ang. *Keystroke Level Model*), który wskazuje standardowe czasy trwania elementarnych operacji (Tabela 1). Model ten wywodzi się z techniki GOMS (ang. *Goals, Operators, Methods, Selection Rules*), która uwzględnia również zadania umysłowe, do jakich między innymi należy wybór sposobu wykonania operacji [8].

Metoda KLM zakłada, że użytkownik jest wprawiony, a więc umie posługiwać się danym interfejsem, zna swoje zadanie i kolejność kroków wykonywanej operacji. Dodatkowo są wyznaczone typowe czasy wykonania poszczególnych działań, dzięki czemu można oszacować ile będą trwały proste zadania oraz dłuższe sekwencje złożone z mniejszych kroków. W modelu KLM zakłada się, że użytkownik zna następujące sekwencje operacji:

- K - Press Key (kliknięcie przycisku klawiatury);
- P - Pointing (wskazanie przy pomocy kursora elementu docelowego);
- H - Hand (przeniesienie ręki pomiędzy myszą, a klawiaturą);
- M - Mental (mentalne przygotowanie się do wykonania operacji manualnej);
- D - Drawing (rysowanie drogi, którą pokonuje kursor);
- R - Response (odpowiedź systemu);
- B - Press Button (kliknięcie przycisku myszy) [9].

Tabela 1 Wybrane operacje użytkownika w modelu KLM [9]

AKCJA	OPIS	CZAS [s]
K	Naciśnięcie klawisza klawiatury - wprawny operator - przeciętny operator - początkujący operator - niedoświadczony oper.	 0,12 0,22 0,28 1,2
P	Wskazanie docelowego obiektu kursorem myszy - na podstawie prawa Fittsa - przeciętnie	 $0{,}1\log_2(0{,}5+\frac{D}{S})$ 1,1
H	Przeniesienie dłoni do miejsca docelowego z/do myszy lub klawiatury	0,4
M	Przygotowanie się do wykonania operacji manualnej pod względem mentalnym	1,35
D	Rysowanie drogi, którą przebywa kursor	zależne od zadania
R	Odpowiedź systemu	zależne od zadania
B	Kliknięcie przycisku myszy	0,1

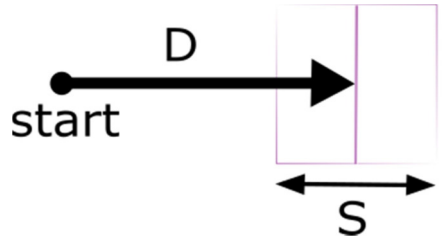
Na podstawie zaprezentowanych w tabeli wartości i danych, możliwe jest opracowanie scenariusza badawczego podzielonego na pojedyncze akcje. Tworzy się go sumując czasy poszczególnych czynności, budując tym samym łańcuchy operacji użytkownika. W rezultacie można oszacować średni czas realizacji całego zadania.

Do badania ruchu kursora systemowego przy pomocy kolejnych kontrolerów ruchowych najbardziej trafne jest wykorzystanie tzw. prawa Fittsa opisującego w sposób liczbowy motoryczne rekcje człowieka. Ten model pomija kwestie pomyłek i precyzji ruchu, a skupia się na wyznaczeniu optymalnego czasu dla wykonywanej operacji (zakłada się, że użytkownik jest wprawiony). Innymi słowy prawo Fittsa pozwala obliczyć czas konieczny do dosięgnięcia obiektu graficznego (znajdującego się na ekranie monitora) przy pomocy kursora lub palcem, z założeniem, że istnieje obiekt startowy (Rysunek 2). Analizie muszą podlegać wyćwiczone wcześniej ruchy, które będą powtarzalne, a obiekt będzie w zasięgu ramienia. Prawo Fittsa można opisać wzorem znanym również jako formuła Shannona [8]:

$$T = \mathrm{a} + \mathrm{b} \times log_2(\mathrm{c} + \frac{D}{S}) \quad (1)$$

gdzie T to całkowity czas mierzony w sekundach wymagany do realizacji dosięgnięcia obiektu docelowego, a to stała opisująca czas mierzony w sekundach potrzebny do reakcji niezbędnej do rozpoczęcia ruchu (opóźnienie), b to stała czasowa mierzona w sekundach związana z prędkością ruchu urządzenia wskazującego, którym steruje użytkownik, c to stała zależna od środowiska człowiek-maszyna, która może przyjmować wartości 0, 0.5 lub 1 (w przypadku systemów informatycz-

nych będzie to 1), D to odległość liczona w jednostkach miary odległości (np. cm lub px) od punktu początkowego obiektu do jego środka, a S to szerokość obiektu docelowego liczona w jednostkach miary odległości (np. cm lub px).



Rysunek 2: Model geometryczny osiągnięcia celu na ekranie [8]

Kluczowe jest, że stałe a i b powinny być wyznaczane w sposób empiryczny i mogą przyjmować inne wartości dla różnych urządzeń użytych w badaniu [9].

2.2. Ocena z udziałem użytkowników

Poza metodami liczbowymi należy zbadać odczucia użytkownika oraz ergonomię interfejsu graficznego i kontrolerów. W tym celu należy wykonać ocenę z udziałem użytkowników, a więc trzeba zebrać grupę badawczą, która wykona testy i odpowie na pytania w przygotowanym odpowiednio wywiadzie.

Aby przeprowadzić testy użyteczności należy zadbać o następujące składowe:

1. Założenia metodyczne:
   a) oszacowanie spodziewanych wyników i ustalenie prawdopodobnego wskaźnika satysfakcji użytkowników,
   b) ustalenie jakie dane będą zbierane podczas testu,
   c) ustalenie jakie kryteria powinny być spełnione, aby zebrane dane były użyteczne oraz jaki może być próg błędu,
   d) ustalenie wymagań jakie musi spełniać grupa badawcza.
2. Aspekt etyczny:
   a) przygotowanie testu o maksymalnej długości 90 minut w celu zapewnienia komfortu psychicznego członków grupy badawczej,
   b) zapewnienie warunków podczas wykonywania testu, w których członkowie grupy badawczej będą czuli komfort pracy i bezpieczeństwo.
3. Zadania testowe:
   a) wydrukowanie i omówienie z użytkownikiem zadań, które muszą być klarowne i dobrze zrozumiałe,
   b) upewnienie się, czy zadanie jest krótkie i czy obejmuje wszystkie założone do zbadania aspekty,
   c) jeśli do zadania należy użyć materiałów pomocniczych, należy je użytkownikowi wskazać lub zapewnić,
   d) kolejne zadania powinny być trudniejsze od poprzednich, ale bez przesadnego stopnia trudności. Należy również pamiętać o fakcie, że z różnych powodów użytkownik może nie ukończyć danego zadania lub zadań.
4. Rekrutacja uczestników:
   a) należy ustalić kryteria szukanych osób do grupy badawczej: liczebność, wiek, płeć, stopień doświadczenia, itp.,
   b) testy użyteczności należy przeprowadzać w oddzielnych grupach lub pojedynczo,
   c) określenie wymagań odnośnie do miejsca, w którym testy użyteczności zostaną przeprowadzone, do metod pozyskania uczestników oraz, jeśli jest taka potrzeba, do wysokości wynagrodzenia.
5. Zbieranie danych:
   a) przygotowanie miejsca i technologii potrzebnej do przeprowadzenia badania oraz zadbanie, by urządzenia były skalibrowane i działały poprawnie,
   b) zapewnienie pomocnika, który będzie gotów pomóc uczestnikom badania na każdym etapie testu,
   c) stworzenie mechanizmu zbierania danych lub nagranie testu przy pomocy kamery,
   d) przeprowadzenie wywiadu po zakończeniu testu użyteczności lub zalecenie wypełnienia ankiety oceny satysfakcji.
6. Miary użyteczności:
   a) oszacowanie czasów ukończenia poszczególnych zadań przez uczestników testu,
   b) stworzenie statystyki błędów oraz powodzeń i niepowodzeń dla zakończonych testów,
   c) wskazanie liczby użytych funkcji i poleceń,`
   d) wskazanie, czy było zapotrzebowanie na pomoc obsługi bądź materiałów pomocniczych, a jeśli tak, to zapisanie czasów dla tego typu zachowań,
   e) wskazanie liczby powtórzeń testów lub, w przypadku licznych niepowodzeń, szukanie alternatywnych metod rozwiązania problemu,
   f) zapisanie liczby zgłoszeń dotyczących niezadowolenia oraz utraty kontroli nad sprzętem,
   g) określenie wszystkich innych miar użyteczności, które będą potrzebne do zbadania interfejsu, ale nie zostały powyżej wymienione [5], [8].

3. Opis wykorzystanych urządzeń i technologii

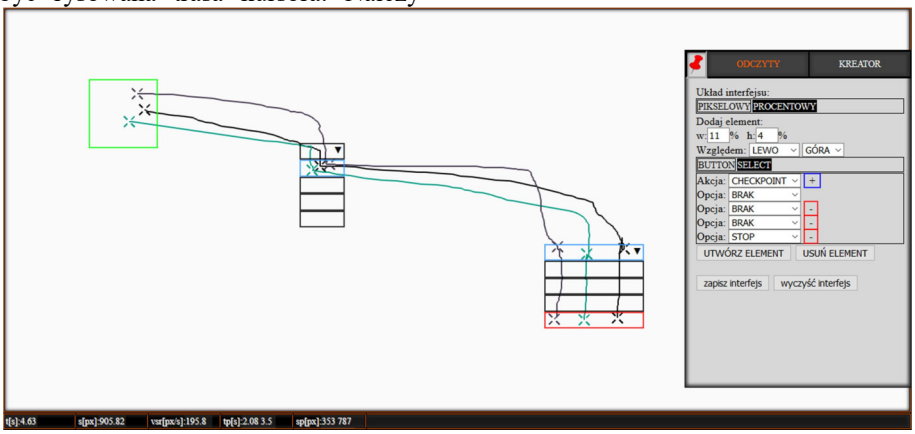
Metody badawcze są niezwykle istotne, jednak są tylko teorią bez środowiska testowego, manipulatorów ruchowych i człowieka, który będzie tej technologii używać. Testy są wykonywane dopiero w momencie, gdy istnieje technologia do zbadania, więc autorzy niniejszego artykułu musieli taką technologię stworzyć lub wykorzystać tę istniejącą na rynku.

3.1. Aplikacja do testowania interfejsów urządzeń

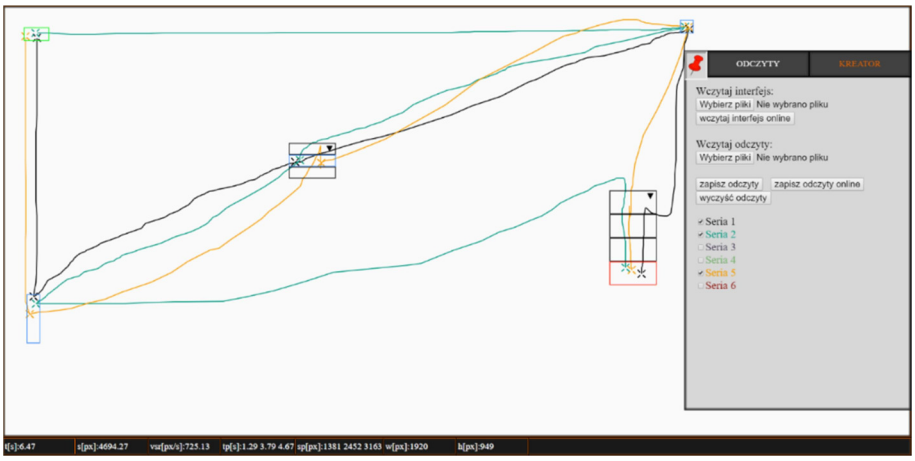
Należy zacząć od opisu środowiska testowego, w którym grupa użytkowników mogłaby zbadać wybrane współczesne kontrolery wejścia pod względem HCI. W tym celu autorzy artykułu przygotowali aplikację webową umożliwiającą przygotowanie odpowiednich testów oraz pozwalającą na zbadanie i zapisanie wyników do dalszej analizy.

Aplikacja została przygotowana w hipertekstowym języku znaczników HTML w wersji 5 w oparciu o arkusz stylów CSS w wersji 3. Wykorzystano również język javascript oraz stworzoną w nim bibliotekę jQuery. Aplikacja działa w oknie przeglądarki internetowej i umożliwia pracę w trybie offline lub online (w tym przypadku musi być umieszczona na serwerze zewnętrznym). Wykorzystane standardy sprawiają, że konieczne jest używanie stosunkowo nowych wersji przeglądarek internetowych [2], [4].

Celem aplikacji jest możliwość stworzenia środowiska testowego dla kursora systemowego, który będzie sterowany różnorodnymi kontrolerami ruchowymi, zwanymi również manipulatorami ruchowymi. Aplikacja umożliwia stworzenie jednego bloku startu i jednego bloku końcowego oraz dowolnej liczby bloków pośrednich, a same bloki mogą być różnej wielkości. Po najechaniu kursorem na blok startu i naciśnięciu odpowiedniego przycisku (np. w przypadku myszy komputerowej LPM) zaczyna być rysowana trasa kursora. Należy klikać w bloki pośrednie, a na końcu w blok końcowy. Po naciśnięciu ostatniego bloku trasa przestanie być rysowana pod warunkiem, że użytkownik nie pominął żadnego z wcześniejszych bloków. Cały ekran testowy działa w przeglądarce internetowej, można go projektować ze stałą szerokością lub z zachowaniem zasad responsywności. Interfejs oraz wyniki można zapisywać oraz wczytywać na lokalnym nośniku danych, np. na dysku twardym komputera lub online na serwerze zewnętrznym. Na samym dole aplikacji jest pasek, na którym wyświetlane są zbierane wyniki pomiarów, takie jak: całościowa droga liczona w pikselach, całościowy czas liczony w sekundach, całościowa prędkość liczona w pikselach na sekundę oraz drogi i czasy pośrednie między kolejnymi blokami. Na poniższym rysunku (Rysunek 3) przedstawiono przykład trasy wraz z bocznym kreatorem interfejsów, a na kolejnym (Rysunek 4) przykład innej trasy z bocznymi opcjami umożliwiającymi uwidocznienie lub ukrycie tras.



Rysunek 3: Widok interfejsu umieszczonego na płótnie z przeprowadzonymi trzema seriami badań.



Rysunek 4: Widok interfejsu z przeprowadzonymi kilkoma seriami badań.

3.2. Fragment kodu aplikacji

Listing 1: Funkcja umożliwiająca rysowanie ruchu kursora oraz obliczenie szukanych wartości.

```
// wykrycie ruchu kursora
$(document).mousemove(function(event) {
// zapisanie aktualnego czasu
t = window.performance.now();
// jeśli pomiar został rozpoczęty
if(started) {
// ustalenie warunków początkowych
if(currentMousePos.x == -1 && currentMousePos.y == -1) {
currentMousePos.x = event.pageX;
currentMousePos.y = event.pageY;
temptime = window.performance.now();
loopI = 0;
}
// wyliczenie różnicy czasowej pomiędzy ostatnim wykonaniem
funkcji, a tym obecnym
deltaT = Math.abs(t - temptime);
// nadpisanie nowo uzyskanym czasem poprzedni czas
```

```
temptime = t;
// obliczenie przebytej drogi od ostatniego wykonania funkcji z wyko-
rzystaniem twierdzenia Pitagorasa
var deltaS = Math.sqrt(Math.pow(currentMousePos.x -
event.pageX,2) + Math.pow(currentMousePos.y - event.pageY,2));
// dodanie drogi składowej do drogi całkowitej
s += deltaS;
sx += Math.abs(currentMousePos.x - event.pageX);
sy += Math.abs(currentMousePos.y - event.pageY);
// nadpisanie aktualnymi wartościami zmiennych przechowujących
starą pozycję kursora
currentMousePos.x = event.pageX;
currentMousePos.y = event.pageY;
// wyświetlenie wartości na ekranie w ustalonych komórkach
$("#droga").text(Math.round((s + 0.00001) * 100) / 100);
$("#drogax").text(sx);
$("#drogay").text(sy);
// wyliczenie szybkości tymczasowej
vtemp = deltaS / deltaT * 1000;
// wyświetlenie wartości na ekranie w odpowiednich polach
$("#predkosc").text(vtemp);
$("#deltaT").text(deltaT);
// dodanie odczytów do tabeli przechowującej dane o ruchu kursora
readings[readings.length-1].push([loopI, currentMousePos.x,
currentMousePos.y, deltaT, window.innerWidth,
window.innerHeight]);
// rysowanie linii na płaszczyźnie canvas
prevX = currX;
prevY = currY;
currX = event.clientX - canvas.offsetLeft + $(document).scrollLeft();
currY = event.clientY - canvas.offsetTop + $(document).scrollTop();
draw();
// zwiększenie licznika wywołań funkcji
loopI++;
}
});
```

3.3. Zastosowane kontrolery ruchowe

Poza środowiskiem testowym, potrzebne są manipulatory, które umożliwiają poruszanie i wykonywanie akcji kursorem systemowym. Na potrzeby artykułu test został przeprowadzony na wybranych urządzeniach o różnej popularności oraz budowie interfejsu wejścia:

a) myszka komputerowa Logitech G403,
b) touchpad zintegrowany z Dell Latitude E6530,
c) manipulator punktowy z Lenovo ThinkPad T590,
d) joystick Thrustmaster T.16000M FCS
e) pilot Magic AN-MR500G do LG Smart TV,
f) bezprzewodowy kontroler Xbox One,
g) tablet Galaxy Tab S5e z ekranem sAMOLED,
h) sensor ruchu Kinect Xbox One,
i) kontroler Leap Motion,
j) 5DT Data Glove Ultra (z 5 czujnikami) [10].

Przykłady wybranych kontrolerów:

Rysunek 5: Przykład kontrolera Microsoft Xbox One [7]

Rysunek 6: Przykład aplikacji ukazująca model dłoni odczytany przez kontroler Leap Motion [6].

Rysunek 7: Przykład rękawic 5DT Data Glove Ultra [1].

Aby dało się porównać wyniki test został przeprowadzany na jednakowych wyświetlaczach. Użytą rozdzielczością było 1080p, czyli 1920x1080px. Większość urządzeń była testowana w systemie operacyjnym Windows 10. Część z urządzeń potrzebowała instalacji odpowiedniego sterownika (z Internetu, najczęściej ze stron producentów testowanych kontrolerów). W przypadku rękawicy 5DT konieczna była modyfikacja istniejącego na rynku sterownika, a Kinect wymagał podłączenia dodatkowego adaptera.

4. Badanie interfejsów urządzeń

4.1 Grupa badawcza

Pierwszym etapem badań, jeśli wszystko działa pod względem technologicznym, jest zebranie grupy badawczej. Istnieją wymogi co do takiej grupy, muszą być to osoby zorientowane w korzystaniu z manipulatorów, ale i takie, które nie korzystają z nich na tyle często, aby zaburzyć wyniki (obawa o zbyt dobre wyniki przy dużym doświadczeniu). W związku z tym została przeprowadzona ankieta, która pozwoliła wyłonić grupę badawczą. W tego typu badaniach zakłada się, że osoby biorące udział w teście podczas wykonywania zadań są w dobrej kondycji psychicznej i są wyspane. Testy muszą być przeprowadzane w przygotowanym wcze-

śniej laboratorium, aby było możliwe sprawne ich przeprowadzenie bez obawy o awarię, jak i w celu zapewnienia maksymalnego komfortu dla uczestników badania. Po wykonaniu testu niezbędne jest zebranie nie tylko wyników liczbowych, ale również przeprowadzenie wywiadu, który pozwoli ustalić stopień zadowolenia użytkowników w kontekście całego testu i poszczególnych urządzeń oraz na zebranie opinii tych użytkowników.

4.2. Charakterystyka obiektu badań

Założeniem aplikacji, która posłużyła jako środowisko testowe w badaniu, było zebranie danych liczbowych, które można opisać w następujący sposób:

x - liczba składowych tras (pomiędzy kolejnymi obiektami), gdzie $x \in \mathbb{C}$

C - zbiór liczb całkowitych

y - liczba bloków wykorzystanych w badaniu (gdzie y_s to liczba bloków startowych, y_p to liczba bloków pośrednich, a y_k to liczba bloków końcowych), z założeniem że $y \in \mathbb{C}$, a więc:

$$y = y_s + y_p + y_k \tag{2}$$

s[px] - droga liczona w pikselach jaką pokonuje kursor pomiędzy blokiem startowym, a blokiem końcowym

s_{px}[px] - pośrednie drogi liczone w pikselach pomiędzy wszystkimi blokami testu (w aplikacji nie jest wyszczególniona ostatnia suma wartości tras pośrednich, ponieważ przyjmuje ona tę samą wartość co droga całkowita, czyli wartość s[px])

t[s] - całościowy czas wykonania zadania testowego liczony w sekundach

t_{px}[s] - czasy pośrednie liczone w sekundach w sposób analogiczny jak składowe tras s_{px}

v_{sr}[px/s] - średnia prędkość kursora podczas realizacji testu liczona w pikselach na sekundę

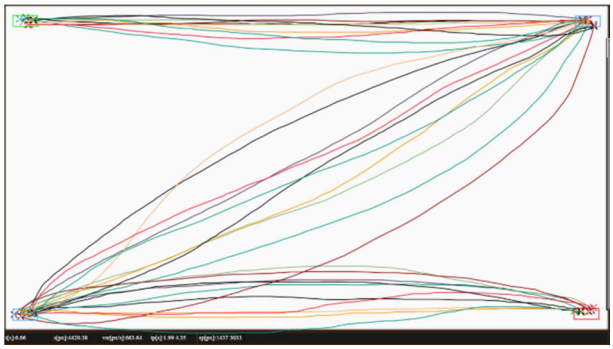
seria z - liczba serii testowych jakie wykonuje użytkownik podczas badania, gdzie pierwsza seria przyjmuje wartość 1, a ostatnia seria wartość z, z założeniem że $z \in \mathbb{C}$

W celu oszacowania możliwej długości trwania testu przy pomocy użytych w badaniu kontrolerów, pomocnicze będzie wykorzystanie wspomnianego w rozdziale 2.1 prawa Fittsa pozwalającego obliczyć czas niezbędny do dotarcia do obiektu graficznego (w tym wypadku dotarcia kursorem od punktu początkowego do środka bloku).

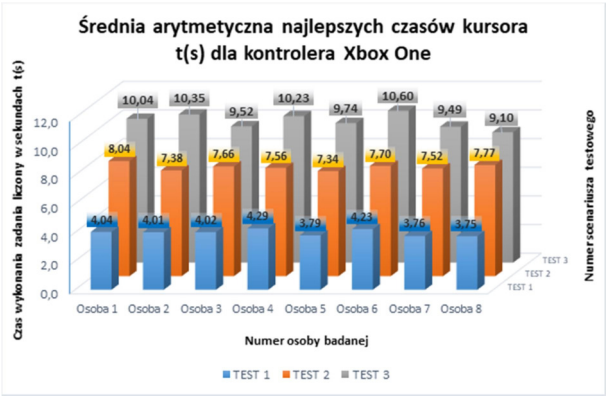
4.3. Rezultat badań

W rozdziale 3.3 wskazano 10 urządzeń, które zostały wykorzystane w badaniu. Samo badanie zostało przeprowadzone na 3 interfejsach testowych o zmiennym stopniu trudności. W celu uzyskania wiarygodnych wyników zostało wykonanych po 10 serii testowych, jednak tylko 5 z nich, które uzyskały najlepsze czasy, było poddanych dalszej analizie. Innymi słowy w całym badaniu każdy z uczestników wykonał 300 serii testowych, ale tylko połowa z nich została użyta do dalszej analizy. Wyniki zostały zaprezentowane przy pomocy tabel (Tabela 2 - 5) oraz wykresów (Rysunek 8 - 10).

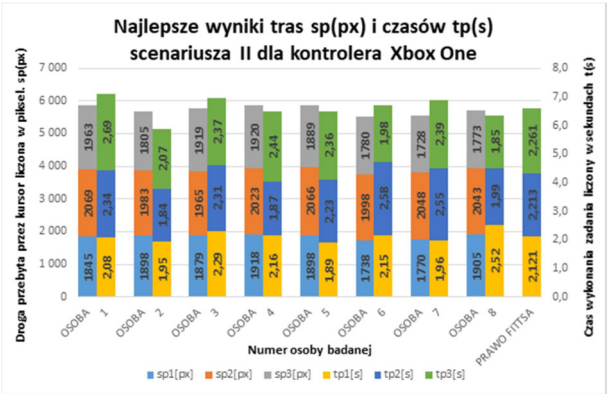
Ze względu na obszerność analizy każdego z kontrolerów podczas badania, w artykule została zaprezentowana jedynie część wyników dotyczących kontrolera Microsoft Xbox One. Pozostałe manipulatory wymienione w rozdziale 3.3 zostały poddane analogicznym badaniom.



Rysunek 8: Przykład wykonanego scenariusza testowego przy użyciu kontrolera Xbox One.



Rysunek 9: Wykres obrazujący średnie arytmetyczne dla 5 najlepszych czasów t(s) uzyskanych przez 8 testerów w 3 scenariuszach testowych przy użyciu kontrolera Xbox One.



Rysunek 10: Wykres obrazujący najlepsze wyniki tras sp(px) i czasów tp(s) uzyskane przez 8 testerów w II scenariuszu testowym przy użyciu kontrolera Xbox One wraz ze spodziewanymi czasami T(s) wyznaczonymi Prawem Fittsa.

Tabela 2: Zestawienie 5 najlepszych czasów t(s) uzyskanych przez 8 testerów w 3 scenariuszach testowych przy użyciu kontrolera Xbox One.

		UCZESTNICY BADANIA								
		1	2	3	4	5	6	7	8	ŚREDNIA
Scenariusz testowy 1: czas wykonania zadania w sekundach t(s)	SERIA 1	4,16	3,58	4,47	4,12	3,49	4,27	3,46	4,01	
	SERIA 2	4,06	4,12	4,24	4,28	3,96	4,78	3,97	3,08	
	SERIA 3	4,15	3,82	4,13	4,65	3,81	4,25	3,86	3,66	
	SERIA 4	4,10	4,25	3,60	3,66	3,82	4,15	3,73	3,96	
	SERIA 5	3,72	4,30	3,68	4,72	3,89	3,72	3,76	4,02	
	ŚREDNIA	4,04	4,01	4,02	4,29	3,79	4,23	3,76	3,75	3,99
Scenariusz testowy 2: czas wykonania zadania w sekundach t(s)	SERIA 1	8,31	7,72	7,28	8,37	7,76	8,11	6,90	7,19	
	SERIA 2	7,11	7,97	6,97	6,47	7,49	6,71	8,16	8,04	
	SERIA 3	8,75	5,86	7,08	8,07	7,85	7,79	7,34	8,25	
	SERIA 4	8,13	7,85	8,25	7,38	7,10	7,71	7,74	8,99	
	SERIA 5	7,88	7,52	8,73	7,53	6,48	8,16	7,47	6,36	
	ŚREDNIA	8,04	7,38	7,66	7,56	7,34	7,70	7,52	7,77	7,62
Scenariusz testowy 3: czas wykonania zadania w sekundach t(s)	SERIA 1	9,50	9,80	10,56	8,56	9,20	11,38	9,54	8,83	
	SERIA 2	10,39	10,93	10,34	10,99	9,71	9,04	9,90	9,73	
	SERIA 3	10,50	10,98	10,15	10,55	9,78	10,43	9,96	9,47	
	SERIA 4	9,60	10,10	8,02	10,01	10,29	11,64	8,34	9,30	
	SERIA 5	10,23	9,95	8,54	11,04	9,73	10,52	9,73	8,18	
	ŚREDNIA	10,04	10,35	9,52	10,23	9,74	10,60	9,49	9,10	9,89

	<----	najkrótszy czas		<----	najlepszy uśredniony wynik
	<----	najdłuższy czas		<----	najgorszy uśredniony wynik

Tabela 3: Zestawienie parametrów potrzebnych do wyznaczenia czasu T(s) wymaganego do realizacji dosięgnięcia obiektu docelowego przy użyciu kontrolera Xbox One oparte o Prawo Fittsa.

PRAWO FITTSA			T = a + b * log2 (D/S + 1)				
TEST 1	D1	915	DLA KONTROLERA XBOX ONE:	a1	0,61	<----	opóźnienie dla trasy nr 1
	D2	950		a2	0,13	<----	opóźnienie dla trasy nr 2
	S1	100		b	0,35	<----	stała czasowa opisująca prędkość ruchu kontrolera
	S2	40					
TEST 2	D1	1790	DLA KONTROLERA XBOX ONE:	a1	0,39	<----	opóźnienie dla trasy nr 1
	D2	1950		a2	0,44	<----	opóźnienie dla trasy nr 2
	D3	1790		a3	0,53	<----	opóźnienie dla trasy nr 3
	S1	60		b	0,35	<----	stała czasowa opisująca prędkość ruchu kontrolera
	S2	60					
	S3	60					
TEST 3	D1	1720	DLA KONTROLERA XBOX ONE:	a1	0,37	<----	opóźnienie dla trasy nr 1
	D2	955		a2	0,77	<----	opóźnienie dla trasy nr 2
	D3	870		a3	0,07	<----	opóźnienie dla trasy nr 3
	D4	1755		a4	1,53	<----	opóźnienie dla trasy nr 4
	S1	25		b	0,32	<----	stała czasowa opisująca prędkość ruchu kontrolera
	S2	100					
	S3	25	Dx – odległość liczona w pikselach od punktu początkowego do środka obiektu				
	S4	95	Sx – szerokość obiektu docelowego liczona w pikselach				

Tabela 4: Zestawienie najlepszych wyników tras sp(px) i czasów tp(s) uzyskanych przez 8 testerów w 3 scenariuszach testowych przy użyciu kontrolera Xbox One wraz ze spodziewanymi czasami T(s) wyznaczonymi Prawem Fittsa.

		UCZESTNICY BADANIA								ŚREDNIA NAJLEPSZYCH WYNIKÓW
		1	2	3	4	5	6	7	8	
SCENARIUSZ TESTOWY 1 (NAJLEPSZA SERIA)	sp1[px]	1065	1122	1248	1215	1001	890	973	995	
	sp2[px]	1063	1082	1246	1106	1040	1070	1047	1012	
	tp1[s]	1,89	2,08	1,65	1,89	1,85	2,01	1,67	1,23	1,784
	tp2[s]	1,83	1,50	1,95	1,77	1,64	1,71	1,79	1,85	1,755
	T1[s]	1,780								
	T2[s]	1,750								
SCENARIUSZ TESTOWY 2 (NAJLEPSZA SERIA)	sp1[px]	1845	1898	1879	1918	1898	1738	1770	1905	
	sp2[px]	2069	1983	1965	2023	2066	1998	2048	2043	
	sp3[px]	1963	1805	1919	1920	1889	1780	1728	1773	
	tp1[s]	2,08	1,95	2,29	2,16	1,89	2,15	1,96	2,52	2,125
	tp2[s]	2,34	1,84	2,31	1,87	2,23	2,58	2,55	1,99	2,214
	tp3[s]	2,69	2,07	2,37	2,44	2,36	1,98	2,39	1,85	2,269
	T1[s]	2,121								
	T2[s]	2,213								
	T3[s]	2,261								
SCENARIUSZ TESTOWY 3 (NAJLEPSZA SERIA)	sp1[px]	1723	1950	1886	1856	1760	2000	1711	1860	
	sp2[px]	974	954	1000	1058	1018	1065	985	930	
	sp3[px]	1309	956	850	853	885	821	930	1061	
	sp4[px]	1911	2403	1821	1765	2180	1836	1745	1920	
	tp1[s]	2,36	2,47	2,01	2,22	2,63	2,72	2,17	2,11	2,336
	tp2[s]	1,90	1,83	1,61	2,07	1,64	2,22	1,77	1,82	1,858
	tp3[s]	1,83	1,59	1,76	1,40	1,95	1,71	1,81	1,76	1,726
	tp4[s]	3,41	3,91	2,64	2,87	2,98	2,39	2,59	2,49	2,910
	T1[s]	2,330								
	T2[s]	1,858								
	T3[s]	1,722								
	T4[s]	2,901								

	<----	najlepszy wynik dla danego etapu		<----	najgorszy wynik dla danego etapu

5. Wnioski

Na podstawie przeprowadzonych badań udało się zebrać liczne wyniki liczbowe oraz uzyskać ich wizualizację w tabelach i na wykresach. Po badaniu przeprowadzono wywiad z uczestnikami badania, który uzupełnił suche liczby o odczucia użytkowników, co pozwoliło uzyskać ciekawe wnioski.

Badanymi wartościami była droga kursora przebyta w pikselach, czas liczony w sekundach, prędkość kursora liczona w pikselach na sekundę oraz składowe tych wartości na pośrednich trasach. Pierwszym ciekawym wnioskiem dotyczącym wszystkich kontrolerów jest to, że najlepsze uzyskane czasy najczęściej nie pokrywały się z najkrótszą pokonaną trasą. Okazuje się, że precyzja ruchu kursora nie jest wprost proporcjonalna do czasu wykonania zadania ekranowego. Użytkownicy nie poruszali się po liniach prostych, a po krzywych przypominających łuki oraz fale. Prędkość ruchu kursora miała natomiast kluczowe znaczenie, a ta osiągała najwyższe wyniki przy średniej precyzji ruchu.

Zgodnie z założeniami HCI okazało się, że dopiero zestawienie wyników liczbowych z wywiadem z grupą badawczą pokazało pełny obraz badania. Przechodząc do omówienia wyników liderem pod względem uzyskanych czasów, nie licząc ekranu dotykowego w trybie klikania, okazała się najpopularniejsza mysz komputerowa, ale w opinii użytkowników nie zawsze mogłaby nim być. Uczestnicy badania stwierdzili, że swój wynik myszka osiągnęła tylko dzięki warunkom sprzyjającym poruszaniu się tym kontrolerem, gdyż badanie było wykonywane przy biurku, a mysz komputerowa wymaga płaskiego podłoża. Myszka nie należy też do najbardziej precyzyjnych kontrolerów pod względem poruszania się po liniach prostych.

Analizując kolejne kontrolery o lekko gorszych czasach, ale podobnej specyfice, touchpad umożliwia pracę bez podłoża, a manipulator punktowy taki jak trackpoint pracę w warunkach lekkiej wilgoci lub na mrozie. Joystick uzyskał średnie czasy w badaniu, ale pozwolił na wysoką precyzję ruchu. Nieco podobny kontroler Xbox One pomimo niewiele lepszych wyników czasowych był oceniony jako najprzyjemniejszy w użyciu ze względu na intuicyjny interfejs i wysoką ergonomiczność, gdyż można z niego korzystać w dowolnej pozycji ciała (np. na leżąco).

Ekran dotykowy przetestowano w trybie rysowania drogi oraz klikania bloków pośrednich. Jak było wspomniane czasy w trybie klikania deklasują nawet mysz komputerową, natomiast tryb rysowania trasy okazał się bardzo przyjemnym doświadczeniem, chociaż uzyskane wyniki czasowe były w nim średnie. Użytkownicy jednak czuli się bardzo komfortowo podczas tego badania na co wpływa zapewne obecna popularyzacja urządzeń dotykowych.

Kolejne kontrolery dały uczestnikom, pomimo uzyskania miernych czasów badania, ogromną satysfakcję i zabawę przy wykonywaniu testów. Kinect pozwolił na niezwykłe doświadczenie, gdyż wbudowana kamera precyzyjnie wykrywała dłonie użytkowników, a ruchy kursorem oraz kliknięcia były równie dokładne. Pilot Magic dał za to doświadczenie w korzystaniu z telewizora jak z komputera. W Leap Motion chociaż sposób poruszania kursorem bardzo zaciekawił uczestników badania, to złe rozpoznawanie gestów i męczące trzymanie rąk nad urządzeniem szybko zniechęciło z jego używania.

Na koniec zbadano rękawicę 5DT, która uzyskała najgorsze czasy, ale pozwoliła na wysoką precyzję ruchu. W opinii użytkowników, gdyby dopracować kwestię kalibracji, rękawica mogłaby się świetnie nadać do zdalnego sterowania maszynami takimi jak dźwigi.

Podsumowując kontrast między wynikami liczbowymi, a odczuciami po badaniu były zaskoczeniem zarówno dla samych uczestników badania jak i jego autorów. Autorzy spodziewali się, że myszka komputerowa nie zostanie liderem pod względem uzyskanych czasów, ale może okazać się ulubionym kontrolerem grupy badawczej, a jest zupełnie odwrotnie.

Niestety autorzy nie przewidzieli, że w przypadku środowiska testowego nie należy zadbać jedynie o aplikację do testowania, ale również o warunki otoczenia. W teście przy biurku to myszka jest liderem pod względem czasów, ale bez płaskiego podłoża zapewne by przegrała, dlatego należałoby powtórzyć badanie w kilku odmiennych warunkach otoczenia. Na szczęście same testy użyteczności wykazały już, że inne kontrolery w pewnych sytuacjach są o wiele lepsze od myszy komputerowej, co autorzy artykułu uważają za sukces. Zostaje jednak niedosyt i rekomendacja przeprowadzenia badań na podobnych zasadach, ale w odmiennych warunkach otoczenia.

Literatura

[1] 5DT Data Glove Ultra Controller, http://5dt.com/wp-content/uploads/2011/06/hw_data_glove_wireless01.jpg, [24.02.2020].

[2] T. Atkins Jr. i inni, CSS Snapshot 2017, https://www.w3.org/TR/css-2017/, [24.02.2020].

[3] D. Benyon, Designing Interactive Systems - A Comprehensive Guide to HCI, UX and Interaction Design, Pearson ELT (2014).

[4] I. Hickson i inni, A vocabulary and associated APIs for HTML and XHTML, https://www.w3.org/TR/html5/, [24.02.2020].

[5] S. Krug, Nie każ mi myśleć! O życiowym podejściu do funkcjonalności stron internetowych, Helion (2014).

[6] Leap Motion Controller, https://3bonlp1aiidtbao4s10xacvn-wpengine.netdna-ssl.com/wp-content/uploads/2017/03/leap-motion-3d-motion-gesture-controller-10-large.jpg, [24.02.2020].

[7] Microsoft Xbox One Controller, https://steamcdn-a.akamaihd.net/steamcommunity/public/images/steamworks_docs/polish/xboxone_controller.png, [24.02.2020].

[8] M. Miłosz, Ergonomia systemów informatycznych, Politechnika Lubelska, 2014.

[9] M. Sikorski, Interakcja Człowiek-Komputer, Polsko-Japońska Wyższa Szkoła Technik Komputerowych, 2010.

[10] Wikipedia. Wolna encyklopedia, http://www.wikipedia.org, [24.02.2020].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 30–36
Received: 27 November 2020
Accepted: 12 February 2021

Comparative analysis of tools for the integration of IT systems

Narzędzia do integracji systemów informatycznych - analiza porównawcza

Vladyslav Shkuta*, Marek Miłosz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This article describes the methods, approaches and technologies of information systems integration, compares the functionality of the selected tools for information systems integration from a selected point of view. In this case, it is an evaluation of the ability to perform the main functions of the Enterprice Service Bus and the resulting benefits.

Keywords: integration; backbone; function; comparison

Streszczenie
Ten artykuł opisuje metody, podejścia i technologie integracji systemów informatycznych, porównuje funkcjonalność wybranych narzędzi do integracji systemów informatycznych z wybranego punktu widzenia. W tym przypadku jest to ocena zdolności do pełnienia głównych funkcji Korporacyjnej Magistrali Usług i wynikających z tego korzyści.

Słowa kluczowe: integracja; magistrala; funkcja; porównanie

*Corresponding author
Email address: **vladyslav.shkuta@pollub.edu.pl** (V. Shkuta)

1. Introduction

System integration (SI) is an IT or engineering process or phase concerned with joining different subsystems or components as one large system. It ensures that each integrated subsystem functions as required [1].

SI is also used to add value to a system through new functionalities provided by connecting functions of different systems [1].

When integrating systems, any objects can be involved, for example: computer networks, various kinds of equipment, integration software, host applications, user screens, ERP-, CRM-, SCM-applications, distributed objects, legacy systems, as well as custom or proprietary applications [2] and many others.

2. Methods of system integration

On the one hand, the integration mechanism can be represented by synchronous communication methods with blocking pending a response, based either on remote procedure call (RPC), or on its variations. Direct communication methods based on RPC (such as CORBA, DCOM, RMI) provide tight coupling and require intrusion in the parts to be integrated and also provide point-to-point communication. They may have the advantage of using a binary data exchange format with minimal latency, but the downside of this is the lack of Internet support. Web services, based on RPC, solve the web support problem, but the XML format they use is slower.

Another communication capability is provided by message queues controlled by queue managers. Such a connection is reliable and guarantees the delivery of messages to the recipient under any conditions, even if he is not online, waiting for him and storing the messages until he appears. This method is asynchronous, does not block actions, and allows for loosely coupled systems; however, it is possible to implement synchronous communication provided by two queues (request and response).

Integration via files and databases is used in cases where other methods are not available. This type of integration can be applied to older systems that cannot be changed.

In UI level integration, two applications are integrated in such a way that a user can perform an operation that includes two different applications - without having to take into account that he is actually launching two applications.

In terms of enterprise integration, enterprise application integration (EAI or A2A) enables the exchange of data and business processes between any connected applications or data sources in the enterprise (in particular, ERP-, CRM-, SCM-applications). The connection between pairs of applications can of course also be direct (so-called point-to-point), for example, using protocol and / or format adapters at one or both ends (using technological APIs such as FTP, IIOP, RPC, remote or batch interfaces), but this is labor-intensive and may require intrusion in these applications. An easier way is to use middleware to minimize interference and ensure reliable data transfer.

It is best of course to use integration tools, because they provide a wide range of capabilities: adapters, message queues, guaranteed message delivery when needed, brokers, business process modeling, application deployment, composite applications, routing, format conversion, security, administration and monitoring capabilities. and many others. They allow to build a service-oriented architecture (using web services), to represent applications, systems, functions as reusable services located on a single backbone, standardizing formats and reducing the number of links (in comparison with a point-to-point topology). The capabilities

depend on the tool (be it a message broker, ESB, application server, or even a BPM tool) and what the manufacturer has put into it, and the tool must be used depending on the needs, because for a small integration (up to three applications) a simple point-to-point solution can be enough (without the need for special tools that only complicate the situation).

B2B integration requires standardization of formats and coordination of partners, because different companies can use different technologies and solutions, therefore consistency is necessary, especially for financial institutions and international partners.

B2C integration is related to e-commerce, which in turn depends on Internet technology. Organizing this type of trading also requires standards and organizational methods that enable the discovery of services provided and the exchange of information in real time and with high reliability. A service-oriented architecture (SOA) with SOAP, WSDL and UDDI formats and the WS-Security specification can help in it.

3. Brief overview of selected tools for IT systems integration

From a variety of integration software, the following products were selected for comparison:

- OpenESB (Glassfish) – an open source and developed by the community (and previously Sun Microsystems) ESB;
- MuleESB – one of the most popular ESBs;
- Microsoft BizTalk Application Server – Application Server, which has become an example to follow.

OpenESB is a distributed integration middleware infrastructure (a Java-based open source enterprise service bus) providing Web and non-Web services support, transformation, routing, and orchestration [3].

OpenESB is based on various industry standards such as JBI, WSDL, XML and Java. It enables connecting to various heterogeneous systems, gets them together to exchange messages in a standard way and collaborate with each other seamlessly. These standards help in shortening the learning curve, faster development and ease of deployment and managing [3].

OpenESB consists of 5 parts: the framework, the container, the components, the Integrated Development Environment and the development plugins [4].

The framework consists of a lightweight implementation of JBI (Java Business Integration) in Java. This implementation is container-independent and can run on any platform and any container. In addition to being lightweight, the OpenESB framework is also reliable and scales well. The framework implements a virtual bus known as the Normalized Message Router (NMR). This is a powerful asynchronous intelligent communication channel between components [4].

In comparison will participate the Glassfish-based version of OpenESB.

The JBI specification defines 2 component types: The services engine (SE) and the binding component (BC). The SE and BC implement the same interface contract; however, they behave differently [4]:

- Binding components act as the interface between the outside world and the bus, being able to generate bus messages upon receipt of stimuli from an external source or generate an external action/interaction in response to a message received from the bus.
- Service engines receive messages from the bus and send messages to the bus. SE's have no direct contact with the outside world. They rely on the bus for interaction with other components, whether binding components or other service engines.

OpenESB offers a set of graphical tools to ease complex SOA and integration developments. XLM, XML Schema, WSDL, BPEL editor, data mapping and Composition Applications graphical editors are proposed with OpenESB. Similarly, build, deploy, undeploy, run, test and debug tasks are managed by graphical tools. OpenESB provides the best ergonomics for ESB and SOA developments [4].

MuleESB is a scalable and distributed object manager from Mulesoft that can handle interactions with services and applications that support a variety of transport and communication technologies.

MuleESB was designed to be lightweight and easy to be embedded into Java applications and application servers, or to act as an independent server. It integrates with a number of frameworks such as Spring, Hivemind and Plexus and supports many transport and service components, including [5, 6]:

- JMS (Java Message Service);
- SOAP (Simple Object Access);
- REST (Representational State Transfer);
- MQ (Message Queuing);
- AQ (Oracle Advanced Queuing);
- Caching (Caching);
- JavaSpaces (a service providing a distributed mechanism for exchanging and coordinating Java objects);
- GigaSpaces;
- IM;
- JCA (J2EE Connector Architecture);
- Data Queues;
- System I / O (System Input-Output);
- JBI (Java Business Integration);
- BPEL (Business Process Execution Language);
- EJB (Enterprise Java Beans);
- AS / 400 (Application System / 400, IBM System i);
- HTTP (HyperText Transfer Protocol);
- JDBC (Java DataBase Connectivity);
- TCP (Transmission Control Protocol);
- UDP (User Datagram Protocol);
- SMTP (Simple Mail Transfer Protocol);
- File (FILE);
- FTP (File Transfer Protocol);
- much more.

The platform is Java oriented but can be a mediator for other platforms such as .NET using web services or sockets [6].

MS BizTalk Application Server is exceptionally well-suited to more complex IT landscapes where heavy integration demands [7]:

- Enterprise Application Integration (EAI): Is there a necessity to connect ERP system to a CRM system? Internally or externally? With BizTalk Server it becomes a whole lot easier, thanks to support for standards (JSON, XML, FlatFile) and an extended collection of adaptors that are provided for all kinds of packages and systems: SAP, Oracle, Siebel, MS SharePoint, IBM DB2, AS400.
- Business Process Management (BPM): thanks to integration, a whole slew of business processes can be simplified, which in turn leads to a shorter time-to-market and therefore cost savings. With transparent IT monitoring thanks to the BizTalk Server, it is known from the top what is really happening within an organization and which systems are linked together.
- Business-to-Business integration (B2B): BizTalk Server comes as standard with support for typical integration needs within so-called "industry verticals" such as Healthcare, Transport and Financials, because it supports a range of protocols and report formats like HL7, SWIFT, X12 and EDIFACT.
- Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB): with BizTalk Server one can be adding a versatile platform to the business that one can expand internally to meet the needs using an Enterprise Service Bus (ESB), thanks to the flexibility and scalability of the .NET framework. Given that BizTalk Service includes full support for WCF (SOAP) and Web API (REST) it can also be used as a hosting platform for the services within SOA architecture.
- High Availability and Scalability: thanks to its robust underpinning by Windows and SQL Server Clustering, the BizTalk Server is a platform that provides total support for roll-out in a multi-node architecture, possibly across geographically discrete datacenters, providing support for critical processes of the business.

4. Tools for IT systems integration comparison methodology

The main goal of the article is to evaluate the ability of selected software products for integration of IT systems (integration tools) to provide their main integration functionality, that is the functionality of the integration backbone.

The research will consist of several stages.

The goal of the first stage is to identify the ability of each selected product (integration tool) to provide certain functional capabilities. There are such groups of functional capabilities as the ESB functions described by Falko Menge in [8]. The ESB functions are placed and explained in Table 1.

Table 1: ESB Functions.

Function	Description
Invocation (call)	The ability of the product to send requests and receive responses from integration services and integrated resources.
Routing	The ability to determine the destination of messages during transport.
Mediation	The ability to transform or move non-equivalent resources in message transports.
Adaptability	The ability to adapt to other service providers.
Security	The ability to provide secure messaging at all points of message transport.
Administration	The ability to provide monitoring, logging, auditing, and integration scenarios.
Process Orchestration	The ability to perform complex business processes described in the standard language (BPEL).
Complex Events Processing	The ability to interpret events and the existing relationship between them for further guidance.
Development Tool	The ability to provide tools for designing, developing, testing, and deploying applications for professional development.

The functions presented in Table 1 are divided into functional capabilities.

At the first stage, the functional capabilities will be used as criteria for analyzing the ability of the product to perform the function to which they belong. The products, in turn, are presented in section 3 "Brief overview of selected tools for IT systems integration", the functional capabilities will be described in the analysis process for each function.

Table 2 shows the scale of possible ratings to assign to the product for each criterion. The evaluation of support for each capability is determined by its ability to perform, completeness of vision, standards monitoring, and ease of realization.

Table 2: The values of characteristics evaluation.

Rating	Characteristic
1	The product initially offers functional capabilities or offers guidance on how to achieve them using certain abilities.
0.75	The product offers functional capabilities, but adapters may be required to meet the requirements in complete.
0.5	The product provides only a manual or requires the use of adapters to achieve some of the required functionality.
0	The product does not offer any documentation or functionality to meet the requirement.

The minimum score is 0, which equals 0% of the ability, and the maximum score is 1, which equals 100%.

The average of ratings for all product's criteria determines its functional ability.

The goal of the second stage is to identify the overall ability of products to match the reference (benchmark) product of the Total compliance with the enterprise integration backbone class tool.

The reference product is an integration tool that can perform 100% of the Total compliance with the enterprise integration backbone functions. The more percentages the tool scores, the closer it is to the reference

product. This means that such a tool can be used in more complex scenarios and for broader integration needs and is perfectly suited for enterprise environments and developments.

At this stage, the criteria will be ESB functions, and the evaluation of the ability of products to meet the reference product of the Total compliance with the enterprise integration backbone class tool will be determined by the arithmetic mean between the ratings for all the criteria for the product.

5. Comparison of selected tools for IT systems integration

Based on the methodology described in section 4 "Tools for IT systems integration comparison methodology", OpenESB, MuleESB, and Microsoft BizTalk are compared.

First, a description of the function and its capabilities is provided, and then is provided a comparison.

General evaluation of all functions and average of ESB functions capabilities for every tool is in the end.

Invocation (call) is the ability of a product to send requests and receive responses from integration services and integrated resources. It includes the following capabilities:

- Asynchronous call. These are non-blocking actions that allow the main program to continue processing.
- Synchronous messaging. Ability to simulate synchronous communication when a program call waits for a result before continuing processing.
- Translation of protocols. Ability to transfer from one type of communication Protocol to another (for example, TCP/IP to HTTP).
- FTP support. It is a Protocol commonly used for file exchange over a TCP/IP-based network to manage files on a computer on that network.
- Support for SFTP. FTP Protocol with additional built-in security.
- HTTP support. A communication Protocol used for transmitting information in intranets and WWW. Hypertext transfer protocol.
- Support for POP3. Protocol for sending mail messages to the server on request.
- SMTP support. Simple Protocol for sending mail messages.
- IMAP support. Mail Protocol of the Internet.
- EDI support. Ability to exchange electronic files.
- Support for JMS. Ability to process messages from Java services.

The ability to comply with the call rules is shown in Table 3.

Table 3: Compliance with invocation rules.

	Characteristic	BizTalk	OpenESB	MuleESB
Invocation (call)	Asynchronous invocation	1	1	1
	Synchronous messaging	1	1	1
	Translation of protocols	1	0	0.75
	FTP support	1	1	1
	Support for SFTP	0	0	1
	HTTP support	1	1	1
	Support for POP3	1	0	1
	SMTP support	1	0.5	1
	IMAP support	1	0.5	1
	EDI support	1	1	1
	Support for JMS	0	1	1
	Ability	**0.81**	**0.64**	**0.97**

Routing is the ability to determine the destination of messages during transport. It includes the following capabilities:

- Content-based Routing. Searches for message routing based on the current content of the message itself, rather than on a specific destination.
- Endpoint independence. Localizing a service that doesn't depend on the running application.
- Delivery guarantee. Describes a Protocol that allows messages to be reliably delivered between distributed applications, despite failures of software components, systems, or networks.
- Load balancing. Ability to deploy multiple instances of the service and use a load balancer to send requests and release the required traffic.

The ability to comply with routing rules is shown in Table 4.

Table 4: Compliance with message routing rules.

	Characteristic	BizTalk	OpenESB	MuleESB
Routing	Content-based Routing	1	0.5	1
	Endpoint independence	1	0.5	1
	Delivery guarantee	1	0.5	1
	Load balancing	1	0.5	0.5
	Ability	**1**	**0.5**	**0.87**

Mediation is the ability to transform or move non-equivalent resources in message transports. It includes the following capabilities:

- Message Verification. Validation is a simple check that the incoming message contains a well-formed XML format and matches a specific schema or WSDL document that describes the message.
- Displaying diagrams. A tool for making it easier to display message schemas.
- Routing messages with errors. When a message fails to be received, it is sent to another location to perform additional actions.

- Provision and registration of services. Ability to accompany new services and register them in a configuration-based style.

The ability to comply with mediation rules is shown in Table 5.

Table 5: Compliance with mediation rules.

	Characteristic	BizTalk	OpenESB	MuleESB
Mediation	Message Verification	1	0.75	0.75
	Displaying diagrams	1	0.75	1
	Routing messages with errors	1	0.5	0.5
	Provision and registration of services	1	0	1
	Ability	**1**	**0.66**	**0.81**

Adaptability is the ability to adapt to other service providers. It includes the following capabilities:

- Microsoft SharePoint Services. Integration with Microsoft SharePoint Services.
- Custom adapters. Examples and existing documentation for creating custom adapters.
- MS SQL Server. Integration with MS SQL Server.
- Sybase ASE Server. Integration with the Sybase ASE Server.
- Oracle Databases. Integration with Oracle databases.
- .NET Adapter. Adapter for. NET.
- IBM MQ support. Support for the IBM MQ Series.
- MSMQ support. Integration with the MSMQ queue manager.
- Sockets support. Adaptation to communications via Sockets.
- MS Windows systems. Integration with Microsoft Windows systems.
- Linux/Unix systems. Integration with Linux/Unix systems.
- Other OSs. Integration with other operating systems.

Compliance with the adapter support requirements is shown in Table 6.

Table 6: Compliance with adapter support requirements.

	Characteristic	BizTalk	OpenESB	MuleESB
Adaptability	Microsoft SharePoint Services	1	0	0.75
	Custom adapters	1	0	0.5
	MS SQL Server	1	0.75	0.5
	Sybase ASE Server	1	0.75	0.5
	Oracle Databases	1	1	1
	.NET Adapter	1	0.5	1
	IBM MQ	1	0.5	0.5
	MSMQ support	1	0.5	1
	Sockets support	1	0	0.5
	MS Windows systems	1	1	1
	Linux/Unix systems	0	1	1
	Other OSs	0	1	1
	Ability	**0.83**	**0.58**	**0.77**

Security is understood as the ability to provide secure messaging at all points of message transport. It includes the following capabilities:

- Notification. The ability to report incidents based on certain parameters.
- High availability. Constant availability of the service regardless of the state of the server it is hosted on or the dependent servers it runs on.
- Crash recovery. Ability to restore all resources after a disaster.
- Security in web services. WS-Security describes improvements to SOAP messages to provide quality protection through message integrity, message confidentiality, and single-message authentication.
- Encryption/decryption of content. Support for message content encryption.
- Content authentication and authorization. Authentication and authorization based on the message content.
- Digital signature. The ability to use digital signatures to authorize access.
- Keys syncing. Mechanisms for ensuring key synchronization (if the ESB connects to other systems that can synchronize credentials in multi-transition systems).
- The impossibility of renouncing authorship. The ability to guarantee that the transmitted message was sent and received by the parties claiming to send or receive such a message.
- A Single Login (Single Sign-On). A service that allows administrators to map a Windows or non-Windows user account.

The ability to comply with safety requirements is shown in Table 7.

Table 7: Compliance with security requirements.

	Characteristic	BizTalk	OpenESB	MuleESB
Security	Notification	1	0.75	1
	High availability	1	1	1
	Crash recovery	0.75	0.75	1
	Security in web services	1	0.5	0.75
	Encryption/decryption of content	1	0.5	1
	Content authentication and authorization	1	0.75	1
	Digital signature	1	0	1
	Keys syncing	1	0.5	1
	The impossibility of renouncing authorship	1	0	0
	Single Sign-On	1	0	0.5
	Ability	**0.9**	**0.39**	**0.82**

Administration is the ability to provide monitoring, logging, auditing, and integration scenarios. It includes the following capabilities:

- Prohibit or restrict messages (throttling). A configuration that allows only a certain number of messages

to reach the service before a certain time period expires.
- Logging and auditing. The possibility of logging of messages and ease of implementation of these audit logs.
- Exception log processing. Logging exceptions related to server operations.
- Performance monitoring. A tool for monitoring system performance.
- Compilation of statistical data. Ability to save data for statistics on the use of server services and resources.
- Processing of Poison messages (correction, auto-forwarding). "Poison message" is a message that has exceeded the number of delivery attempts to the target application. This situation can occur when a queue-based application cannot process a message due to errors.
- Integrated development environment (IDE). A project-oriented graphical tool for specific developments.

The ability to comply with administrative requirements is shown in Table 8.

Table 8: Compliance with administrative requirements.

	Characteristic	BizTalk	OpenESB	MuleESB
Administration	Prohibit or restrict messages	0	0	1
	Logging and auditing	1	0.75	1
	Exception log processing	0.75	0.5	1
	Performance monitoring	1	0.75	1
	Compilation of statistical data	0.75	0.75	1
	Processing messages with errors (correction, auto-forwarding)	0.5	0	0.5
	Integrated development environment (IDE)	1	0.5	1
	Ability	**0.71**	**0.46**	**0.92**

Process Orchestration is the ability to perform complex business processes described in the standard language (BPEL). It includes the following capabilities:
- Separation of rules. The ability to provide rules that will then be reused.
- Reuse rules between processes. Ability to reuse the provided rules.
- Dynamic reconfiguration. Dynamically adds a new service producer and consumer to the stage (orchestration).
- Exception handling. Mechanism for handling exceptions that occur during orchestration.
- Long-term transactions. Ability to handle orchestrations that take a long time to complete.
- Generation of web services. Ability to publish or create web services from orchestrations.
- Atomic transactions. Ability to support short-lived centralized operations or processes when a failure occurs and needs to be identified quickly.
- Coordination of web services. An advanced framework that provides protocols that coordinate the actions of distributed applications.
- Extended service support. Ability to interact programmatically with external services. These services are web services published with the ESB.
- Message tracking. A tool for tracking messages as they pass through the service layer.
- Publish and subscribe. Subscriptions to events that occur in orchestrations. Ability to publish events for interested parties.

Compliance with the requirements for process orchestration is shown in Table 9.

Table 9: Compliance with process orchestration requirements.

	Characteristic	BizTalk	OpenESB	MuleESB
Process Orchestration	Separation of rules	1	0	0
	Reuse rules between processes	1	0	0.5
	Dynamic reconfiguration	0	0	0
	Exception handling	1	0.5	0.75
	Long-term transactions	1	0.5	0.75
	Generation of web services	1	1	1
	Atomic transactions	1	0.5	0.5
	Coordination of web services	1	0.75	1
	Extended service support	1	0	1
	Message tracking	1	0.5	0.5
	Publish and subscribe	1	1	1
	Ability	**0.9**	**0.43**	**0.63**

Complex Events Processing is the ability to interpret events and the existing relationship between them for further guidance. It includes the following capabilities:
- Dynamic resource allocation. Dynamic allocation of resources on the server.
- Data protection and cleaning. Data storage mechanisms, as well as a set of parameters for clearing data.
- Publish and subscribe to business events. Subscriptions to events that occur in orchestrations. Ability to post events for interested parties.
- Managing the publication of business events. A tool for managing business event publications.
- Business events subscription management. A tool for managing subscriptions to business events.

Compliance with complex events support requirements is shown in Table 10.

Table 10: Compliance with complex events support requirements.

	Characteristic	BizTalk	OpenESB	MuleESB
Complex Events Processing	Dynamic resource allocation	0	0	0
	Data protection and cleaning	0	0.75	0.5
	Publish and subscribe to business events	1	0.5	1
	Managing the publication of business events	1	0.5	1
	Business events subscription management	1	0.5	1
	Ability	**0.6**	**0.45**	**0.7**

Development Tool includes tools for designing, developing, testing, and deploying applications for professional development. It includes the following capabilities:

- Streaming compilation. A tool for streaming compilation.
- Errors reporting. A tool for reporting and managing flow errors.
- Real-time monitoring of processes. Real-time monitoring of processes.
- Monitoring and debugging threads. A graphical tool that allows to monitor and debug the tech. process.
- Easy application deployment. A tool to help deploy services, trace, and so on.
- Ease of applications migration. A tool for facilitating service migration between endpoints (instances).
- IDE. A tool for developing integrated applications.
- Authorship and defining of business rules. A tool for creating business rules.
- Versions control. Ability to deploy new versions of business rules. The ability to have multiple versions that can be deployed.
- Development and implementation of business rules. Published API for interacting with business rules from external applications.

Compliance with the requirements for development tools is shown in Table 11.

Table 11: Compliance with the requirements for development tools.

	Characteristic	BizTalk	OpenESB	MuleESB
Development Tool	Compilation of streams	1	0.5	1
	Errors reporting	1	0.5	1
	Real-time monitoring of processes	1	0.5	1
	Monitoring and debugging threads	1	0.5	1
	Easy application deployment	1	0.5	1
	Ease of migration of applications	1	0	1
	IDE	1	0.5	1
	Authorship and defining of business rules	0.75	0.5	1
	Versions control	0.5	0.5	0.5
	Development and implementation of business rules	1	0	1
	Ability	**0.92**	**0.4**	**0.95**

To summarize, it is necessary to sum up the evaluation results for each function in a single table. And find the overall compliance of each tool with the enterprise integration backbone (average of ESB functions capabilities).

Compliance with the reference product is indicated in Table 12.

Table 12: General evaluation of product characteristics.

	Characteristic	BizTalk	OpenESB	MuleESB
Functions	Invocation	0.81	0.64	0.97
	Routing	1	0.66	0.81
	Mediation	1	0.66	0.81
	Adaptability	0.83	0.58	0.77
	Security	0.9	0.39	0.82
	Administration	0.71	0.46	0.92
	Process Orchestration	0.9	0.43	0.63
	Complex Events Processing	0.6	0.45	0.7
	Development tool	0.92	0.4	0.95
	Average of ESB functions capabilities	**0.85**	**0.52**	**0.83**

6. Conclusions

Although the evaluation is quite subjective, it gave a General idea about the compared tools and demonstrates the difference that exists between the Microsoft BizTalk Application Server, MuleESB and OpenESB (Glassfish).

The evaluation revealed that Microsoft BizTalk Application Server and MuleESB are tools that can be used in more complex scenarios and for broader integration needs, so they are great for enterprise environments and development.

References

[1] What is System Integration (SI)?, https://www.techopedia.com/definition/9614/system-integration-si, [10.08.2020]

[2] D. S. Linthicum, Enterprise Application Integration, Addison-Wesley, 2000, https://www.academia.edu/8531349/Types_of_EAI_21, [2.09.2020]

[3] Welcome to OpenESB An ESB based on industry standards, https://www.logicoy.com/openesb/, [5.10.2020].

[4] OpenESB, https://en.wikipedia.org/wiki/OpenESB, [27.10.2020].

[5] Mule (software), https://es.wikipedia.org/wiki/Mule_(software), [20.10.2020].

[6] MuleESB, https://ru.wikipedia.org/wiki/MuleESB, [20.10.2020].

[7] Microsoft BizTalk Server Enterprise Integration with Microsoft, https://www.realdolmen.com/en/solution/microsoft-biztalk-server, [27.10.2020].

[8] F. Menge, Enterprise Service Bus. In: FrOSCon: Free and Open Source Software Conference, University of Applied Science Bonn-Rhein Sieg, 2007.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 37–41
Received: 04 December 2020
Accepted: 06 January 2021

Comparison of frameworks for creating web services using the Axis2/C and gSOAP examples

Porównanie technologii tworzenia usług sieciowych na przykładzie Axis/C i gSOAP

Roman Bondarev*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The main purpose of this paper was to compare two frameworks (Axis2/C and gSOAP) that implement the SOAP protocol, a protocol for the exchange of structured messages in a computer environment. The paper describes experiments showing the level of SOAP performance depending on the components of each framework and the methods of their implementation, such as: parsing, serialisation, working with various message formats, the time effectiveness of protocol, the efficiency of message processing models, etc. After the practical part was completed, performance studies of each of the frameworks were carried out, the collected data helped to define a more efficient and faster tool for transferring data between the server and the client.

Keywords: web-services; serialisation; deserialisation; SOAP

Streszczenie
Głównym celem tego artykułu było porównanie dwóch frameworków (Axis2/C and gSOAP), które implementują protokół SOAP do wymiany ustrukturyzowanych komunikatów w środowisku komputerowym. W artykule opisano eksperymenty badające wydajność SOAP dla wskazanych frameworków oraz metody ich implementacji, takie jak: parsowanie, serializacja, praca z różnymi formatami wiadomości, efektywność czasowa protokołu, efektywność modeli przetwarzania wiadomości itp. Do wykonania badań przygotowano aplikacje testowe, które umożliwiły porównanie wydajności każdego z frameworków. Analiza zebranych danych pozwoliła wskazać wydajniejsze i szybsze narzędzie do przesyłania danych pomiędzy serwerem a klientem, którym okazał się gSOAP.

Słowa kluczowe: web-serwisy; serializacja; deserializacja; SOAP

*Corresponding author
Email address: **roman.bondarev@pollub.edu.pl** (R. Bondarev)

1. Wstęp

W systemach informatycznych przez wiele lat istniał wymóg komunikowania się przez Internet, niezależnie od platformy, języka, lokalizacji lub technologii, w której zostały tworzone aplikacje. Rozwój XML, powszechnego formatu wymiany danych oraz akceptacja architektury zorientowanej na usługi, ze strony największych liderów rynkowych, sprawiły, że usługi sieciowe, to dziś zjawisko wszechobecne. Przez dwadzieścia lat rozwoju webserwisów najbardziej rozbudowanymi i popularnymi narzędziami do tworzenia usług sieciowych w języku C++ stały się gSOAP i Axis/C, oparte na protokole SOAP [1].

Wymagania aplikacji opartych na architekturze web-serwisowych są zróżnicowane, a dostępne frameworki są zaprojektowane tak, żeby spełniały różne wymagania programistów. Każdy z nich posiada określoną wydajność „end-to-end", wydajność serializacji i deserializacji, określony poziom wykorzystania pamięci, skalowalność itp.

Obecnie szybkość realizacji usług sieciowych to główne kryterium dla użytkowników. Jeżeli wymiana danych zajmuje zbyt dużo czasu, to spowoduje, że klienci będą szukać innych sposobów komunikowania. Taka teza jest właściwa w odniesieniu do wszelkich nowoczesnych aplikacji: bankowych, aplikacji przeznaczonych do przesyłania multimediów i informacji tekstowych (serwisy społecznościowe), aplikacji obsługujących sklepy internetowe, czy rezerwacji miejsc [2].

W ciągu ostatnich lat, zwłaszcza w związku z rewolucją mobilną, wiele działań gospodarczych i społecznych zostało przeniesionych do sieci. Z tego powodu postrzegana przez użytkowników wydajność sieci jest obecnie podstawowym miernikiem nowoczesnych usług sieciowych. Ponieważ przepustowość pozostaje stosunkowo tania, opóźnienie w sieci jest aktualnie główną przeszkodą w poprawie wydajności. Co więcej, wiadomo, że opóźnienie sieci powoduje zmniejszenie zysku. Na przykład Amazon szacuje, że każde 100 ms opóźnienia zmniejsza zyski o 1% [3].

Głównym celem niniejszego artykułu było przeprowadzenie analizy porównawczej dwóch popularnych frameworków implementujących SOAP w języku C++ (Axis/C i gSOAP) oraz wskazanie bardziej wydajnego narzędzia do przesyłania danych w sieci.

2. gSOAP i Axic/C

Protokół SOAP jest protokołem, czyli mechanizmem transportu informacji. Przenoszone informacje są uporządkowane (posiadają strukturę i typy). Dane zapisane są w języku XML. Za pomocą SOAP można przenosić różnego rodzaju dane, które są kodowane w razie potrzeby do postaci wymaganej przez XML.

W przypadku przesyłania danych protokołem SOAP, stosuje się następujący algorytm:

1. Analiza struktury przesyłanych danych - określenie typu i rozmiaru zmiennych do przesłania przez sieć.
2. Konwersja danych na format ASCII/UTF (niezbędne dla XML). W przypadku ciągów ASCII konwersja jest prosta i szybka. Konwersja liczb całkowitych na ASCII obejmuje konwersję binarną na dziesiętną. Konwersja liczb zmiennoprzecinkowych na ASCII wymaga również konwersji binarnej na dziesiętną, ale konwersja zmiennoprzecinkowa jest znacznie bardziej skomplikowana niż konwersja liczb całkowitych.
3. Zapis danych ASCII w formacie XML w pamięci. Sposób przechowywania ciągu znaków, może wpływać na liczbę wymaganych operacji z pamięcią.
4. Wysłanie komunikatu HTTP i XML przez system operacyjny.

W przypadku odbierania danych postępuje się analogicznie, tylko poszczególne kroki algorytmu są wykonywane w odwrotnej kolejności.

Operacje te wpływają na działanie całego systemu. W każdym z dwóch analizowanych frameworków etapy te realizowane są w różny sposób.

gSOAP obsługuje serializowanie podstawowych typów danych C/C++ do składni XML, co oznacza, że dowolny typ danych, obiekt, struktura, zdefiniowane przez użytkownika, muszą być konwertowane do postaci najprostszych typów przed ich transmisją. Narzędzia gSOAP zapewniają powiązanie SOAP/XML z językami C i C++, aby ułatwić rozwój usług internetowych i dostępność aplikacji klienckich na szerokim zakresie urządzeń. Framework zawiera kompilator, za pomocą którego automatycznie mapuje natywne i zdefiniowane przez użytkownika typy danych C i C++ do typów danych XML i odwrotnie. gSOAP wykorzystuje techniki „xml-predictive" i „pull-based", co oznacza, że podczas serializacji/deserializacji koperty XML, w zależności od poziomu jej komplikacji, odbywa się podwyższenie lub obniżenie częstotliwości procesora, niezbędnej do realizacji konkretnej operacji. Wykorzystanie takiej techniki parsowania XML pozwala osiągnąć większą wydajność i zmniejszyć wykorzystanie pamięci RAM.

Z kolei Axis korzysta z własnego modelu obiektowego AXIOM i API do parsowania XML (StAX) w celu zwiększenia wydajności [4]. Biblioteka AXIOM zapewnia implementację modelu obiektowego zgodnego z XML, który obsługuje budowę drzewa obiektów na żądanie. Również obsługuje model, który umożliwia wyłączenie budowania drzewa i bezpośredni dostęp do strumienia za pomocą interfejsu API StAX. Posiada również wbudowaną obsługę pakietu XML Optimized Packaging (XOP) i MTOM, których kombinacja pozwala XML na wydajną i przejrzystą transmisję danych binarnych. Połączenie tych czynników stanowi łatwy w użyciu interfejs API o wysokiej wydajności. Opracowany jako część Apache Axis2, Apache AXIOM jest rdzeniem Apache Axis2, ale również może być używany niezależnie od Apache Axis2 [5].

Jedną z kluczowych zalet Axis jest to, że przechowuje logikę i dane w oddzielnych komponentach, co pozwala zoptymalizować wydajność podczas przesyłania danych.

Protokół transportowy w omawianych frameworkach też jest wykorzystywany w różny sposób. Axis korzysta z pamięci RAM do zapisywania informacji w postaci drzewa DOM, po czym, w porządku kolejki, wysyła dane do klienta. W tym względzie gSOAP jest zupełnie inny od Axis. Korzysta z gniazd API i częściowo zapisuje dane bezpośrednio do strumienia wyjściowego tych gniazd.

Wykorzystanie pamięci może być czynnikiem decydującym o wdrożeniu aplikacji usług internetowych na urządzeniach wbudowanych i serwerach o dużym obciążeniu. Duże zużycie pamięci wpływa na wydajność wielowątkowego serwera, który musi obsługiwać wiele jednoczesnych połączeń. Ponadto wzrost wykorzystania pamięci dynamicznej może negatywnie wpłynąć na efektywność pamięci podręcznej. Chociaż alokator i dealokator ułatwiają alokację pamięci z punktu widzenia programisty, może to stanowić problemem dla wydajności aplikacji.

Podejście Axis do rozwiązania tego problemu, to możliwość użycia obiektu WSDLWrapper, który pozwala zoptymalizować zużycie pamięci. Niestety nie wiadomo jakie techniki wykorzystuje w celu zmniejszenia tego zużycia, a w dokumentacji można znaleźć tylko "Implementacja WSDL Definition wykorzystuje różne strategie do kontrolowania zużycia pamięci" [6].

gSOAP unika kosztownych operacji kopiowania buforów przy użyciu algorytmu dwóch iteracji. Pierwsza iteracja ocenia strukturę danych i oblicza rozmiar danych do przesyłania (w bajtach), druga iteracja ładuje dane bezpośrednio do gniazda. Także wykorzystuje dodatkowe techniki zmniejszające zużycie pamięci, m.in. technikę parsowania "schema-specific" w celu zmniejszenia wymagań dotyczących konsumpcji pamięci podczas pracy z XML przy użyciu jednego, wstępnie przydzielonego bufora dla wejścia/wyjścia, wykorzystanie TLB (ang. translation lookaside buffer) dla szybkiego dostępu do zawartości XML oraz kompresji HTTP w celu zmniejszenia rozmiaru wiadomości.

3. Przegląd literatury

Bardzo podobna tematyka do yej poruszanej w niniejszym artykule, została podjęta w pracy [7]. Prace dotyczyły procesorów XML i były prowadzone przez pracowników Uniwersytetu Stanu Nowy Jork we

współpracy z Uniwersytetem Stanu Florydy. Autorzy sugerują, aby wybór frameworka uzależniać od wydajności procesora XML [7]. Wbrew tej opinii, nie można ignorować innych modułów, które wpływają na działanie SOAP, np.: protokoły TCP/IP i HTTP, zarządzanie kolejkami, wielowątkowość, automatyczna dealokacja pamięci (ang. *garbage collector*), przepustowość sieci itp.

W pracy profesorów Helsińskiego Instytutu Technologii Informacyjnych [8] skupiono się na wykorzystaniu protokołów transportowych i ich zorganizowaniu tak, żeby były maksymalnie wydajne w zestawieniu z pozostałymi elementami SOAP. Wówczas nie istniał jeszcze osobny i powszechnie akceptowany standard komunikacyjno-transportowy. W pracy tej określono i omówiono kwestie wykorzystania SOAP: możliwość asynchronicznych zapytań, wykorzystanie web-serwisów na telefonach komórkowych, wykorzystanie pamięci głównej, metody kompresji danych i metody ulepszenia istniejących protokołów do przesyłania danych. Wyniki przedstawionych tam badań nie określają wydajności SOAP jako monolitycznego, wyspecyfikowanego i ogólnie przyjętego narzędzia. W badaniu tym SOAP jest narzędziem, a nie przedmiotem badania.

4. Metoda badań

W środowisku programistycznym Visual Studio stworzono cztery aplikacje: serwerowe i klienckie, przy użyciu każdego z porównywanych frameworków.

Badania były przeprowadzone na komputerze o następujących parametrach:

- system operacyjny: Microsoft Windows 10 Pro x64;
- CPU: Intel Core i5-6200U Skylake;
- GPU: AMD Radeon R5 M330;
- RAM: Samsung 4 GB.

Wykorzystano oprogramowanie:

- IDE: Microsoft Visual Studio Community 2017 15.9.23;
- Build acceleration: IncrediBuild 1.5.0.3;
- Platform toolset: Visual Studio 2015 (v140).

Testowano frameworki w wersjach:

- Apache Axis2/C (axis2c-src-1.6.0);
- gSOAP 2.8.101.

Aplikacje wykorzystano do testowania:

- opóźnienia czasowe w przypadku wysyłania i odbierania wywołań SOAP;
- serializacji/deserializacji dla różnych typów danych (Void, Double, String, Integer, Struct, obrazów typu MIME image/jpg);
- wykorzystania pamięci RAM.

Badania przeprowadzone zostały dla różnych typów danych, począwszy od tych najmniej obciążających protokół transportowy i silnik XML. Dzięki temu można było łatwo zauważyć wzrost czasu i mocy obliczeniowych podczas zwiększania rozmiaru i złożoności danych przesyłanych przez sieć. Rezultaty otrzymano po przeprowadzeniu dziesięciu pomiarów wydajności dla zbioru danych liczących od dziesięciu do miliona zmiennych, obiektów.

5. Wyniki testów

W tabelach 1, 3 oraz na rysunkach 1, 3 porównywano wyniki pomiarów czasu przesyłania danych różnego typu. W tabelach 2, 4 i na rysunkach 2,4 przedstawiono wykorzystanie pamięci.

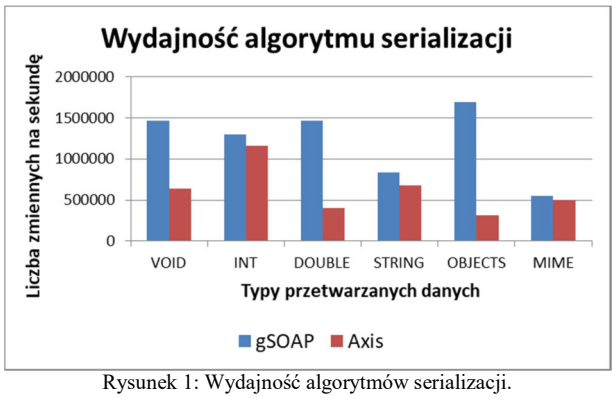
Axis miał problemy ze stabilnością, które uniemożliwiały ukończenie testów dla danych składających się z ponad 5000 zmiennych.

Model obiektowy realizowany w Axis, w porównaniu z gSOAP, wypada gorzej prawie we wszystkich testach, oprócz przetwarzania danych typu string. Jednak dane w postaci ciągu znaków są najprostsze do przetwarzania, ponieważ przekształcenie do postaci ciągu znaków jest jednym z etapów serializacji [9].

Tabela 1 i rysunek 1 pokazują, że w przypadku serializacji, we wszystkich przypadkach gSOAP jest szybszy niż Axis. W przypadku deserializacji (Tabela 3, Rys. 3), gSOAP działa również szybciej. Skrótem [zm/s] określono liczbę przesyłanych zmiennych w ciągu sekundy.

Tabela 1: Porównanie wydajności algorytmu serializacji.

Serializacja			
Typ danych	**gSOAP [zm/s]**	**Axis [zm/s]**	**Różnica [%]**
VOID	1,461,965	639,146	128.74%
INT	1,301,413	1,163,028	11.90%
DOUBLE	1,466,225	400,229	266.35%
STRING	832,806	681,776	22.15%
OBJECTS	1,688,468	313,886	437.92%
MIME	549,936	495,487	10.99%

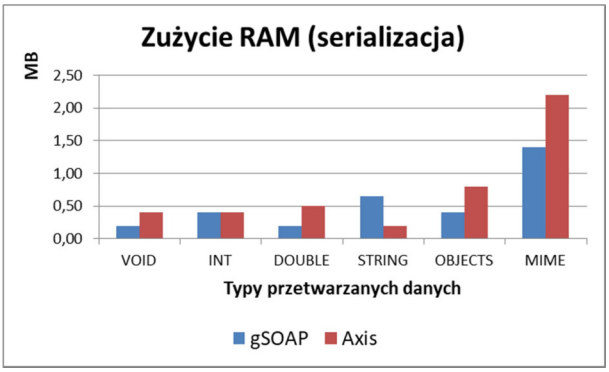


Rysunek 1: Wydajność algorytmów serializacji.

Tabela 2: Porównanie zużycia RAM algorytmem serializacji.

RAM (serializacja)			
Typ danych	**gSOAP (MB)**	**Axis (MB)**	**Różnica (%)**
VOID	0.2	0.4	50.00%
INT	0.4	0.4	0.00%
DOUBLE	0.2	0.5	60.00%
STRING	0.65	0.2	225.00%
OBJECTS	0.4	0.8	50.00%
MIME	1.4	2.2	36.36%

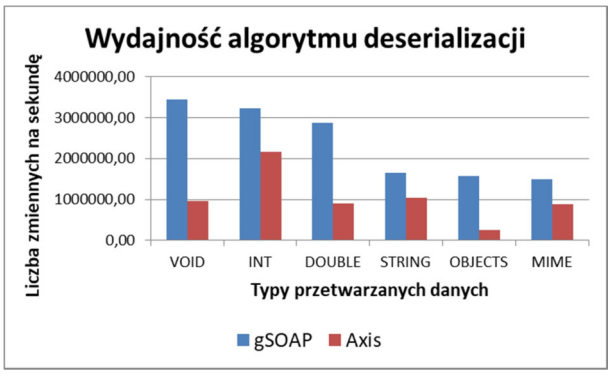
W przypadku przekształcania danych typu string z punktu widzenia zużycia pamięci (Rys. 2) jest lepszy Axis. W tym przypadku algorytm nie przekształca danych do postaci ASCII\UTF a są one bezpośrednio mapowane do składni XML.



Rysunek 2: Zużycie pamięci podczas serializacji.

Tabela 3: Porównanie wydajności algorytmów deserializacji.

Deserializacja			
Typ danych	**gSOAP (zm/sek)**	**Axis (zm/sek)**	**Różnica (%)**
VOID	3,441,712	963,241	257.31%
INT	3,233,707	2,159,858	49.72%
DOUBLE	2,863,494	899,128	218.47%
STRING	1,644,745	1,047,961	56.95%
OBJECTS	1,577,937	244,914	544.28%
MIME	1,490,912	875,489	70.29%

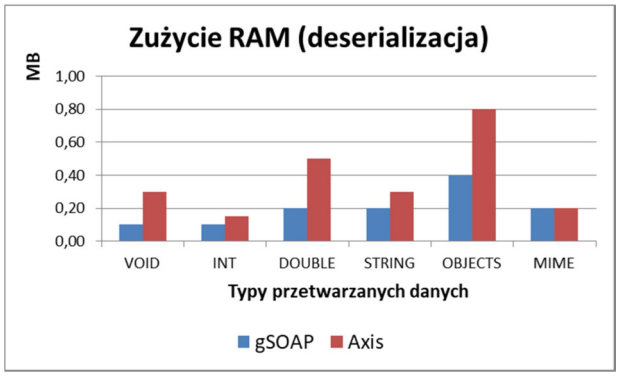


Rysunek 3: Wydajność algorytmów deserializacji.

Tabela 4: Porównanie zużycia RAM algorytmem deserializacji.

RAM (Deserializacja)			
Typ danych	**gSOAP (MB)**	**Axis (MB)**	**Różnica (%)**
VOID	0.1	0.3	66.67%
INT	0.1	0.15	33.33%
DOUBLE	0.2	0.5	60.00%
STRING	0.2	0.3	33.33%
OBJECTS	0.4	0.8	50.00%
MIME	0.2	0.2	0.00%

W przypadku wytwarzania aplikacji, które muszą przetwarzać duże zbiory danych liczbowych, rozsądniej będzie korzystać z gSOAP. Jeżeli architektura serwera jest oparta na paradygmacie programowania OOP i transmisji danych w postaci obiektów to gSOAP również jest wydajniejszym rozwiązaniem. Gdy głównym założeniem aplikacji jest wymiana plików multimedialnych - gSOAP zużywa mniej pamięci.



Rysunek 4: Zużycie pamięci podczas deserializacji.

Zużycie pamięci przez aplikację SOAP jest zwykle niskie. Rozmiar kodu zależy liniowo od rozmiaru schematów WSDL i/lub XML, które są mapowane na typy C/C++. Wykorzystanie pamięci podczas realizacji zależy od rozmiaru danych, które mają być przechowywane w czasie wykonywania aplikacji, które obejmują rozmiar zmiennych C/C++ serializowanych i deserializowanych w XML. Dane są bezpośrednio serializowane w formacie XML. Pewna ilość pamięci stosu i sterty jest używana przez kontekst silnika gSOAP, który ma wewnętrzny bufor IN/OUT (rozmiar bufora to 64 KB), przechowywany na stosie w celu optymalizacji transferu danych (z wykorzystaniem gniazd). W rezultacie standardowej kompilacji aplikacji klienckiej, jej całkowity rozmiar to 224 Kb (20 Kb jest przydzielane na stercie i 66 Kb na stosie).

Optymalizacja wydajności w Axis odbiła się na zużyciu pamięci podczas działania aplikacji. Możliwość wysłania wielu buforów danych za pośrednictwem pojedynczego wywołania skutkuje tym, że niezbędne jest przechowywanie wszystkich danych w pamięci. Podobnie przesyłanie dużej ilości danych przez sieć jednocześnie znacznie zwiększa czas transmisji danych.

Po zbadaniu przepustowości łącza dla danych potrzebujących fragmentacji (tabela 5) pomiary pokazały, że w przeciwieństwie do Axis, gSOAP eliminuje opóźnienie w czasie potrzebne do rozbicia danych, co wskazuje na różne realizacje protokołów wykorzystywanych podczas działania frameworków, a także na różne sposoby ich wykorzystania. gSOAP, podobnie jak Axis oferuje standardowy zakres narzędzi do sterowania protokołem TCP.

Tabela 5: Porównanie wydajności TCP+HTTP dla różnych rozmiarów danych.

Ping			
Fragmentacja danych	**gSOAP (pak/sek)**	**Axis (pak/sek)**	**Różnica (%)**
Tak	930	616	50.92%
Nie	968	342	183.26%

gSOAP, korzysta z TCP Hybla, który ma na celu wyeliminowanie opóźnienia połączenia TCP. Ten protokół zamyka połączenie z dużym opóźnieniem (na podstawie średniego czasu opóźnienia) i i ponownie próbuje nawiązać połączenie. Protokół osiąga to poprzez analityczną ocenę dynamiki przeciążenia, która steruje połączeniami i eliminuje niski RTT [10].

Przesyłanie danych przez sieć jest kosztowne w przypadku, gdy należy zoptymalizować wydajność pod względem zużycia pamięci. Architektura Axis pozwala na wysłanie wielu buforów danych za pośrednictwem pojedynczego wywołania systemowego, co spowalnia przepustowość łącza. Natomiast gSOAP wysyła dane podczas ich przetwarzania, bezpośrednio z użyciem gniazd. W przypadku gSOAP wykorzystanie tej techniki i tryb wykorzystania protokołu TCP jest bardziej wydajny zarówno pod względem przesyłania danych, jak i zużycia pamięci (tabela 6).

Tabela 6: Porównanie zużycia RAM protokołem TCP+HTTP dla danych różnych rozmiarów.

RAM			
Fragmentacja danych	**gSOAP (MB/1000 pak)**	**Axis (MB/1000 pak)**	**Różnica (%)**
Tak	7.6-50	10-50	~4.16
Nie	7.6-50	25-70	~36.84

6. Wnioski

W pracy przeanalizowano wydajność przesyłania danych za pomocą protokołu SOAP i porównano zużycie pamięci przez oba frameworki. Wyniki badań mogą pomóc w projektowaniu i opracowaniu nowych zestawów narzędzi SOAP oraz mogą stanowić wskazówki dla użytkowników co do wyboru frameworkam, czy sposobu jego dopasowania do szczegółowych wymagań aplikacji.

Uzyskane wyniki pomiarów pozwalają stwierdzić, że w przypadku wytwarzania aplikacji, które muszą przetwarzać duże zbiory danych liczbowych, rozsądniej będzie korzystać z gSOAP. Prędkość przekształcania danych typu string jest szybsza w gSOAP. Jeżeli architektura serwera jest oparta na paradygmacie programowania OOP i transmisji danych w postaci obiektów - gSOAP ponownie jest wydajniejszym rozwiązaniem. Gdy głównym założeniem aplikacji jest wymiana plików multimedialnych, gSOAP także ma przewagę pod względem zużycia pamięci. Podsumowując, wybierając narzędzie do dokonania wydajnej transmisji danych gSOAP jest lepszym wyborem.

Axis jest prostszym narzędziem i pomimo tego, że jest mniej wydajnym rozwiązaniem, to można go polecić programistom, którzy nie posiadają doświadczenia w budowaniu web-serwisów.

Literatura

[1] Lista popularnych frameworków protokołu SOAP, https://en.wikipedia.org/wiki/List_of_web_service_frameworks, [4.09.2020].

[2] Szkody spowodowane opóźnieniem przesyłania usług internetowych, https://www.coxblue.com/7-ways-slow-internet-speeds-are-hurting-your-business/, [6.09.2020].

[3] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, R. Govindan, Reducing Web Latency: The Virtue of Gentle Aggression, In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (2013) 159-170.

[4] AXIOM, https://ws.apache.org/axiom/, [10.10.2020].

[5] Model obiektowy AXIOM, https://www.ibm.com/developerworks/ru/library/ws-java2/index.html, [25.10.2020].

[6] Dokumentacja Axis, https://axis.apache.org/axis2/java/core/apidocs/org/apache/axis2/jaxws/util/WSDL4JWrapper.html, [25.10.2020].

[7] M. R. Head, Benchmarking XML processors for applications in grid web services, In Proceedings of the 2006 ACM/IEEE conference on Supercomputing (2006) 121-es.

[8] J. Kangasharju, S. Tarkoma, K. Raatikainen, Comparing SOAP Performance for Various Encodings, Protocols, and Connections, In IFIP International Conference on Personal Wireless Communications, Springer, Berlin, Heidelberg, (2003) 397-406.

[9] Serializacja i deserializacja, https://isocpp.org/wiki/faq/serialization, [1.10.2020].

[10] Opisy różnych realizacji protokołu TCP/IP, http://book.itep.ru/4/44/tcp.htm#8, [10.10.2020].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 42–48
Received: 07 December 2020
Accepted: 20 December 2020

The comparative analysis of web applications frameworks in the Node.js ecosystem

Analiza porównawcza szkieletów do budowy aplikacji internetowych w ekosystemie Node.js

Bartosz Miłosierny*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The subject of the research was the comparative analysis of three frameworks for building web applications, i.e. Express (version 4.17.1), Hapi (version 20.0.1) and Koa (version 2.13.0), operating in the Node.js ecosystem. An experiment was prepared consisting of a number of scenarios, during which the server response times to incoming requests from the client were measured. As part of the work, server test applications handling HTTP requests (GET, POST, PUT, DELETE) performing typical data operations were implemented. The applications contained the same functionalities and were built using the three tested frameworks. In individual scenarios, 1,000 requests of a given type were sent from independent clients, the times of successive responses were measured and their averages were calculated. On the basis of the obtained results, Hapi and Koa frameworks for GET, POST, PUT, DELETE requests, operating on one object or a string *Hello World!* have achieved the best, although very similar, response times. In the case of the GET request, the Koa framework proved to be the best for higher loads, achieving response times approximately 20% better than the Express framework. For high loads, the Hapi framework achieved the worst results, reaching response times over 2 times longer than the Koa framework.

Keywords: JavaScript frameworks; Node.js environment; performance comparative analysis; server applications

Streszczenie

Przedmiotem badań była analiza porównawcza trzech szkieletów do budowy aplikacji internetowych działających w ekosystemie Node.js: Express (wersja 4.17.1), Hapi (wersja 20.0.1) oraz Koa (wersja 2.13.0). Przygotowano eksperyment składający się z szeregu scenariuszy, podczas których dokonano pomiarów czasów odpowiedzi serwera na żądania przychodzące ze strony klienta. W ramach pracy zaimplementowano serwerowe aplikacje testowe obsługujące żądania HTTP (GET, POST, PUT, DELETE) realizujące typowe operacje na bazie danych. Aplikacje zawierały te same funkcjonalności i zostały zbudowane przy pomocy trzech testowanych szkieletów. W poszczególnych scenariuszach wysyłano od niezależnych klientów po 1000 żądań danego typu, dokonywano pomiarów czasów kolejnych odpowiedzi oraz obliczano ich średnie. Na podstawie uzyskanych wyników okazało się, że szkielety Hapi i Koa dla żądań typu GET, POST, PUT, DELETE, operujące na jednym obiekcie lub ciągu znaków *Hello World!* osiągnęły najlepsze, choć bardzo zbliżone czasy odpowiedzi. W przypadku żądania GET, przy większych obciążeniach zdecydowanie najlepszym okazał się szkielet Koa, uzyskując czasy odpowiedzi w przybliżeniu o 20% lepsze niż Express. Przy dużych obciążeniach zdecydowanie najgorzej wypadł szkielet Hapi osiągający ponad 2 razy dłuższe czasy odpowiedzi niż szkielet Koa.

Słowa kluczowe: szkielety programistyczne JavaScript; środowisko Node.js; analiza porównawcza wydajności; aplikacje serwerowe

*Corresponding author

Email address: bartosz.milosierny@pollub.edu.pl (B. Miłosierny)

1. Wstęp

Szybki rozwój technologii informatycznych, w tym w szczególności urządzeń mobilnych powoduje, że wiele firm przenosi prowadzoną działalność do internetu, dając w ten sposób klientom natychmiastowy i nieograniczony dostęp do swoich usług z różnych urządzeń zarówno stacjonarnych jak i mobilnych. Korzyści płynące z rozwoju usług prowadzonych z wykorzystaniem technologii informatycznych w świecie wirtualnym powodują, że aplikacje internetowe są obecnie najszybciej rozwijającą się klasą systemów oprogramowania. Brak konieczności tworzenia oddzielnych programów dostosowanych do danego rodzaju urządzenia i różnych systemów operacyjnych, a także uniknięcie problemu synchronizacji danych to tylko niektóre przykłady zalet oprogramowania działającego w internecie.

Aplikacje webowe to skomplikowane systemy składające się na ogół z dużej liczby komponentów opracowanych w różnych technologiach. Obecnie do ich budowy często stosuje się szkielety programistyczne, których używanie przyczynia się do zwiększenia wydajności programowania. Wynika to z tego, że szkielety wykorzystują wbudowane metody, których samodzielna

implementacja byłaby czasochłonna [1]. Szkielety programistyczne są na ogół udostępniane bezpłatnie. Ponadto ich zastosowanie obniża końcowe koszty wytworzenia aplikacji, ponieważ oprogramowanie powstaje szybciej i zawiera mniej błędów. Za używaniem szkieletów przemawiają także względy bezpieczeństwa. Aktywna społeczność wykorzystująca w swoich aplikacjach dany framework, natychmiast zgłasza napotkane problemy, które są niezwłocznie naprawiane.

Bogaty wachlarz technologii oferujących podobne możliwości sprawia, że już sam wybór szkieletu programistycznego jest dużym wyzwaniem. W tym celu przeprowadza się różnego typu badania, na specjalnie do tego celu przygotowanych aplikacjach testowych, budowanych na podstawie porównywanych frameworków. Wykorzystując do tego celu różne dodatkowe narzędzia, stwarzane są warunki, w których opracowywane oprogramowanie może się w przyszłości znaleźć.

Celem niniejszej pracy jest wykonanie analizy porównawczej szkieletów do budowy aplikacji webowych działających na platformie Node.js pod kątem wydajności czasowej oraz objętości kodu.

Do przeprowadzenia testów wykorzystano trzy identyczne pod względem funkcjonalności aplikacje serwerowe wysyłające żądania typu GET, POST, PUT, DELETE, do budowy których wykorzystano testowane szkielety programistyczne: Express (wersja 4.17.1), Hapi (wersja 20.0.1) oraz Koa (wersja 2.13.0). Aplikacje napisane zostały w języku JavaScript, który jest natywnie wspierany poprzez środowisko Node.js. Wykorzystano je do porównania wymienionych w tytule trzech szkieletów, biorąc pod uwagę czasy odpowiedzi serwera w reakcji na żądania przychodzące ze strony aplikacji klienckiej. Dodatkowo w analizie wzięto pod uwagę objętości fragmentów kodu odpowiedzialnych za realizację określonych funkcjonalności utworzonych aplikacji.

Analizowane szkielety oparte są na istniejącym od stosunkowo długiego czasu środowisku Node.js. Jednak w związku z zastosowaniem do budowy aplikacji języka JavaScript i rozwijanego przez firmę Google bardzo wydajnego silnika V8, Node.js wciąż zyskuje na popularności, a coraz więcej różnej wielkości firm decyduje się na przejście właśnie na tę platformę [2].

Do przeprowadzenia testów wydajnościowych wykorzystano zaimplementowane funkcje, obsługujące poszczególne rodzaje żądań, bibliotekę Perfy oraz zewnętrzne oprogramowanie o nazwie Postman, służące do testowania interfejsów komunikacyjnych aplikacji, poprzez wysyłanie do serwera żądań HTTP.

2. Przegląd literatury

W przypadku aplikacji internetowych, które mają wielowarstwową architekturę, są heterogeniczne i jednocześnie może z nich korzystać duża grupa użytkowników, bardzo ważną kwestią jest ich jakość i związana z nią wydajność. W związku z tym testowanie tego typu aplikacji jest procesem trudnym i pracochłonnym. W artykule [3], jego autorzy podkreślają wagę testowania aplikacji pod względem wydajności przy zastosowaniu różnych obciążeń uzyskiwanych poprzez zwiększanie liczby zapytań oraz poprzez wysyłanie różnej wielkości danych. Podstawową miarą wykorzystywaną w tego typu badaniach jest czas wykonania poszczególnych czynności i odpowiedzi serwera.

Magnus Greiff i André Johansson w swojej pracy licencjackiej [4] porównali szkielety programistyczne Symfony i Express przeznaczone do budowy aplikacji po stronie serwera. Badania przeprowadzano wysyłając 100, 1000, 10000 i 100000 zapytań do programów działających na serwerze, zbudowanych przy użyciu tych dwóch szkieletów, testując przy tym i porównując poprawność realizacji zapytań, czas potrzebny na ich wykonanie oraz poziom wykorzystania procesora. Wyniki testów wykazały, że aplikacja serwerowa zbudowana przy użyciu szkieletu programistycznego Express uzyskiwała krótsze czasy odpowiedzi, realizując przy tym poprawnie większą liczbę zapytań niż to było w przypadku programu zbudowanego na bazie Symfony. Aplikacja serwerowa oparta na szkielecie Express wykazała większe zużycie procesora niż aplikacja bazująca na Symfony, jednak podczas wykonania poszczególnych operacji potrzebowała ona mniej pamięci operacyjnej.

W artykule [5] autorzy Kai Lei, Yining Ma oraz Zhi Tan porównali i ocenili technologie służące do budowy aplikacji internetowych: PHP, Python i Node.js. Badania wykonano, przeprowadzając te same testy na aplikacjach zawierających takie same funkcjonalności, ale zbudowanych przy użyciu tych trzech popularnych dzisiaj technologii. Testy aplikacji: wyświetlającej tekst *Hello World!*, obliczających i zwracających dziesiąty, dwudziesty i trzydziesty wyraz ciągu Fibonacciego, jak i przeprowadzających operacje na bazie danych, w większości przypadków wykazały największą wydajność środowiska Node.js w stosunku do pozostałych, testowanych technologii.

Z kolei w artykule [6] porównano trzy technologie serwerowe Node.js, PHP/Apache oraz Nginx. Przeprowadzone badania pokazały, że aplikacja serwerowa utworzona na platformie Node.js charakteryzowała się większą wydajnością od aplikacji serwerowych Apache i Nginx. Wynikało to z tego, że Node.js lepiej wykorzystuje dostępne zasoby serwera, co przekładało się na obsługę większej liczby zapytań na sekundę. Jednak serwer Node.js okazał się być mniej wydajnym od serwera Nginx w operacjach na plikach statycznych, w których Nginx osiągał lepsze wyniki od niego i od serwera PHP/Apache.

3. Wykorzystane technologie i narzędzia

3.1. Node.js

Node.js jest wieloplatformowym środowiskiem uruchomieniowym o otwartym źródle, umożliwiającym wykonywanie kodu napisanego w języku JavaScript po stronie serwerowej. Aplikacje utworzone w środowisku Node.js, dzięki asynchronicznym operacjom wejścia/wyjścia, mogą pracować na pojedynczym procesie bez potrzeby tworzenia oddzielnych wątków dla każdego z żądań przychodzących do serwera. Środowisko, operując na architekturze zbudowanej na zdarzeniach,

uniemożliwia zatrzymywanie się kodu aplikacji podczas wczytywania danych, przechodząc do następnych instrukcji. Po ukończeniu pobierania danych, aplikacja wraca do pominiętego kodu oczekującego na wymagane dane [7].

3.2. REST

Representational State Transfer (REST) jest to styl architektury oprogramowania, wprowadzający określony schemat budowy serwisów internetowych i usług sieciowych. REST stał się wzorcem budowy aplikacji internetowych, który narzucił programistom określony sposób tworzenia programów serwerowych, ułatwiając równocześnie implementację części klienckich, które podążając za tym samym stylem, mogą w prosty sposób komunikować się z serwerem.

Architekturę REST można opisać za pomocą sześciu cech [8-10]:

- model klient-serwer – aplikacja kliencka jest odseparowana od serwerowej,
- bezstanowość – żadna informacja o stanie nie jest przechowywana przez serwer i musi być każdorazowo przesyłana przez klienta,
- przechowywanie danych w pamięci podręcznej – każde żądanie zawiera informację o tym, czy powinno być ono buforowane po stronie klienta,
- jednolity interfejs – każdy zasób powinien być dostępny przez unikalny dla niego adres URI, a odpowiedź na żądanie powinna zawierać informacje, które jasno definiują format danych, jakie określona odpowiedź zawiera,
- warstwowość - dane, jakie klient otrzymuje od serwera muszą być odseparowane od logiki systemu, która stoi za wygenerowaniem tych danych,
- kod na żądanie - serwer może udostępniać aplikacji klienckiej kod w postaci skryptów lub apletów do operowania na zasobach po stronie klienta.

3.3. Postman

Postman jest narzędziem służącym do testowania i rozwijania interfejsów programistycznych aplikacji (API), udostępnionym bezpłatnie do realizacji małych projektów przez grupy kilkuosobowe. Pozwala on na wielokrotne wysyłanie żądań HTTP do serwera, który następnie zwraca odpowiedzi wraz ze wszystkimi danymi i informacją o czasie, jaki upłynął od wysłania żądania do otrzymania odpowiedzi.

Aplikacja pozwala na zapisywanie żądań w celu ich późniejszego wykorzystania, a także użycia przez inne osoby pracujące wspólnie nad tym samym projektem [11].

Postman daje możliwość spreparowania wszystkich składników żądania według potrzeb użytkownika testującego daną aplikację. Mogą to być na przykład parametry załączane do adresu URL, nagłówek, ciało żądania, elementy autoryzacji w postaci różnych kluczy czy tokenów. Ponadto Postman działając w środowisku przeglądarki internetowej pozwala na przeglądanie danych cookie [12].

3.4. Biblioteka Perfy

Perfy jest bardzo prostym narzędziem, udostępnionym na darmowej licencji MIT, służącym do pomiaru z dokładnością do nanosekundy, w czasie rzeczywistym, wydajności wykonywania kodu aplikacji tworzonych w środowisku Node.js [13].

Użycie biblioteki Perfy polega na utworzeniu instancji wydajnościowej poprzez wywołanie metody *start* z parametrem będącym nazwą tej instancji, która jednocześnie rozpoczyna odmierzanie czasu. Zakończenie pomiaru czasu następuje w momencie wywołania metody *end*, również z parametrem będącym nazwą wcześniej utworzonej instancji. Metoda ta zwraca obiekt wraz z informacją zawierającą dokładny czas, który upłynął pomiędzy wywołaniami metod *start* i *end* [14].

4. Porównywane szkielety programistyczne

Platforma Node.js będąca środowiskiem uruchomieniowym języka JavaScript jest dojrzałą technologią, cieszącą się coraz większą popularnością wśród programistów odpowiedzialnych zarówno za część kliencką jak i serwerową. Obecnie istnieje bardzo wiele szkieletów programistycznych opartych na języku JavaScirpt, które usprawniają realizację typowych funkcjonalności aplikacji www. Do analizy, zrealizowanej w ramach tej pracy, wybrano trzy szkielety programistyczne działające w ekosystemie Node.js: Express, Hapi oraz Koa.

4.1. Express

Express.js jest szkieletem programistycznym służącym do budowy aplikacji internetowych oraz interfejsów programistycznych, który powstał jako projekt fundacji OpenJS Foundation [15]. Jest on jednym z najbardziej popularnych szkieletów do budowy interfejsów programowania aplikacji na platformie Node.js [16], a na jego bazie powstało wiele innych frameworków. Express z założenia jest bardzo minimalistyczny, co przyczynia się do zwiększenia wydajności tworzonych przy jego użyciu aplikacji, a wszystkie potrzebne funkcjonalności można dodawać za pomocą dostarczanych przez menedżera pakietów *npm* oraz wtyczek.

4.2. Hapi

Szkielet Hapi pierwotnie został stworzony do obsługi dużego ruchu sieciowego jednej z największej sieci supermarketów na świecie – Walmart. Powstał on z myślą o budowie dużych i elastycznych aplikacji za pomocą jak najmniejszej ilości kodu i nakładu pracy. Jako jeden z niewielu szkieletów, Hapi nie wykorzystuje funkcji pośredniczących zwanych „middleware". Za poświadczenia, autoryzację oraz walidację treści żądań przychodzących do serwera, odpowiadają wewnętrzne wbudowane funkcje oraz procedury zawarte w samym szkielecie [17]. Pomimo braku wsparcia dla funkcji pośredniczących, Hapi w pełni współpracuje z innymi wtyczkami oraz zewnętrznymi pakietami stworzonymi przez społeczność do pracy w Node.js.

4.3. Koa

Koa jest minimalistycznym szkieletem, który powstał z przeznaczeniem do tworzenia wydajnych aplikacji internetowych oraz interfejsów programistycznych. Aplikacja wykonana za pomocą tego szkieletu jest obiektem zawierającym tablicę funkcji pośredniczących, które na żądanie są składane i wywoływane w sposób podobny do stosu. Dzięki funkcjom pośredniczącym nie ma potrzeby używania wywołań zwrotnych [18]. Szkielet ten zapewnia prostą obsługę błędów oraz ułatwia zarządzanie kodem. Duża wydajność jest osiągana poprzez mały objętościowo kod, uruchamianie wielu zadań w sposób równoległy, używanie w kodzie asynchronicznych interfejsów API oraz kompresję gzip.

5. Metoda badań

5.1. Opis eksperymentu

Do przeprowadzenia analizy porównawczej wybranych szkieletów programistycznych służących do budowy aplikacji internetowych działających na platformie Node.js opracowano 5 scenariuszy badawczych, w ramach których wykonano:

1. pomiary czasów odpowiedzi na żądania GET,
2. pomiary czasów odpowiedzi na żądania POST,
3. pomiary czasów odpowiedzi na żądania PUT,
4. pomiary czasów odpowiedzi na żądania DELETE,
5. pomiar objętości fragmentów kodu.

Wyzwaniem dla każdej aplikacji jest możliwie najkrótszy czas, który upłynie od wysłania żądania, poprzez jego przetworzenie i otrzymanie odpowiedzi przez klienta. W ramach pracy opracowano eksperyment, podczas którego badano wydajność trzech aplikacji zawierających te same funkcjonalności i zbudowanych przy pomocy jednego z trzech testowanych szkieletów programistycznych. Zadaniem każdej aplikacji była odpowiedź na serię tysiąca żądań HTTP odpowiedniego typu (GET, POST, PUT, DELETE). Głównym parametrem branym pod uwagę do oceny wydajności danego szkieletu był średni czas cyklu życia żądania.

Na potrzeby przeprowadzenia testów przygotowany został zbiór danych w postaci obiektów JSON, składających się z losowych danych imitujących wpisy bazodanowe. Każdy taki wpis składał się z numeru identyfikacyjnego użytkownika, identyfikatora wpisu, tytułu oraz treści. Dla żądania GET, PUT oraz DELETE do pamięci komputera, na którym przeprowadzany był eksperyment, załadowano od 1 do 10000 obiektów pobranych wcześniej z serwisu JSONPlaceholder [19]. Czasy odpowiedzi serwera na żądania typu GET, POST, PUT oraz DELETE, zostały pozyskane dzięki wewnętrznej funkcji o nazwie „Runner”, wbudowanej w aplikację Postman oraz bibliotece „Perfy”, służącej do pomiarów czasu w programach tworzonych w środowisku Node.js.

5.2. Środowisko testowe

W tabeli 1 znajduje się charakterystyka środowiska testowego, wykorzystanego do realizacji badań. Zawiera ona parametry sprzętu oraz wersje oprogramowania.

Tabela 1: Parametry środowiska testowego.

Sprzęt	
Procesor	Intel(R) Core(TM) i7-3770K
Pamięć RAM	16 GB
Karta sieciowa	Atheros GbE LAN
System operacyjny	Ubuntu 20.04.1 LTS
Oprogramowanie	
Express	4.17.1
Hapi	20.0.1
Koa	2.13.0
Perfy	1.1.5
Postman	7.34.0

5.3. Scenariusz 1 - pomiar czasów odpowiedzi serwera na żądania GET

Czasy mierzono od momentu przyjścia żądania do serwera, wysłanego przez aplikację kliencką, do momentu zwrócenia klientowi odpowiedniej liczby obiektów razem z kodem „200” oznaczającym pomyślne wykonanie operacji. Zmierzone zostały czasy zwrócenia 1, 10, 100, 500, 1000, 5000 oraz 10000 obiektów jako pojedynczej odpowiedzi. Dodatkowo dokonano pomiaru czasu odpowiedzi bardzo prostej aplikacji, zwracającej ciąg znaków *Hello World!*.

Na listingu 1 zaprezentowano fragment kodu aplikacji testowej utworzonej za pomocą szkieletu Express, która po otrzymaniu żądania GET pod adresem *http://localhost/api/hello*, zwraca klientowi ciąg znaków o treści *Hello World!*.

Listing 1: Fragment kodu aplikacji testowej *Hello World!* zbudowanej za pomocą szkieletu programistycznego Express.

```
app.get('/api/hello', async (req, res) => {
    perfy.start( name: 'get-time');
    await res.send('Hello World!');
    const time = perfy.end( name: 'get-time').fullMilliseconds;
    await times.push(time);
});

app.listen(3000, () => console.log('Express listens on 3000...'));
```

W przypadku aplikacji zwracających obiekty JSON, za pośrednictwem funkcji *data_loader* klientowi zwracana jest odpowiednia liczba obiektów. Funkcjonalność tą, zrealizowaną za pomocą szkieletu Hapi, pokazano na listingu 2.

Listing 2: Fragment kodu aplikacji testowej obsługującej żądania GET zbudowanej za pomocą szkieletu Hapi.

```
(async () => {
    data = await dataLoader(/*ilość obiektów*/);
    await server.start();
    console.log('Hapi listens on 4000...')
})();
```

```
server.route({
    method: 'GET',
    path: '/api/posts',
    handler: (request, h) => {
        perfy.start( name: 'get-time');
        return data;
    }
})

server.events.on('response', async () => {
    const time = perfy.end( name: 'get-time').fullMilliseconds;
    await times.push(time);
});
```

5.4. Scenariusz 2 - pomiar czasów odpowiedzi serwera na żądania POST

W tym scenariuszu czas jest odmierzany od momentu otrzymania żądania zawierającego zagnieżdżony w swoim ciele obiekt. Następnie dodawany jest ten obiekt do pamięci, a aplikacji klienckiej zwracany jest kod „201” - oznaczający utworzenie przez serwer nowego zasobu. Po wykonaniu tych operacji następuje zakończenie pomiaru czasu i zapis wyniku.

Listing 3 przedstawia fragment kodu aplikacji testowej utworzonej za pomocą szkieletu Koa.

Listing 3: Fragment kodu aplikacji testowej obsługującej żądanie POST, zrealizowanej za pomocą szkieletu Koa.

```
app.use( fn: async (ctx, next) => {
    perfy.start( name: 'get-time');
    await next();
    ctx.res.on('finish', async () => {
        const time = perfy.end( name: 'get-time').fullMilliseconds;
        await times.push(time);
    });
});

router.post('/api/posts', async (ctx) => {
    await data.push(ctx.request.body.post);
    ctx.response.status = 201;
});
```

Po utworzeniu i zainicjowaniu zmiennych oraz przygotowaniu serwera do obsługi żądania POST, zawierającego obiekt JSON, wraz z uruchomieniem serwera, rozpoczyna się nasłuchiwanie na wskazanym porcie. Po otrzymaniu od klienta żądania POST zawierającego obiekt, rozpoczyna się odmierzanie czasu realizowane za pomocą funkcji *perfy*. Serwer umieszcza otrzymany obiekt w jednowymiarowej tablicy *data*, po czym wysyła klientowi kod „201”, oznaczający pomyślne utworzenie nowego zasobu. W tym momencie kończy się odmierzanie czasu, wynik zostaje zapisany do tablicy *times*, która następnie jest przekazywana do funkcji *logger*, a po zakończeniu pracy serwera zostaje zapisana w pliku tekstowym.

5.5. Scenariusz 3 - pomiar czasów odpowiedzi serwera na żądania PUT

W tym przypadku pomiar czasu rozpoczyna się w momencie przyjścia żądania do serwera, które w adresie URL zawiera numer modyfikowanego zasobu, a w swoim ciele przechowuje obiekt, przeznaczony do modyfikacji, która może być dokonana po stronie klienta. Po wykonaniu zmian na tym, przechowywanym w pamięci serwera obiekcie, aplikacja zwraca klientowi kod „204” - oznaczający zakończone sukcesem wykonanie operacji i nie dołącza żadnych dodatkowych danych zwrotnych. Potem następuje zaprzestanie mierzenia czasu i zapis wyniku.

Na listingu 4 przedstawiono fragment kodu aplikacji testowej utworzonej za pomocą szkieletu Express.

Listing 4: Fragment kodu aplikacji testowej obsługującej żądania PUT zbudowanej za pomocą szkieletu Express.

```
app.put('/api/posts/:id', async (req :Koa.Context , res) => {
    perfy.start( name: 'get-time');
    const id = req.params.id;
    data[id - 1] = await req.body.post;
    await res.sendStatus( statusCode: 204);
    const time = perfy.end( name: 'get-time').fullMilliseconds;
    await times.push(time);
```

Po inicjalizacji zmiennych, aplikacja serwerowa gotowa jest do obsługi żądania PUT. W momencie otrzymania od klienta żądania zawierającego numer przechowywanego w pamięci aplikacji obiektu, przeznaczonego do modyfikacji, rozpoczyna się odmierzanie czasu. W adresie *http://localhost/api/posts/id*, id jest identyfikatorem modyfikowanego obiektu. Następnie zachodzi podmiana wskazanego obiektu, obiektem zawartym w żądaniu przesłanym przez klienta. Po dokonaniu modyfikacji serwer zwraca kod „204”, który oznacza pomyślne wykonanie operacji, i w tym momencie kończy się odmierzanie czasu, który jest zapisywany w tablicy *times*. Po zakończeniu pracy serwera, ogłoszonym sygnałem *SIGINT*, funkcja *logger* zapisuje czasy odpowiedzi w pliku tekstowym.

5.6. Scenariusz 4 - pomiar czasów odpowiedzi serwera na żądania DELETE

W tym scenariuszu celem było zmierzenie czasu, jaki zajmuje aplikacji serwerowej obsługa żądania DELETE. Pomiar czasu rozpoczynał się w momencie przyjścia żądania zawierającego w adresie URL numer usuwanego zasobu. Po wykonaniu tej operacji i zwróceniu kodu „204”, kończył się pomiar czasu i następował jego zapis.

Listing 5 przedstawia kod aplikacji testowej utworzonej za pomocą szkieletu Hapi.

Listing 5: Fragment kodu aplikacji testowej obsługującej żądania DELETE zbudowanym za pomocą szkieletu Hapi.

```
server.route({
    method: 'DELETE',
    path: '/api/posts/{id}',
    handler: async (request, h) => {
        perfy.start( name: 'get-time');
        const id = request.params.id;
        await data.splice(id - 1, 1);
        const response = h.response();
        response.statusCode = 204;
        return response;
    }
})

server.events.on('response', async () => {
    const time = perfy.end( name: 'get-time').fullMilliseconds;
    await times.push(time);
});
```

Po utworzeniu i inicjalizacji zmiennych, przygotowywana jest odpowiedź serwera na żądanie DELETE. Po otrzymaniu takiego żądania, co następuje pod adresem *http://localhost/api/posts/id*, w którym *id* oznacza numer usuwanego obiektu, aplikacja rozpoczyna odmierzanie czasu. Następnie zostaje usunięty z pamięci wskazany przez użytkownika obiekt i w końcu aplikacja odpowiada klientowi kodem „204", który oznacza poprawne wykonanie operacji.

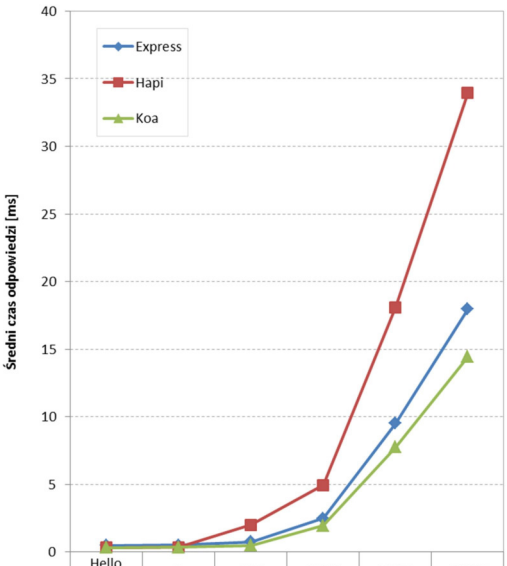
5.7. Scenariusz 5 - pomiar objętości kodu

Objętość kodu była dodatkowym wskaźnikiem, który obok wydajności, został wzięty pod uwagę podczas analizy porównawczej. Pomiar objętości był prostą czynnością polegającą na zliczeniu linii kodu, odpowiednich fragmentów stanowiących implementację danej funkcjonalności w danym szkielecie. Pustych wierszy lub wierszy zawierających wyłącznie komentarze nie uwzględniano w obliczeniach.

6. Wyniki badań

6.1. Analiza czasów odpowiedzi serwera dla żądania GET

Rysunek 1 przedstawia wyniki uzyskane po zrealizowaniu scenariusza badawczego nr 1, podczas którego wykonano pomiary czasów obsługi żądania GET przy różnych obciążeniach: 1, 100, 1000, 5000, 10000 obiektów oraz krótkiego tekstu *Hello World!*. Pomiary powtarzano 1000 razy dla każdego typu obciążenia. Następnie otrzymane wyniki uśredniono.



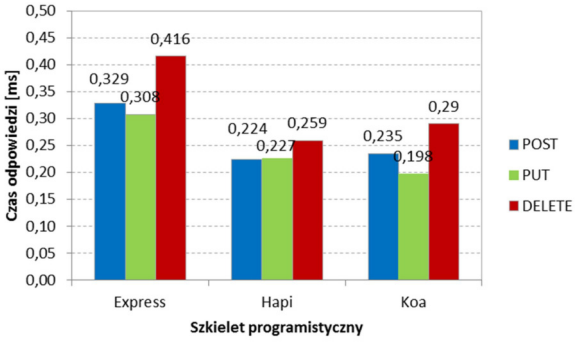
	Hello World!	1	100	1000	5000	10000
Express	0,474	0,503	0,728	2,488	9,513	17,954
Hapi	0,311	0,349	1,987	4,924	18,096	33,954
Koa	0,314	0,332	0,476	1,94	7,737	14,416

Rysunek 1: Wykres przedstawiający średnie czasy obsługi żądania GET przy różnych rodzajach obciążeń dla 3 szkieletów.

W przypadku aplikacji, która w odpowiedzi na żądanie zwracała tekst *Hello World!*, najkrótszy średni czas obsługi żądania GET miały programy zbudowane na bazie szkieletów Hapi oraz Koa. Również dla małych obciążeń, gdy w odpowiedzi zwracany był pojedynczy obiekt, czasy obsługi żądań były najmniejsze dla szkieletów Hapi i Koa. Dla obciążeń, gdy zwracanych było jednocześnie 100 obiektów, najkrótszy średni czas odpowiedzi uzyskał szkielet Koa. Na przygotowanie i wysłanie odpowiedzi szkielet Express potrzebował dwa razy dłuższego czasu, natomiast szkielet Hapi aż 5 razy dłuższego czasu niż Koa. W przypadku dużych obciążeń (1000, 5000 i 10000 zwracanych obiektów JSON) różnice w czasie obsługi żądania GET, między testowanymi szkieletami były jeszcze większe.

6.2. Analiza czasów odpowiedzi serwera dla żądań POST, PUT i DELETE

Na rysunku 2 zaprezentowano wyniki uzyskane po przeprowadzeniu badań według scenariuszy nr 2, 3 i 4. Zawierają one średnie czasy obsługi trzech typów żądań (POST, PUT i DELETE) realizowanych za pomocą aplikacji testowych opracowanych na podstawie trzech testowanych szkieletów.



Rysunek 2: Wykres przedstawiający średnie czasy obsługi żądań POST, PUT i DELETE przy różnych obciążeniach dla 3 szkieletów.

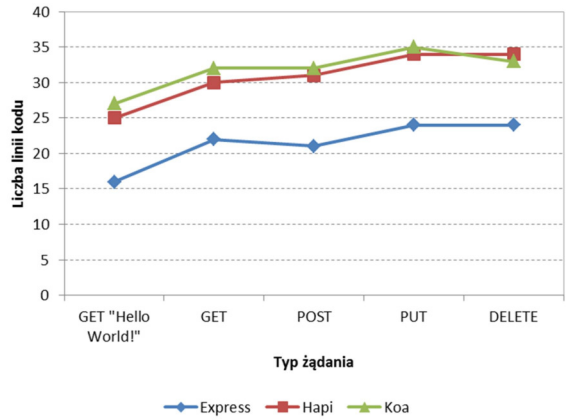
Z wykresu przedstawionego na rysunku 2 można wysnuć następujące wnioski:

1. Dla żądania POST najkrótszy średni czas odpowiedzi miał szkielet Hapi, niewiele dłuższy Koa, a najdłuższy Express.
2. Dla żądania PUT, najkrótszy średni czas odpowiedzi miał szkielet Koa, nieznacznie dłuższy Hapi i najdłuższy, również w tym przypadku szkielet Express.
3. Dla żądania DELETE, najkrótszy średni czas uzyskał szkielet Hapi, trochę dłuższy Koa i znacznie dłuższy Express.

6.3. Analiza objętości kodów

Rysunek 3 obrazuje wyniki pomiarów objętości kodu (wyrażonej w liczbie linii) zrealizowanych wg scenariusza nr 5. Biorąc pod uwagę liczbę linii kodu, fragmentów programu obsługujących poszczególne rodzaje żądań, zdecydowanie najlepszym okazał się szkielet Express. Aplikacja testowa wykonana na podstawie tej platformy programistycznej, dla każdej operacji powiązanej z określonym żądaniem, miała najmniejszą objętość. W przypadku Hapi oraz Koa wyniki były zbliżone, choć znacznie gorsze w porównaniu do szkieletu Express. Większa ilość wierszy kodu w przypadku tych dwóch szkieletów wynikała po pierwsze z potrzeby dołączania dodatkowych bibliotek odpowiedzialnych na

przykład za dodanie możliwości dostępu do ciała żądania, oraz po drugie ze specyfiki implementacji funkcji obsługujących żądania w poszczególnych szkieletach.



Rysunek 3: Wykres prezentujący objętość kodu odpowiedzialnego za obsługę poszczególnych typów żądań przez aplikacje wykonane na podstawie danego szkieletu.

7. Wnioski

Przed przystąpieniem do budowy aplikacji, wiedząc jakie będzie jej przeznaczenie, przewidując w jakich warunkach obciążeniowych będzie ona funkcjonowała oraz jakie będą jej stawiane wymagania, należy w sposób przemyślany, oparty na wynikach badań lub na wynikach badań innych ekspertów, dokonać optymalnego wyboru technologii. Praca ta powstała z myślą, aby wspomóc programistów i ułatwić im podjęcie decyzji dotyczącej wyboru szkieletu programistycznego opartego na języku JavaScript i funkcjonującego w ekosystemie Node.js.

Eksperyment, przeprowadzony na podstawie czterech scenariuszy, dał wyniki pozwalające na wykonanie analiz ilościowych. W porównaniach wzięto pod uwagę dwie miary: średni czas obsługi czterech typów żądań oraz objętość kodu wyrażoną w liczbie linii. Porównanie szkieletów było możliwe po utworzeniu aplikacji testowych, z których każda, choć posiadała dokładnie te same funkcjonalności, zbudowana została za pomocą innego szkieletu.

Analiza czasu obsługi żądania GET była rozszerzona, gdyż uwzględniała różne rodzaje i wielkości obciążeń. Zatem odpowiedzi generowane przez serwer miały różne wielkości. Dla żądań POST, PUT i DELETE odpowiedzi miały postać kodu liczbowego, informującego o zakończonym sukcesem wykonaniem operacji.

Na podstawie otrzymanych wyników można sformułować dwa generalne wnioski:

1. W przypadku żądania GET, najkrótsze czasy obsługi żądań uzyskiwała aplikacja zbudowana za pomocą szkieletu Koa. Nieco gorszą wydajność osiągała aplikacja oparta na szkielecie Express.
2. Dla żądań POST, PUT i DELETE najlepsze czasy miały Koa i Hapi. Wyniki aplikacji opartej na szkielecie Express były o około 30% gorsze.

W ramach pracy wykonano również analizę objętości kodów odpowiedzialnych za obsługę poszczególnego typu żądania, aplikacji testowych zbudowanych za pomocą różnych szkieletów. Wyniki jednoznacznie pokazały, że najkrótszy kod miała aplikacja zbudowana na bazie platformy programistycznej Express.

Literatura

[1] A. Wójcik, M. Wolski, J. B. Smołka, Performance analysis of the Symfony framework for creating modern web application based on selected versions, Journal of Computer Sciences Institute 13 (2019) 293-297.

[2] Global companies using Node.js in production, https://www.tothenew.com/blog/how-are-10-global-companies-using-node-js-in-production/, [04.11.2020]

[3] G. A. Di Lucca, A. R. Fasolino, Testing web-based applications: The state of the art and future trends, Information and Software Technology 48 (2006) 1172-1186.

[4] M. Greiff, A. Johansson, Symfony vs Express: A Server-Side Framework Comparison, Dissertation 2019.

[5] K. Lei, Y. Ma, Z. Tan, Performance comparison and evaluation of web development technologies in PHP, Python and Node.js, IEEE Computer Society (2014) 661-668.

[6] I. K. Chaniotis, K. D. Kyriakou, N. D. Tselikas, Is Node.js a viable option for building modern web applications? A performance evaluation study, Computing 97 (10) (2015) 1023-1044.

[7] Rozpoczęcie pracy z Node.js, https://riptutorial.com/pl/node-js, [07.11.2020].

[8] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, 2000.

[9] Wstęp do REST API, https://devszczepaniak.pl/wstep-do-rest-api/, [04.11.2020].

[10] Mastering REST, https://medium.com/@ahmetozlu93/mastering-rest-architecture-rest-architecture-details-e47ec659f6bc, [04.11.2020].

[11] Postman, https://www.postman.com/, [04.11.2020].

[12] B. Mehta, RESTful Java Patterns and Best Practices, Packt Publishing, 2014.

[13] Perfy, https://www.npmjs.com/package/perfy, [04.11.2020].

[14] Dokumentacja Perfy, https://github.com/onury/perfy, [04.11.2020].

[15] Dokumentacja Express, https://expressjs.com/, [04.11.2020].

[16] Zestawienie szkieletów Node.js, https://www.simform.com/best-nodejs-frameworks/, [04.11.2020].

[17] Dokumentacja Hapi, https://hapi.dev/, [04.11.2020].

[18] Dokumentacja Koa, https://github.com/koajs/koa, [04.11.2020].

[19] JSONPlaceholder, https://jsonplaceholder.typicode.com/, [04.11.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 49–54
Received: 14 December 2020
Accepted: 9 February 2021

Human-social interaction robots to improve shared attention in children with autism

Wykorzystanie interakcji człowiek-robot w celu poprawy rozproszonej uwagi u dzieci z autyzmem

Ali Ashrafi*

English Department, Islamic Azad University, Shahr-e-Qods Branch, Tehran, Iran

Abstract

The motivation behind the momentum research is use of robots explicitly human robots which has filled drastically in various domains, including treatment. In this examination, a humanoid robot was used to improve open consideration in children with mental imbalance. One of the troubles of this system is that contamination makes remarkable conditions for the patient that the presence of the guide and some other new thing isn't easily recognized. The subsequent test is to pick appropriate figuring and systems for following the head and understudy of the eye in youthful steers with mental awkwardness. One of the credits of which is obligatory and uncontrolled improvements of the head and eyes to the sides. The third issue is the treatment and investigation strategies. The treatment cycle and the arranged tests should not to reason extravagant instigation in the youth. The aim is to beat the referred to troubles, not-withstanding the high-block progressing understudy following count, without the use of business gear. Fleecy decision tree has been used to join clinical and planning information during therapy, ultimately the possibility of instinctive therapy for the improvement of restoratively withdrawn young people has been introduced.

Keywords: Social Assistant Robots; Autism; Pupil Detection; Human-Robot Interaction; Fuzzy Decision Making

*Corresponding author

Email address: **Ali.Ashrafi.Sarvak11@gmail.com** (A. Ashrafi)

1. Introduction

Mental imbalance or autism is an issue that is influencing an ever increasing number of individuals. The common consideration is one of the fundamental boundaries of this infection and its improvement is straightforwardly identified with the general treatment of the illness, and not focusing on it greatly affects the movement of the sickness [1]. The rate of mental imbalance on the planet has developed altogether. In addition, some experts believe that 1 out of 10 individuals is brought into the world with autism [2]. Due to the lack of specific and definitive treatment of this disease, and the special characteristics of patients with it .For example rejection of treatment and inability to interact with the environment; the use of technologies such as computers and robotics is of particular importance, because different treatment scenarios can be planned and tested with their help and their effects. As a result, a suitable method can be provided for the treatment of cognitive diseases and used in the treatment of patients [3]. Kids with autism regularly experience issues perceiving the enthusiastic conditions of everyone around them. For example, an upbeat face or a terrifying face. In order to address this, specialists utilize flawless looking robots to show feelings to kids with autism can mirror them and react to them fittingly. This kind of treatment is the best treatment [4]. On the off chance that the robot can give an away from of the kid's conduct and show when he is energized or focusing during the treatment, he will help the treatment cycle [5]. Helpful techniques utilized in this examination to treat the imparted consideration of individuals to chemical imbalance are PC games or game situation plan with social help robots. In this exploration, two situations have been planned. The principal situation is an augmented simulation and the subsequent situation is utilizing the Nao robot [6]. The outcomes show that notwithstanding improving the treatment cycle of patients, we had the option to precisely assess the utilization of advanced mechanics and augmented reality in association with medically introverted individuals and report the most fitting approach to treat this illness [7]. The augmented simulation game is with the end goal that in the wake of being played on a PC. In any case, it is shown on a screen by a video projector and the kid follows a fledgling which moves between the cells of a table, and simultaneously, the understudy of the eye by the inherent camera before the kid is followed continuously. Eventually, the aftereffects of the trial are introduced as outlines and with the assessment of the game, history can perform by the kid. The means of the game change to quicken the mending cycle [8]. The subsequent situation is finished utilizing NaO humanoid robot in a genuine climate and sensor. In the wake of beginning the situation, the robot focuses to a particular objective pointing at the kid and requests that he take a gander at it [9]. At the same time, the head and body turn data is separated by the sensor. All together

for the proposed answer for be intelligent (ie, the robot reacts as indicated by the kid's presentation), the fluffy choice tree is utilized [10]. In such manner, the measure of pivot of the youngster's head and body point comparative with the objective, just as the kid's clinical history, which is as of now inserted in a fluffy master framework. Additionally, it is utilized by the robot to choose the following choice [11]. Following quite a while of examination, Aldebaran Mechanical technology has dispatched a profoundly progressed social humanoid robot called NAO. This bipedal robot, which is 58 cm tall, is one of the pioneers of advanced mechanics with its legitimate plan and mix of programming and equipment. Highlights of the NAO robot incorporate full programming capacity, sensors, underlying PCs, controller ability and light and lovely body. This study investigates the effect of humanoid social robots as assistant therapists in the rehabilitation and education of children with autism as one of the first groups to use this technology.

2. Theoretical background

Massachusetts Institute of Technology specialists have built up a remarkable sort of AI that assists robots with surveying their kid's advantages while collaborating with every youngster's extraordinary information [12]. As indicated by considers, 60% of robots 'impression of kids' circumstances are predictable with the specialists' understandings [13]. The drawn out objective of this arrangement isn't to supplant these robots with restorative specialists [14]. However, they assist advisors with finishing key data in remedial correspondence by making normal and valuable connections between the robot and the mentally unbalanced kid [15]. Personalization is significant for treatment in light of the fact that every youngster with mental imbalance has interesting responses to correspondence [16]. The utilization of AI and man-made consciousness in the treatment of mental imbalance is troublesome. Since for the most part in computerized reasoning strategies, a great deal of comparative information is required for the preparation cycle [17]. Rudovic, top of the examination gathering, said that the utilization of ordinary neural organization strategies in a medically introverted patient had fizzled [18]. They utilized profound customized learning in different territories. Their discoveries improve the consequences of checking and anticipating the movement of Alzheimer's sickness [19].

3. Research Methodology

The present study was conducted using a descriptive method that gathered by library methodology, which was compiled by studying Human-social interaction robots with autism, related issues, articles and books which have mentioned in the reference. In a general division, Human-social interaction methods, features, software's have been examined. The purpose of this study is to investigate the methods of Human-social interaction in order to help children who suffer autism. These methods include Virtual reality method and the use of NAO robots. The results of this study show that in addition to improving the treatment process of patients, we were able to accurately assess the use of robotics and virtual reality in interaction with autistic people and report the appropriate method for the treatment of this disease.

4. Results and discussion

4.1. Behavior Rating Inventory of Executive Function

A rundown of conduct rankings of chief capacities has been created to look at different parts of the elements of the front piece of the temple. This poll is planned in two structures: parent and instructor and is utilized for kids and teenagers of young men and young ladies matured 1-7 years. In the current investigation, the parent structure is utilized.

4.2. Performance Continuous Test

This test requires restraint of undesirable reactions and nonstop observing of target reactions [20]. The subject should focus for some time to a moderately basic arrangement of visual improvements on the PC screen and give a reaction key when the objective boost shows up [21].

4.3. AO robot

The specialists utilized the NAO humanoid robot in the examination. 60 cm tall robot that resembles saints and artists. NAO passes on various feelings by changing the shade of the eyes, moving the lips and changing the manner of speaking [22]. The greater part of the youngsters in most of examinations responded to the robot as a toy as well as a genuine individual. As indicated by Rudovic, advisors said a couple of moments of youngster contact could be a major test for them. Robots can catch a kid's eye. People express their feelings in an unexpected way, while the robot consistently acts similarly and the youngster can be prepared in an organized manner [23].

4.4. Individual focused AI

The MIT research group found that a sort of AI called profound learning is helpful for restorative robots. However, they can comprehend kids' conduct all the more normally. The profound learning framework utilizes a various leveled structure and numerous information layers to deal with crude information [24]. In spite of the fact that the example of profound learning has existed since the 1980s, the presence of registering power has as of late made this kind of man-made rea-

soning more down to earth. Profound learning is utilized in programmed discourse acknowledgment and item acknowledgment programs by getting a few highlights of face, body and sound. In addition, it can introduce theoretical ideas. For instance, on account of facial indications, which portion of the face is significant in deciding connection?

Profound learning permits the robot to separate information straightforwardly and without the requirement for people. The exploration group has gone above and beyond and made an individual system for showing every youngster the information gathered. Some specialists make recordings of the child's outward appearances, signals, head and body developments, and voice. They additionally gather information on pulse, internal heat level and skin sweat through a sensor on the child's wrist. A customized taking in organization from physiological information, video and sound, gives data on the conclusion and capacity of mentally unbalanced youngsters, their way of life and sex. Scientists can analyze the robot's gauge of a kid's conduct with that of five specialists dependent on kids' sound and video chronicles, and measure the youngster's degree of bliss, bitterness, energy and cooperation. This human-prepared, coded individual information is tried on undeveloped information and unadjusted models and fundamentally improving the social assessment of the kids examined. The scientists discovered fascinating data about kids' social contrasts [25].

4.5. The presented approach

Two scenarios have been considered to solve the problem. In the first scenario, the amount of shared attention of the child is evaluated by the virtual reality technique and real-time tracking of the pupil of the child's eye. In addition, by advancing the game in the later stages, it tries to increase the amount of shared attention. In the second scenario which is done using a humanoid robot, the child engages in physical and mental interaction with the robot, and while playing with it, the head-body position data is received and left to fuzzy decision making.

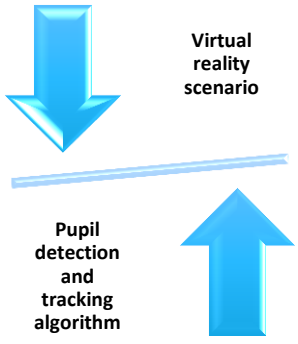


Figure 1: Two scenarios in the presented approach.

The decision tree selects different games with the aim of increasing shared attention and forces the robot to play with the child [26].

4.5.1. Virtual reality scenario

Virtual reality is a sort of innovation where a virtual environment is put before the client's eyes and interfaces with that environment through head and body developments. The principal situation of this examination in which the kid interfaces with the PC and utilizing Virtual reality innovation to create programming analysed [27].

4.5.2. Pupil detection and tracking algorithm

Methods for diagnosing the center of the eye have been proposed, which can be generally divided into the Table 1.

Table 1: Methods for diagnosing the center.

Method 1	Feature-based methods
Method 2	Model-based methods
Method 3	Combined methods

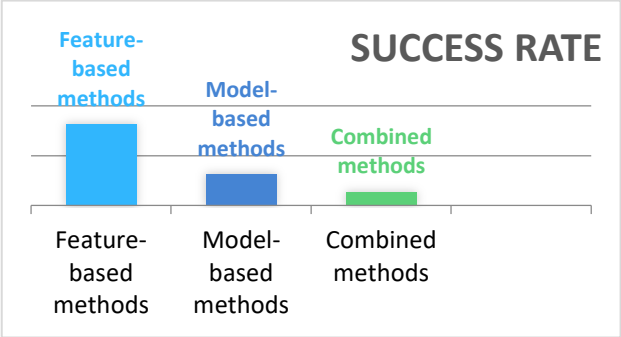


Figure 2: Success rate of diagnosing Methods.

Method 1

In order to situate the focal point of the eye, the focal point of a semi round example is where the biggest angle of the picture meets. Consequently, a numerical administrator that gets its greatest in the focal point of the circle logo is utilized. Handling strategies with the point of lessening the issues brought about by glasses and eye appearance in the eyebrows, just as fortifying the calculation introduced in an unexpected way. In the following formula, the displacement vector $\boldsymbol{d}_1$ and the gradient vector $\mathbf{g_1}$ do not have the same orientation, while on the right both orientations are the same. Geometrically, the center of a circular object can be obtained by analyzing the gradient vector. They also analyze the vector gradients. In addition, the specific properties of the vector field are formulated to describe the relationship between the probable center of the eye and all-gradient orientations of the vector. This is done with the aim of locating the center of the eye. In this formulation, the probabilistic center are shown with $\mathbf{C_1}$ and the

gradient of vector shown with $\mathbf{x_1}$. As a result, the normal displacement vector $\boldsymbol{d}_1$ must have a rotational gradient if we use the gradient vector field, we will be able to obtain this vector by calculating the internal multiplication between the displacement vectors as follow:

$$= \arg\max \{\frac{1}{n}\sum_1 N(D^T_I G_I)^2 \, c^* \quad d_{i=}\frac{(xi-c)}{xi-ci} \quad , \quad \forall_i \mid g_i \mid 2 = 1 \tag{1}$$

Method 2

In this method, in order to achieve equal weight for the position of all pixels, the displacement vector d_i is scaled to one length, and to improve stability against linear changes in lighting and contrast, [28] gradient vectors must also be scaled to one length which is given as:

$$g_i = \frac{di(xi,yi)^2}{dxi\,,dxy} \tag{2}$$

Method 3

In this method, the BioID data set should be used to evaluate the proposed method. In addition to the changes that occur in light conditions, the position and position of the subjects also change. In this method, the position of the face is first recognized and based on that, the areas of the eyes are identified according to the proportions between the eyes and the size of the face that is as follow:

$$e \leq (\frac{1}{d}) \max(e_1, e_2) \tag{3}$$

However. e_1 and e_2 are Euclidean distances between the centers of the right and left eyes and d is the distance between the centers of the real eye.

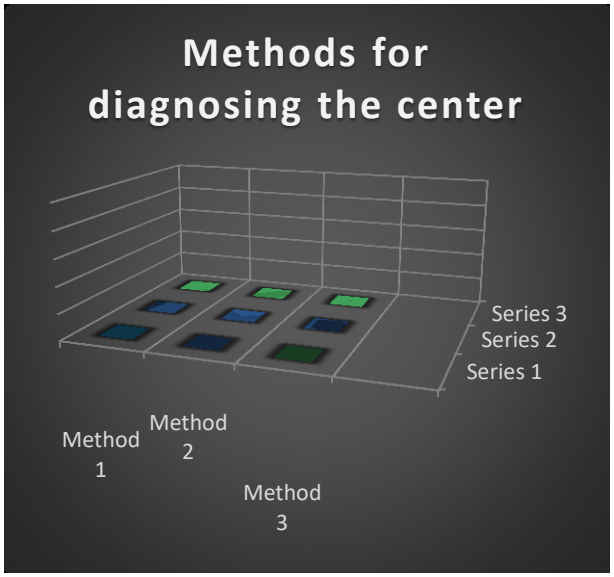


Figure 3: Methods for diagnosing the center.

4.6. Designing a game to perform a virtual reality scenario

This scenario is done by moving a bird randomly in the cells of a table. The child follows this bird with his eyes. In order to be able to check the exact movement of the pupil and the case. However, a contract analysis, the movement of the head must be restrained for which purpose the chin holder ophthalmology has been used. After the test begins, the server extracts the baby's pupil movement information through a network camera. In order to normalize the movement of the pupil and ensure its accuracy, the information is read 4 times per second and averaged between them to represent the final target.

4.7. Data collection

4.7.1. Human-robot collaboration situation

The situation intended for an individual to play with a robot has a few phases. Prior to beginning the test, the specialist advises the framework regarding the essential data required, for example, name, family name, and the degree of the last test used to set up the patient document. In this situation we will have three phases of the game, every one of which is isolated into three sections, the fluffy framework chooses the stages and their planning and sends it to the robot. To figure the mistake and get information from the situation execution measure, all execution steps in a game motor are reproduced, which are clarified underneath [29].

4.7.2. Solidarity game motor

Solidarity game motor is programming for making and creating computer games. One of the main highlights that are normally executed by game motors is the delivering motor for 3D and 2D designs, actual motor or crash recognition, sound, content.

4.8. Game plan Scenario of human collaboration with the robot

In this trial, a doll or any gadget that is appealing to kids is appended. In the principal stage, the robot goes to one of the dolls and subsequent to being under the doll, it holds its hands towards it. In the subsequent stage, the robot remains at a fixed point and focuses to one of the dolls with his hand and requests that the youngster focus on it. In the third step, the robot goes under one of the dolls and focuses to the contrary doll with his hands. Toward the start of the test, the robot begins playing from one of the stages as per the degree of the test. All through the test, the actualized calculation gets the visual sensor data and concentrates the situation of the youngster's head and body comparative with the subsequent position, and returns it to the worker. Subsequent to finishing three pieces of a phase, the measure of mistake is determined in each part. Contingent upon the test

level, the fluffy choice tree chooses the game stage, the framework can keep the game at a similar level, or relying upon the test yield, start a harder or simpler stage. Also, to have more precise data about the testing cycle, all the means acted in Unity programming have been mimicked. After the robot shows the objective to the youngster, the worker begins perusing data from the sensor for 2 seconds. It utilizes the got data to find and track the head. Besides, by utilizing a three-dimensional sensor, the skeleton of the body is first identified, at that point the head is eliminated lastly. The purpose of contact of the vector with the divider is set apart in yellow, the planned framework realizes that the cell is the objective and presentations it on the screen. The sample of the study will consist of a group of normal children and a group of children with autism in the age group of three to six years. In order to evaluate the comprehensiveness of the proposed solution, autistic children with different degrees of autism are selected. This test is also done with normal children to talk about system performance and scenarios. In addition, experiments are also designed to analyze the proposed algorithms, study scenarios and performance of individuals.

4.8.1. Experiment 1. Analyze the direction of the person and the position of the target

The target appears in one of the field cells every 2 seconds during the whole game time, and a diagram of its various positions is formed. Also after processing in each 10 seconds, a cell is displayed as the child has looked at the most, and a graph is drawn for it.

4.8.2. Experiment 2. Production of heat map

During the first scenario, each time the bird appears in one of the cells, in addition to recognizing the child's gaze and which cell it is staring at, the map of the child's gaze relative to the other cells is also used to perform analyses [30]. It is recorded more advanced. The heat map uses the color spectrum from black to white. The brighter the house, the more the child looks at the cell.

5. Conclusion

The utilization of robots to interface and treat boundaries, for example, shared consideration is a test that a significant number of the world's driving colleges have picked as their exploration field. In this examination, an endeavor has been made to step forward. To be helpful, social collaborator robots should be planned and worked by remedial necessities. Following quite a while of examination, Aldebaran Mechanical technology has dispatched a profoundly progressed social humanoid robot called NAO. This bipedal robot, which is 58 cm tall, is one of the pioneers of advanced mechanics with its legitimate plan and mix of programming and equipment. Highlights of the NAO robot incorporate full programming capacity, sensors, underlying PCs, controller ability and light and lovely body. Given the foundation of the issue, there is a need to plan more mind-boggling robots than current robots. These robots can encourage the treatment cycle. Likewise, experts in the field of treatment should be engaged with the plan of calculations and robots. In this investigation, a two-route collaboration between the robot and the patient was set up utilizing the fluffy choice tree. In this communication, the improvement of the infection level is recognized by the robot and the robot settles on its choices considering that the patient has improved. This investigation explicitly centers on the treatment of patients' shared consideration. In such a manner, two calculations for constant eye and head discovery and location were created. After the improvement of these two calculations, situations were intended for testing. As the outcomes show, the tainted kids at first wouldn't speak with the robot, yet in the wake of making developments by the robot, they were pulled into it and consented to partake in the tests. When all is said in done, the utilization of social help robots, particularly those whose conduct can be constrained by kids, can significantly affect improving the social practices of youngsters with the mental imbalance and give a restorative answer for their correspondence issues.

Reference

[1] B. Scassellati, A. Henny, M. Maja, Robots for use in autism research, Annual review of biomedical engineering 14 (2012) 275-294.

[2] A. S. Kasha, A. Zakipour, M. Alemi. M. Meghdari, Design and realization of sing language educational humanoid robot, Journal of intelligent & Robotic System (2018) 1-15.

[3] O. Rudovic, M. Zhang, B. Schuller, R. Picard, Multimodal Active Learning From Human Data: A Deep Reinforcement Learning Approach, In 2019 International Conference on Multimodal Interaction (ICMI'19), October 14–18, 2019, Suzhou, China, ACM, New York, USA.

[4] S. Robinson, L. Goddard, B. Dritschel, M. Wisley, P. Howlin, Executive functions in children with autism spectrum disorders, Brain Cognit 71(3) (2009) 362-368.

[5] S. A. Samadi, R. McConkey, Screening for autism in Iranian preschoolers: contrasting M-CHAT and a scale developed in Iran, Autism Dev Disord 45(9) (2015) 2908-16.

[6] S. M Anzalone, E. Tilmont, S. Boucenna, J. Xavier, A. L. Jouen, N. Bodeau, K. Maharatna, M. Chetouani, D. Cohen, MICHELANGELO Study Group, How children with autism spectrum disorder behave and explore the 4-dimensional (spatial 3D+ time) environment during a joint attention induction task with a robot, Research in Autism Spectrum Disorders 8 No 7 (2014) 814-826.

[7] N. J. Pyun, S. Halima, Y. Nicole, Adaptive haar-like features for head pose estimation, International Conference Image Analysis and Recognition, Springer International Publishing (2014) 94-101.

[8] M. Sikandar Lal Khan, L. Zhihan, L. Haibo, Head orientation modelling: Geometric head pose estimation using monocular camera, The 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing 2013.

[9] B. Scassellati, A Henny. M. Maja, Robots for use in autism research, Annual review of biomedical engineering 14 (2012) 275-294.

[10] R. Varadharajan, M. K. Priyan, P. Panchatcharam et al, A new approach for prediction of lung carcinoma using back propagation neural network with decision tree classifiers, J Ambient Intell Human Comput (2018) 1-12.

[11] S. M. Anzalone, E. Tilmont, S. Boucenna, J. Xavier, A. L. Jouen, N. Bodeau, K. Maharatna, M. Chetouani, D. Cohen, MICHELANGELO Study Group, How children with autism spectrum disorder behave and explore the 4-dimensional (spatial 3D+ time) environment during a joint attention induction task with a robot, Research in Autism Spectrum Disorders 8 No 7 (2014) 814-826.

[12] N. J. Pyun, S. Halima, V. Nicole, Adaptive haar-like features for head pose estimation, International Conference Image Analysis and Recognition, Springer International Publishing (2014) 94-101.

[13] M. Sikandar Lal Khan, L. Zhihan, L. Haibo, Head orientation modelling: Geometric head pose estimation using monocular camera, The 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing 2013.

[14] S. M. Srinivasan, I. M. Eigsti, L. Neelly, A. N. Bhat, The effects of embodied rhythm and robotic interventions on the spontaneous and responsive social attention patterns of children with Autism Spectrum Disorder (ASD): A pilot randomized controlled trial, RASD 27 (2016) 54-72.

[15] F. Abdollahipour, M. Alizadeh, M. A. Fahimi, S. K. Esmaeili, Study of face and content validity of the Persian Version of behavior Rating Inventory of Executive function, J Rehabil 17(1) (2016) 12-19.

[16] D. O. David, C. A. Costescu, S. Matu, A. Szentagotai, A. Dobrean, Developing joint attention for children with autism in robot-enhanced therapy, Int J Soc Robot 10(5) (2018) 595-605.

[17] C.D.C (Centers for Disease Control and Prevention), Prevalence of autism spectrum disorders among children aged 8 years: Autism and developmental disabilities monitoring network, sites, United States 2010.

[18] M. C. Di Lieto, E. Castro, E. Cecchi, F. Cioni, G. Dell'Omo, et al. Educational Robotics intervention on Executive Functions in preschool children: A pilot study. Comput Hum Behav 71 (2017) 16-23.

[19] S. Costa, C. Santos, F. Soares, M. Ferreira, F. Moreira, Promoting interaction amongst autistic adolescents using robots, 32nd Annual International Conference of the IEEE/EMBS, Buenos Aires, Argentina (2010) 3856-3859.

[20] D. Feil-Seifer, M. J. Matarić, Automated detection and classification of positive vs. negative robot interactions with children with autism using distance-based features, Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction; New York, NY: ACM Press (2011) 323-330.

[21] R. W. Picard, Emotion research by the people, for the people, Emotion Review 2 (2010) 250-254.

[22] D. J. Ricks, M. B. Colton, Trends and considerations in robot-assisted autism therapy; 2010 IEEE International Conference on Robotics and Automation (IRCA), (2010) 4354-4359.

[23] M. South, J. J. Diehl, Neurobiology: FMRI, In: Hollander E, Kolevzon A, Coyle J, editors. Textbook of autism spectrum disorders. Arlington, VA: American Psychiatric Publishing (2010) 485-496.

[24] J. Wainer, E. Ferrari, K. Dautenhahn K, Robins, The effectiveness of using a robotics class to foster collaboration among groups of children with autism in an exploratory study, Personal Ubiquitous Computing 14 (2010) 445-455.

[25] E. I. Barakova, Robots for social training of autistic children, Empowering therapists in intensive training programs, In: Abraham et al. editors. Proceedings of IEEE WICT (2011) 14-19.

[26] J. J. Diehl, L. Schmitt, C. R. Crowell, M. Villano, The clinical use of robots for children with autism spectrum disorders: a critical review, Res Aut Spect Dis 6 (2012) 249-62.

[27] D. Feil-Seifer, M. J. Matarić, Using robots to augment (not replace) people in therapeutic settings, Refereed workshop: robotics: science and system (2011) Los Angeles.

[28] J. M. Hollerback, M. T. Mason, H. Christensen, A roadmap for U.S. robotics – from internet to robotics, Updated 2009. http://wwww.us-robotics.us, accessed 1 Mar 2012.

[29] L. D. Riek, Wizard of oz studies in HRI: a systematic review and new reporting guidelines, J Hum Robot Int 1 (2012) 119-136.

[30] J. Wainer, E. Ferrari, K. Dautenhahn, B. Robins, The effectiveness of using a robotics class to foster collaboration among groups of children with autism in an exploratory study, Personal Ubiq Comput 14 (2010) 445-55.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 55–60
Received: 16 December 2021
Accepted: 02 January 2021

Selection of the type of cooling for an overclocked Raspberry Pi 4B minicomputer processor operating at maximum load conditions

Wybór rodzaju chłodzenia dla przetaktowanego procesora mikrokomputera Raspberry Pi 4B pracującego w warunkach maksymalnego obciążenia

Jakub Machowski*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The Raspberry Pi is a computer platform that is widely used in education, has a very large user community and extensive documentation. Therefore, it can be a good and cheap alternative to a traditional computer, a TV streaming device or a console for less demanding games. In the case of observing a lower efficiency of the microcomputer, one of many possibilities of improvement which this device offers is overclocking the processor. It is associated with a proper selection of parameters (voltage, clocking) and software in order to achieve the highest possible performance of the dedicated Raspbian system. However, increasing the work efficiency causes the temperature rise up to the limit values. Therefore, an appropriate, i.e. effective, kind of cooling should be applied. Taking all these circumstances into account, an experiment was developed in which temperature measurements were taken during the maximum processor load on all cores at the clock setting that enabled reaching the highest performance. During the research three cases were considered: without the use of cooling, with passive cooling and with active cooling. The obtained results showed that only the use of active cooling noticeably improves the operating conditions of the device, due to lowering the temperature by about 15°C compared to the situation without cooling or with the use of a passive radiator.

Keywords: Raspberry Pi 4B; overclocking; cooling; stress testing

Streszczenie
Raspberry Pi jest platformą komputerową, która ma szerokie zastosowanie w edukacji, posiada bardzo dużą społeczność użytkowników i bogatą dokumentację. W związku z tym może być dobrą i tanią alternatywą dla tradycyjnego komputera, przystawki do telewizora czy konsoli dla mało wymagających gier. W przypadku odczucia mniejszej wydajności pracy mikrokomputera, jedną z wielu możliwości poprawy tego stanu, które oferuje urządzenie jest przetaktowanie (ang. overclocking) procesora. Wiąże się ono z odpowiednim doborem parametrów pracy (napięcia, taktowania) i oprogramowania dla uzyskania jak najwyższej wydajności działania dedykowanego systemu Raspbian. Jednak zwiększanie wydajności pracy urządzenia powoduje wzrost temperatury aż do osiągnięcia wartości granicznych. W związku z tym należy zastosować odpowiedni, tzn. skuteczny, rodzaj chłodzenia. Uwzględniając wszystkie wymienione okoliczności, opracowano eksperyment, w którym dokonano pomiarów temperatury podczas maksymalnego obciążenia procesora na wszystkich rdzeniach przy ustawieniu taktowania, które umożliwiło uzyskanie największej wydajności Podczas badań rozpatrywano 3 przypadki: bez użycia chłodzenia, z chłodzeniem pasywnym oraz chłodzeniem aktywnym. Na podstawie uzyskanych wyników okazało się, że tylko zastosowanie chłodzenia aktywnego wyraźnie poprawia warunki pracy urządzenia, za sprawą obniżenia temperatury o około 15°C w stosunku do sytuacji bez chłodzenia czy z zastosowaniem radiatora pasywnego.

Słowa kluczowe: Raspberry Pi 4B; przetaktowanie; chłodzenie; testowanie warunków skrajnych

*Corresponding author
Email address: **jakub.machowski@pollub.edu.pl** (J. Machowski)

1. Wstęp

Raspberry Pi jest komputerem jednopłytkowym mającym szerokie spektrum zastosowań m.in. w edukacji. Pracuje on pod kontrolą dedykowanego systemu operacyjnego Raspbian, który bazuje na dystrybucji Debiana [1]. System ten jest wyposażony w prawie wszystkie niezbędne języki programowania takie jak: Python, Scratch oraz C [2, 3]. Zaletą tego mikrokomputera jest jego niska cena i łatwa dostępność dla użytkowników. Mikrokomputery jednopłytkowe pozwalają na budowanie w pełni funkcjonalnych komputerów przy użyciu urządzeń peryferyjnych dołączanych do portów wyjściowych, którymi mogą być porty USB [2, 3]. Podłączyć do nich można myszkę, klawiaturę, słuchawki i monitor. Raspberry Pi obsługuje także urządzenia Bluetooth. Do połączenia się z internetem można wykorzystać transmisję przewodową lub bezprzewodową. Komputer jednopłytkowy jest urządzeniem uniwersalnym, łatwym w użyciu i programowaniu, zawierającym niezbędne zasoby [2, 3]. W prosty sposób można zmienić jego konfigurację: zmniejszyć lub zwiększyć taktowanie czy napięcie procesora. Na rynku istnieje obecnie wiele firm zajmujących się projektowaniem i produkcją

tego typu miniaturowych komputerów, które różnią się wielkością, parametrami czy liczbą możliwych do zastosowania kompatybilnych akcesoriów. Dla firm zajmujących się wytwarzaniem mikrokomputerów jednopłytkowych ich użytkownicy tworzą ważną społeczność, która wymieniając się doświadczeniami, jednocześnie wspiera producenta i wytycza dalsze ścieżki rozwoju tych urządzeń. Powoduje to rosnącą popularność istniejących już modeli oraz pojawianie się ich nowszych ulepszonych wersji. Najliczniejszą liczbę użytkowników ma Raspberry Pi, zaprojektowane i produkowane przez firmę Raspberry Pi Foundation.

Najnowszy i najmocniejszy model Raspberry Pi 4B poszerza paletę zastosowań. W porównaniu do popularnego mikrokontrolera Arduino, Raspberry Pi jest w pełni funkcjonalnym komputerem [2, 3]. Jest on wyposażony w podstawowe narzędzie programistyczne jakim jest powłoka sh. Języki skryptowe cechują się dużą uniwersalnością i nie wymagają dodatkowych narzędzi obciążających procesor, takich jak kompilator, dzięki czemu mogą skoncentrować się na dokładniejszych pomiarach. Popularnymi językami skryptowymi są Perl, oraz zyskujący coraz większą popularność Python. Natomiast język powłoki sh nie jest „pełnowartościowym" językiem, ponieważ ma tylko najprostsze typy danych i dość ograniczony zbiór instrukcji. Pozwala on w łatwy sposób scalić instrukcje uniksowe do jednego pliku. Skrypt sh jest plikiem tekstowym, który wykonuje dowolne polecenia i wyrażenia języka sh. Wywoływany skrypt będzie uruchomiony za pomocą właściwego interpretera, w momencie gdy w pierwszej linii znajdował się będzie następujący wpis: `#!/bin/sh`. Lekki język powłoki doskonale nadaje się do prostych zadań, na przykład do przeprowadzenia procesu obciążenia procesora, wykonania pomiarów temperatury i zapisu danych do pliku [4-6].

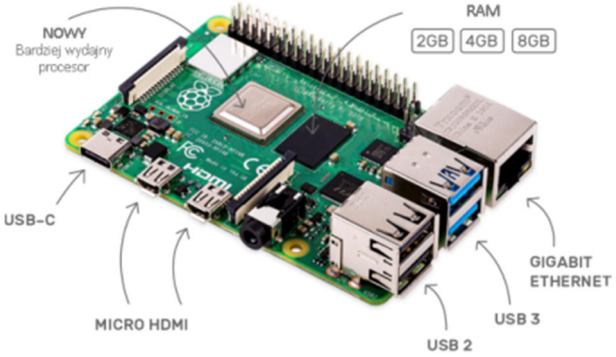
Popularność oraz uwarunkowania licencyjne Linuxa sprawiają, że ciągle powstają nowe dystrybucje tego systemu, które mogą być bezpośrednio uruchamiane z pamięci USB. Dzięki temu system jest mobilny i konfigurowalny, umożliwia zapis danych użytkownika i aplikacji oraz trwałe ich przechowywanie. Ponadto system operacyjny można uruchamiać z poziomu karty MicroSD [5, 7, 8].

Podzespoły komputerowe dostępne obecnie na rynku dysponują zapasem mocy. W związku z tym coraz częściej spotykanym zjawiskiem jest przetaktowanie, które zmusza procesor do stabilnej pracy z wydajnością większą od skonfigurowanej fabrycznie. Przetaktowanie można zrealizować w procesorze, karcie graficznej, pamięci RAM oraz płycie głównej. W przypadku Raspberry Pi zastosowano tę operację na procesorze. W przypadku procesorów AMD do przetaktowania wykorzystuje się oprogramowanie dołączane przez producentów płyt głównych, które upraszcza dokonanie zmiany taktowania [9-12]. Przetaktowanie może być również wykonane sprzętowo poprzez połącznie odpowiednich mostków na płycie głównej lub za pomocą BIOSu i ustawień programowych [9-12]. W Raspberry Pi do zmiany taktowania wykorzystuje się metodę programową bez ingerencji w podzespoły mikrokomputera, poprzez modyfikację ustawień w pliku konfiguracyjnym.

Zwiększanie wydajności procesora pociąga za sobą wzrost zużycia energii [9-12], co wiąże się z większym wydzielaniem się ciepła i w rezultacie skutkuje przegrzewaniem się procesora oraz pozostałych układów znajdujących się na płycie mikrokomputera. Można temu przeciwdziałać stosując chłodzenie, które ma na celu odprowadzenie ciepła wydzielanego podczas pracy urządzenia. Do wyboru są trzy metody chłodzenia: pasywne, aktywne i hybrydowe będące połączeniem dwóch pierwszych. Temperatura Raspberry Pi potrafi osiągać bardzo duże wartości ze względu na gęste rozmieszczenie podzespołów elektronicznych i małe rozmiary samej płytki drukowanej. Temperatura ma wpływ na wydajność i niezawodność urządzeń elektronicznych. Wysoka temperatura może powodować powstawanie naprężeń w konstrukcjach wykonanych z materiałów, o różnych współczynnikach rozszerzalności, zanik naprężeń w kontaktach podzespołów stykowych oraz stopniowe pogarszanie właściwości izolacyjnych materiałów. Na przykład, rezystywność powierzchniowa laminatu szklano-epoksydowego, który stosuje się do produkcji płytek drukowanych, zmniejsza się około 14 razy przy wzroście temperatury o 50°C. Wpływ temperatury na pracę eksploatowanych urządzeń elektronicznych i wywoływanych za jej sprawą uszkodzeń był zagadnieniem, które przez wiele lat nurtowało całe rzesze naukowców. Z przeprowadzonych badań i wieloletnich obserwacji dotyczących niezawodności urządzeń elektronicznych wynika wiele wniosków. Na przykład wzrost temperatury o 10°C dla tranzystorów, o 15°C dla kondensatorów, o 35°C dla rezystorów, dwukrotnie zwiększa ich awaryjność. Z kolei wytrzymałość połączenia lutowniczego maleje dwukrotnie przy zmianie temperatury z 27°C na 70°C [13-15]. Ze względu na tak duży wpływ temperatury na urządzenia elektroniczne, Raspberry Pi posiada wbudowany czujnik temperatury, który umożliwia jej monitorowanie w czasie rzeczywistym. Wytwarzane obecnie, coraz mniejsze i coraz mocniejsze podzespoły elektroniczne, wydzielają znacznie więcej ciepła, dlatego aby wydłużyć ich żywotność stosuje się różne metody chłodzenia.

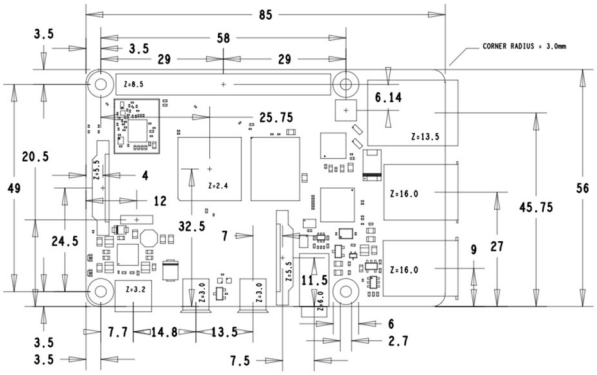
2. Charakterystyka mikrokomputera Raspberry Pi 4B

Raspberry Pi 4B jest najnowszą generacją popularnego mikrokomputera, który ma rozmiar karty kredytowej. Czwarta generacja tego urządzenia oferuje trzy pojemności pamięci RAM LPDDR4: 1 GB, 2 GB oraz 4 GB. Zastosowano w nim nowszej generacji procesor i rdzeń z większym taktowaniem zegara, który oryginalnie wynosi 1,5 GHz. Liczba zastosowanych innowacyjnych zmian jest większa i obejmuje: procesor Broadcom BCM2711 64-bit, rdzeń Quad-Core ARM Cortex-A72, gniazdo zasilania USB C, dwa porty USB 3.0 gniazdo typu A, port Ethernet 100/1000 Mbps, Bluetooth 5.0, dwa złącza microHDMI ze wsparciem H.265 4K 60 kl/s, OpenGL ES 3.0 [16].



Rysunek 1: Wygląd płyty Raspberry Pi 4B; źródło: raspberrypi.org.

Ogólny wygląd płytki PCB (ang. Printed Circuit Board) Raspberry Pi 4B przedstawia rysunek 1. Jej wymiary to długość - 85 mm, szerokość 56 mm oraz wysokość 16 mm. Na rysunku 2 przedstawiono schemat przedstawiający wymiary najważniejszych elementów mikrokomputera. Jest on szczególnie przydatny dla wytwórców obudów czy dodatkowych akcesoriów, takich jak chłodzenie aktywne czy pasywne, które zostało wykorzystane w badaniach przeprowadzonych w ramach tej pracy.



Rysunek 2: Schemat ukazujący wymiary poszczególnych elementów mikrokomputera; źródło: raspberrypi.org.

3. Cel i hipotezy badawcze

Celem przeprowadzonych badań był dobór chłodzenia dla działającego pod maksymalnym obciążeniem przetaktowanego procesora mikrokomputera Raspberry Pi 4B, zapewniającego mu optymalne warunki pracy. Najważniejszym parametrem, który został w badaniach wzięty pod uwagę, i który definiował te warunki była temperatura. Przy stuprocentowym użyciu procesora wydzielało się bardzo dużo ciepła, a temperatura osiągała wartości graniczne, co powodowało resetowanie się urządzenia. W celu przeciwdziałania tej sytuacji zdecydowano się na użycie chłodzenia. W związku z tym przedmiotem badań była weryfikacja trzech trybów pracy urządzenia: bez chłodzenia, z użyciem radiatora oraz z zastosowaniem wentylatora, w celu zmaksymalizowania jego osiągów.

W ramach pracy sformułowano dwie hipotezy badawcze:

1. Zastosowanie pasywnego chłodzenia w postaci radiatora poprawi warunki pracy obciążonego procesora i sprawi, że urządzenie będzie stabilnie pracowało i nie będzie się resetowało.
2. Zastosowanie aktywnego chłodzenia znacznie obniży temperaturę pracy procesora, stworzy stabilne warunki jego pracy i zabezpieczy Raspberry Pi 4B przed powtarzającym się resetowaniem.

4. Metodyka badań

Opracowano eksperyment, polegający na doborze chłodzenia w celu uzyskania jak najlepszych warunków pracy dla obciążonego procesora urządzenia Raspberry Pi 4B. W badaniach uwzględniono trzy scenariusze:

- bez chłodzenia,
- z zastosowaniem chłodzenia pasywnego w postaci radiatora,
- z zastosowaniem chłodzenia aktywnego za pomocą wentylatora.

Podczas eksperymentu przetaktowano procesor do jego granicznych/maksymalnych wartości oraz obciążono algorytmem do wyliczania liczb pierwszych. W badaniach, dla wybranego rodzaju chłodzenia przy maksymalnym obciążeniu procesora, skoncentrowano się głównie na pomiarach i analizie temperatury. Do obciążenia procesora wykorzystano skrypt z zapisem algorytmu obliczeniowego, który był bezpośrednio uruchamiany w powłoce systemowej bash. Dzięki temu można było bezpośrednio wykonywać polecenia bez uruchamiania procesów pochłaniających duże ilości zasobów komputera.

4.1. Stanowisko badawcze

W eksperymencie wykorzystano następujące elementy:

- Raspberry Pi 4B 4GB RAM
- karta microSD SanDisk Ultra A1 16GB
- radiator pasywny
- chłodzenie aktywne Fan Shim
- oficjalna obudowa do Rasperry Pi 4B
- zasilacz USB C 5,1V 3A
- system Raspbian
- skrypt badawczy w bash

4.2. Konfiguracja Raspberry Pi

Podstawową czynnością, którą wykonano w celu poprawienia wydajności Raspberry Pi było prawidłowe ustawienie pliku konfiguracyjnego, który odpowiada za taktowanie procesora, przypisanie pamięci RAM do GPU oraz ustalenie sposobu uruchamiania mikrokomputera. Wszystkie te opcje miały znaczenie i wpływ na przebieg i wyniki eksperymentu. Plik konfiguracyjny (config.txt) w urządzeniach Raspberry Pi używany jest zamiast systemu BIOS. Odczytywany jest on przez GPU przed zainicjowaniem CPU i systemu Raspbian [17]. Plik konfiguracyjny powinien znajdować się na pierwszej partycji (rozruchowej) na karcie microSD w katalogu boot (`/boot/config.txt`). Użytkownik, aby mieć możliwość jego edycji, musi posiadać uprawnienia właściciela (`root`) [18]. Zmiany wprowadzone w pliku konfiguracyjnym zaczną obowiązywać po ponownym uruchomieniu Raspberry Pi. Aktualnie obo-

wiązujące ustawienia można wyświetlić za pomocą następujących poleceń:

- `vcgencmd get_config <config>` - wyświetlenie konkretnej wartości konfiguracji;
- `vcgencmd get_config int` - wyświetlenie listy wszystkich ustawionych opcji konfiguracji liczb całkowitych (różnych od zera);
- `vcgencmd get_config str` - wyświetlenie listy wszystkich ustawionych opcji konfiguracji łańcucha (różna od null).

Istnieje także kilka ustawień konfiguracyjnych, których nie można wyświetlić za pomocą polecenia `vcgencmd`, ponieważ ono ich nie obsługuje. Plik `config.txt` jest odczytywany przez oprogramowanie rozruchowe na wczesnym etapie uruchamiania się systemu Raspbian i z tego względu posiada on prostą budowę. Każde polecenie jest pojedynczą, jednoliniową instrukcją, która wygląda następująco:

```
właściwość = wartość
```

W pliku config.txt istnieje możliwość dodawania komentarzy oraz wyłączania wartości konfiguracyjnych bez konieczności usuwania linii, rozpoczynając wiersz znakiem # [16, 17].

Na potrzeby eksperymentu w pliku konfiguracyjnym zmodyfikowano dwa parametry, a resztę pozostawiono bez zmian z wartościami domyślnymi. Pierwszy parametr odpowiada za regulację napięcia rdzenia procesora w zakresie wartości od -16 do 8, które odpowiadają napięciom z przedziału od 0,8V do 1,4V, z interwałem 0,025V.

```
over_voltage = 6
```

Drugi parametr służy do ustalenia częstotliwości pracy procesora ARM, wyrażonej w megahercach (MHz). Wartość domyślna dla Raspberry Pi 4B to 700 MHz. W eksperymencie dla tego parametru przypisano najwyższą wartość 2147 Hz, stanowiącą maksymalną częstotliwość taktowania.

```
arm_freq = 2147
```

4.3. Opis środowiska testowego

W eksperymencie skupiono się na testowaniu procesora, który był obciążany wykonywaniem obliczeń liczb pierwszych w zakresie do 25 000. Pomysł wykorzystania tego algorytmu zaczerpnięto od autora aplikacji SysBench. Operację tą zrealizowano za pomocą poniższego polecenia:

```
sysbench --test=cpu --cpu-max-prime=25000 run
```

W przypadku Raspberry Pi testy należało wykonać na wszystkich czterech rdzeniach za pomocą polecenia:

```
sysbench --test=cpu --cpu-max-prime=25000
--num-threads=4 run
```

Wykonywanie testu wiązało się z obciążeniem procesora w 100%, co skutkowało wzrostem jego temperatury.

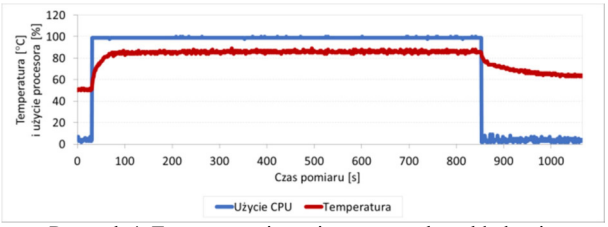
5. Wyniki badań

5.1. Bez zastosowania chłodzenia

Z powodu braku szybkiego i efektywnego odprowadzenia ciepła, podczas pracy w warunkach dużego obciążenia, procesor potrafi osiągać bardzo wysokie temperatury. W momencie przekroczenia granicznej wartości temperatury ustawionej przez oprogramowanie, następuje restart urządzenia. Ponowne uruchomienie lub całkowite wyłączenie Raspberry Pi zabezpiecza podzespoły przed ich trwałym uszkodzeniem. Rysunek 3 przedstawia mikrokomputer bez chłodzenia, użyty w tej części eksperymentu.

Rysunek 3: Mikrokomputer Raspberry Pi 4B bez chłodzenia.

Skrypt testowy został wykonany pomyślnie, obciążając maksymalnie procesor, co przedstawia rysunek 4. W tym przypadku oś X przedstawia kolejne 10660 pomiarów wykonanych w odstępie jednej 1/10 sekundy.



Rysunek 4: Temperatura i użycie procesora bez chłodzenia.

Na wykresie widoczne są wyraźny wzrost i spadek wykorzystania procesora, które odpowiadają momentom rozpoczęcia i zakończenia testu. W spoczynku użycie procesora wynosiło od 2 do 7%. W warunkach dużego obciążenia wykorzystanie procesora wahało się między 98 a 100%.

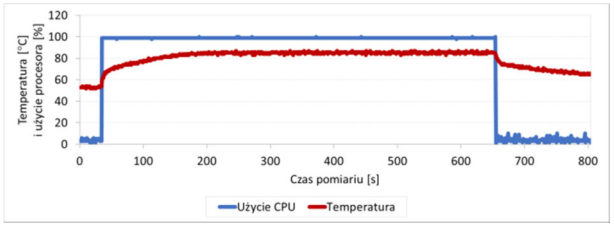
W spoczynku temperatura procesora oscylowała w pobliżu 51°C. W momencie uruchomienia skryptu obciążającego procesor jego temperatura szybko zaczęła wzrastać i utrzymywać się na stałym poziomie 85°C. Dla Raspberry Pi 4B taka temperatura jest wartością krytyczną, o czym informowany jest użytkownik poprzez wyświetlenie w rogu ekranu ikony termometru. Powrót temperatury do stanu początkowego następował bardzo powoli. Temperaturę, która ustabilizowała się na poziomie 64°C, uznano za moment zakończenia pomiaru.

5.2. Chłodzenie pasywne

Pierwszym sposobem, mającym na celu obniżenie zbyt wysokiej temperatury procesora, było zastosowanie chłodzenia pasywnego zrealizowanego za pomocą radiatora wykonanego z tworzywa sztucznego, który umożliwiał łatwiejszą wymianę ciepła z otoczeniem. Na rysunku 5 przedstawiono urządzenie wyposażone w tego typu radiator.

Rysunek 5: Mikrokomputer Raspberry Pi 4B wyposażony w radiator.

Także w tym przypadku do maksymalnego obciążenia procesora wykorzystano skrypt testowy. Tym razem wykonano 8040 pomiary (rysunek 6). W spoczynku użycie procesora osiągało wartości od 2 do 6%, natomiast w warunkach obciążenia wahało się w granicach od 98 do 100%. Wzrost wykorzystania procesora objawiał się emitowaniem dużej ilości ciepła.



Rysunek 6: Temperatura i użycie procesora z chłodzeniem pasywnym.

W spoczynku temperatura procesora oscylowała w granicach 53°C. W chwili, gdy został uruchomiony skrypt obciążający temperatura procesora zaczęła wzrastać i utrzymywała się na poziomie 83°C. W przypadku zastosowania chłodzenia pasywnego osiągnięcie temperatury krytycznej, która dla Raspberry Pi 4B wynosi 85°C, następowało wolniej niż w sytuacji bez użycia chłodzenia. Wskazuje na to mniejsze nachylenie krzywej temperatury od rozpoczęcia maksymalnego obciążenia procesora. Natomiast należy zauważyć, że po wyłączeniu obciążenia, podobnie jak to było w przypadku bez zastosowania chłodzenia, procesor schładzał się wolno.

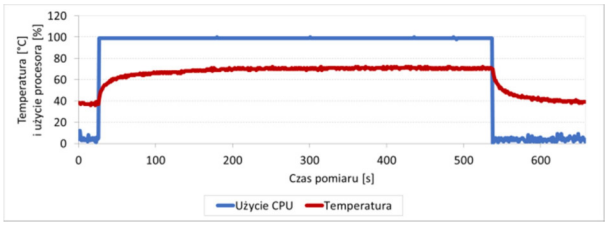
5.3. Chłodzenie aktywne

Skuteczną metodą obniżenia temperatury procesora jest chłodzenie aktywne przy użyciu wentylatora. W badaniu zastosowano chłodzenie aktywne dedykowane dla danego modelu Raspberry Pi. Wentylator powoduje wdmuchiwanie chłodnego strumienia na powierzchnię procesora, natomiast gorące powietrze jest rozpraszane w otoczeniu przez pęd powietrza. Budowę urządzenia z zamontowanym wentylatorem prezentuje rysunek 7.

W tym scenariuszu badawczym wykonano 6580 pomiarów i użyto tego samego skryptu testowego, który maksymalnie obciążał procesor.

Rysunek 7: Mikrokomputer Raspberry Pi 4B z zamontowanym wentylatorem.

Na rysunku 8 widać wyraźne momenty rozpoczęcia i zakończenia testu. W spoczynku użycie procesora osiągało wartości od 1 do 12%. W warunkach obciążenia procesora wartości wahają się od 98 do 100%.



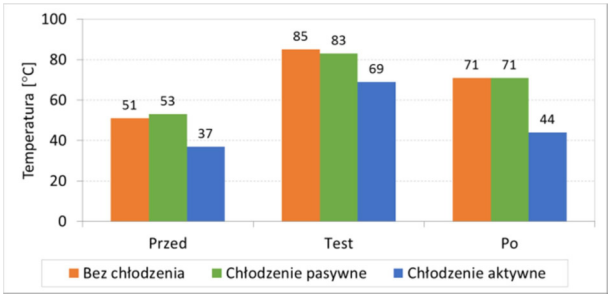
Rysunek 8: Temperatura i użycie procesora z chłodzeniem aktywnym.

W spoczynku temperatura procesora oscylowała w granicach 37°C. W momencie, gdy został uruchomiony skrypt obciążający, jego temperatura zaczęła wzrastać i utrzymywała się w okolicach 71°C. Jest to o 14°C mniej niż wynosi temperatura krytyczna dla Raspberry Pi 4B. Po zaprzestaniu obciążania procesora temperatura ustabilizowała się na poziomie 40°C i dlatego w tym momencie przerwano wykonywanie dalszych pomiarów. Procesor schłodził się dostatecznie szybko do poziomu zbliżonego przed uruchomieniem testu obliczającego liczby pierwsze.

5.4. Dyskusja wyników

Rysunek 9 przedstawia średnie temperatury procesora w trzech fazach: przed obciążaniem, w trakcie testu, po wyłączeniu obciążenia. Z wyników przedstawionych na wykresie można wyciągnąć następujące wnioski:

1. Chłodzenie pasywne nie poprawiło dostatecznie wyraźnie warunków pracy procesora - podczas maksymalnego obciążenia średnia temperatura była zaledwie o 2°C niższa niż w przypadku pomiarów wykonywanych bez użycia chłodzenia.
2. Chłodzenie aktywne wyraźnie obniżyło temperaturę we wszystkich fazach działania procesora. Zaobserwowano spadek o około 14°C w fazie spoczynku przed uruchomieniem testu oraz spadek o około 16°C podczas intensywnej pracy. Po wyłączeniu obciążenia procesor schłodził się bardzo szybko, zmniejszając temperaturę o około 27°C w stosunku do temperatury zmierzonej bez użycia chłodzenia.



Rysunek 9: Zestawienie wyników pomiarów temperatury dla trzech scenariuszy badawczych.

Z wykresów przedstawionych na rysunkach 4, 6 i 8 można wyciągnąć dodatkowe wnioski:

1. Zastosowanie radiatora powodowało wolniejsze nagrzewanie się procesora w czasie pracy pod obciążeniem niż w sytuacji bez zastosowania chłodzenia. Natomiast jego schładzanie po zakończeniu testu obciążeniowego przebiegało w obu przypadkach w podobnym tempie.
2. Użycie wentylatora przyczyniło się do bardzo szybkiego obniżenia temperatury po zaprzestaniu testowania obciążeniowego.

6. Wnioski

Na podstawie przeprowadzonego eksperymentu i uzyskanych wyników okazało się, że maksymalne osiągi urządzenia można uzyskać po zastosowaniu chłodzenia aktywnego. W fazie przed testem, w trakcie jego trwania i po zakończeniu obciążania procesora zaobserwowano w tym przypadku wyraźnie niższą temperaturę niż w dwóch pozostałych scenariuszach badawczych.

Zastosowanie radiatora do chłodzenia procesora nie przyniosło spodziewanych korzyści. Zarówno w sytuacji bez użycia chłodzenia oraz z chłodzeniem pasywnym wyniki tylko nieznacznie się od siebie różniły. W obu tych przypadkach podobne było także tempo schładzania się procesora od chwili zaprzestania obciążania. Pełniejszą wiedzę na ten temat mógłby przynieść pomiar czasów od momentu wyłączenia testu do chwili kiedy procesor osiągnie temperaturę sprzed fazy obciążania. Jednak wykonanie takich badań byłoby trudne w realizacji, ze względu na zjawisko magazynowania ciepła i powolny proces jego oddawania do otoczenia. Przyczyną niewielkiej skuteczności radiatora mogła być niska jakość zastosowanego materiału użytego do zamocowania radiatora na procesorze. Chociaż zamontowany był on prawidłowo, za pomocą fabrycznie dołączonej taśmy termoprzewodzącej, to przewodność cieplna tego połączenia mogła być zbyt niska.

Zrealizowany eksperyment miał na celu weryfikację hipotez postawionych w niniejszej pracy. Na podstawie wyników można stwierdzić, że hipoteza pierwsza nie została potwierdzona, ponieważ zastosowanie chłodzenia pasywnego nie poprawiło warunków działania urządzenia. Natomiast hipoteza druga została potwierdzona, z tego względu, że zastosowanie chłodzenia aktywnego znacznie poprawiło warunki pracy, obniżając temperaturę podczas intensywnej działania procesora o około 16°C i zapobiegło resetowaniu się urządzenia.

Badania wykonane w ramach tej pracy mają pewne ograniczenia. Małe wymiary płytki Raspberry Pi 4B nie pozwalały na zastosowanie większych i cięższych radiatorów lub wentylatorów. Pewnym problemem były również małe rozmiary samego procesora, co uniemożliwiało zastosowanie radiatorów takich, jakie są montowane na typowych procesorach popularnych komputerów. W tej sytuacji w przeprowadzonych badaniach ograniczono się więc tylko do akcesoriów dedykowanych dla Raspberry Pi.

Literatura

[1] Raspbian, https://www.raspbian.org, [21.02.2020].

[2] S. Szabłowski, Raspberry Pi jako środowisko edukacyjne, Uniwersytet Rzeszowski, 2018.

[3] A. Pajankar, Raspberry Pi Supercomputing and Scientific Programming, 2017.

[4] G. J. Nalepa, Podstawy programowania skryptów Sh, Akademia Górniczo-Hutnicza, Kraków, 2000.

[5] R. Blum, C. Bresnahan, Linux Command Line and Shell Scripting Bible, USA, 2008.

[6] C. Johnson, Shell Scripting Recipes. A Problem-Solution Approach, 2005.

[7] J. Kaczmarek, M. Wróbel, Funkcjonalność systemu operacyjnego Linux uruchamianego z pamięci USB, Zeszyty Naukowe Wydział Elektrotechniki i Automatyki Politechniki Gdańskiej 28 (2010) 81-84.

[8] D. Norris, Raspberry Pi: niesamowite projekty: miniaturowy komputer i jego wielka moc!, Helion, Gliwice, 2014.

[9] K. Gązwa, P. Gązwa, A. Sprawka, Overclocking a zużycie energii, Zeszyty Naukowe WSEI. Seria Transport i Informatyka 4 (2014) 19-27.

[10] T. Walsh, How to overclock your PC's CPU, PCWorld, 2017.

[11] PCWorld, Free Speed: Overcloking Your PC, PCWorld, 2005.

[12] E. Rohou, M. D. Smith, Dynamically Managing Processor Temperature and Power, In In 2nd Workshop on Feedback-Directed Optimization (1999).

[13] J. Ćwirko, R. Ćwirko, Skaning temperatury jako narzędzie diagnostyczne modułów elektronicznych, Diagnostyka 4 (40) (2008) 77-80.

[14] B. Chacos, How to check your PC's CPU temperature, PC World, 2005.

[15] Praca zbiorowa pod kier. J. Kijaka, Odporność klimatyczna i wytrzymałość mechaniczna sprzętu elektronicznego, WKiŁ, 1963.

[16] Raspberry Pi, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, [12.02.2020].

[17] W. Gay, Advanced Raspberry Pi. Raspbian Linux and GPIO Integration, 2018.

[18] config.txt, https://www.raspberrypi.org/documentation/configuration/config-txt/README.md, [16.03.2020].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 61–66
Received: 22 September 2020
Accepted: 12 January 2021

Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application

Porównanie wydajności relacyjnych baz danych PostgreSQL oraz MySQL dla aplikacji desktopowej

Bartłomiej Mirosław Klimek*, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

The aim of this paper was to compare the performance of two relational database management systems PostgreSQL and MySQL. For the purpose of the study a relational database was designed and a piece of software was implemented to connect desktop application with the database system. This software shall also creates entities and relations between them in desired empty schemes on servers for databases. There has also been implemented a desktop application in Java programming language, that allows browsing data stored in database and performing the tests of database performance. Tests addressed basic operations of adding, collecting, updating and deleting data. A hypothesis "PostgreSQL is more efficient for desktop application while loaded with small data, in that case 1000 of queries" was confirmed by obtained results.

Keywords: DBMS; MySQL; PostgreSQL; relational databases; comparison of performance

Streszczenie

Celem niniejszej pracy było porównanie wydajności relacyjnych systemów zarządzania bazami danych PostgreSQL i MySQL. Na potrzeby tego badania została zaprojektowana baza danych oraz opracowano i zaimplementowano oprogramowanie mające łączyć aplikację desktopową z poszczególnymi systemami baz danych. Oprogramowanie to ma również tworzyć encje oraz relacje między nimi w wybranych pustych schematach na bazy na serwerach. Zaimplementowano także aplikację desktopowa w języku programowania Java, pozwalającą na przeglądanie zapisanych danych w bazie oraz przeprowadzenie testów wydajności bazy. Testy dotyczyły podstawowych operacji dodawania, pobrania, aktualizacji i usuwania danych. W pracy postawiono hipotezę "PostgreSQL jest bardziej wydajny dla aplikacji desktopowych podczas małego obciążenia danymi, czyli do 1000 zapytań", która została potwierdzona wynikami uzyskanymi z przeprowadzonych badań.

Słowa kluczowe: SZBD; relacyjne bazy danych; porównanie wydajności

1. Wstęp

Obecnie bazy danych towarzyszą ludziom na co dzień, chociaż nie zawsze są tego świadomi. Praktycznie wszędzie spotykają się z aplikacjami korzystającymi z baz danych, czy to przeglądając Internet, czy korzystając z tych desktopowych, zainstalowanych na swoich komputerach.

Baza danych jest w uproszczeniu zwyczajnym skomputeryzowanym systemem przechowywania rekordów, którego głównym celem jest przechowywanie i dostęp do informacji oraz ich aktualizacja [5].

Pierwsze systemy bazodanowe znacznie różnią się od najczęściej używanych w dzisiejszych czasach rozwiązań i były one zapoczątkowane w latach 60 ubiegłego wieku i były to m.in. model sieciowy CODASYL oraz model hierarchiczny o nazwie IMS [6].

Relacyjny model baz danych przedstawił w roku 1970 pracownik firmy IBM Edgar F. Codd, który opublikował poświęconą im pracę. Do tamtej pory głównie używanymi były modele hierarchiczny i sieciowy, a praca ta zrewolucjonizowała przechowywanie danych w komputerach. Autor argumentował poszukiwania nowego modelu "niezależnością danych" i twierdził, że zaprezentowany przez niego model okazał się lepszy od modeli grafowego czy sieciowego [7].

Wraz z rozwojem komputerów systemy informatyczne coraz bardziej się rozrastają, dlatego potrzeba, aby czas dostępu do danych był jak najkrótszy. Dlatego też praca ta skupia się na zbadaniu wydajności dwóch syste-

*Corresponding author
Email address: bartlomiej.klimek@pollub.edu.pl (B.M. Klimek)

mów relacyjnych baz danych PostgreSQL oraz MySQL na przykładzie aplikacji desktopowej.

2. Przegląd literatury

Rozwój technologi powoduje, że pojawiają się prace naukowe dotyczące porównań wybranych zagadnień, w tym systemów bazodanowych, zarówno relacyjnych, jak i nierelacyjnych. Autorzy pracy "Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage" [1] przedstawiają czasowe porównania operacji CRUD (CREATE, READ,UPDATE, DELETE) z podziałem na bazy danych racjonalne i nieracjonalne. W celu przeprowadzenia eksperymentu zostały zaimplementowane dwie analogiczne aplikacje w języku Java, jedna działająca z systemem MySQL, a druga z systemem CouchDB. W artykule przedstawiono dwie struktury danych dla nierelacyjnego systemu CouchDB: pierwsza struktura, w którym każdy dokument musi zawierać odniesienie do innego, a druga struktura, w której każdy dokument zawiera wszystkie dane każdego podmiotu. Druga struktura okazała się bardziej wydajna czasowo niż w podejściu relacyjnym.

Porównanie wydajności czterech systemów relacyjnych z uwzględnieniem wirtualizacji zostało przedstawione w artykule "Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji" [2]. Przeprowadzono serię eksperymentów z użyciem aplikacji, w której zaimplementowane zostały testy do pomiaru czasu wykonania analizowanych instrukcji. Każde zapytanie zostało zmierzone 100-krotnie, a pierwszy pomiar odrzucony. Uzyskane rezultaty potwierdziły hipotezę, że najszybszym systemem bazodanowym jest Oracle.

Wydajność trzech systemów bazodanowych MySQL, PostgreSQL oraz MangoDB wykorzystując aplikację webową napisaną w Django przedstawiono w [3]. W uzyskanych wynikach najbardziej wydajnym systemem okazał się PostgreSQL. W przypadku aplikacji testowej wykonanej na potrzeby tego artykułu, różnica w szybkości między PostgreSQL, a MySQL wynosi około 17%.

Wydajność czterech systemów bazodanowych MySQL, PostgreSQL, MariaDB oraz H2 została przedstawiona w artykule [4]. Porównanie wydajności relacyjnych baz danych polegało na przeprowadzeniu operacji dodania, aktualizacji, usuwania i wybierania danych. Każde badanie zostało wykonane 10 razy dla 1, 100, 2500, 5000, 7500 i 10000 rekordów. MySQL wykazała słabą wydajność podczas pracy z dużą ilością danych. Najlepsze średnie czasy wykonania wszystkich operacji (za wyjątkiem aktualizacji) uzyskała baza danych H2. Wyniki otrzymane dla MySQL, MariaDB oraz PostgreSQL są zbliżone dla operacji wybierania, złączenia i usuwania danych, jednakże PostgreSQL okazał się trochę szybszy od pozostałych dwóch. PostgreSQL wykazał najlepsze wyniki dla operacji aktualizacji.

Artykuł naukowy "Database systems Performance Evaluation for Iot Applications" [8] skupia się na porównaniu wydajności pomiędzy relacyjnymi systemami bazodanowymi MySQL, PostgreSQL i nierelacyjnym systemem MongoDB dla aplikacji w internecie przedmiotów (ang. Internet of Things, IoT). Autorzy przeprowadzili badania na serwerze lokalnym przy użyciu skryptów napisanych w języku Python, aby zminimalizować opóźnienia. Badane scenariusze przewidywały pobranie danych do 500000 rekordów, operacje dodania danych do 500 zapytań, oraz agregację danych dla maksymalnie 10 zapytań.

Według autorów w przypadku małej liczby pobieranych rekordów najlepiej sprawdził się system PostgreSQL, natomiast w przypadku większej liczby pobieranych rekordów (powyżej 20000) MySQL okazał się lepszy niż PostgreSQL, ale gorszy niż MongoDB.

W drugim scenariuszu badawczym przewidującym dodawanie danych dla niewielkiej liczby zapytań najlepiej sprawdził się system MongoDB, drugie miejsce zajął PostgreSQL, a ostatnie MySQL z podobnymi czasami do PostgreSQL. W przypadku większej liczby zapytań najlepiej sprawdził się PostgreSQL.

W przypadku agregacji dla małej ilości danych najlepiej sprawdził się PostgreSQL, natomiast dla większej liczby rekordów najlepszym okazał się MySQL.

Praca naukowa "Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis" [9] skupia się na porównaniu pięciu najpopularniejszych systemów zarządzania bazami danych. Zbadano między innymi operacje dodawania, pobierania i aktualizacji od 6 do 400000 rekordów. Autorzy doszli do wniosków, że najlepiej sprawdził się system SQLite, natomiast drugie miejsce w badaniach zajął PostgreSQL, który był od niego lepszy w przypadkach dodawania rekordów, ale nie dorównał mu w pozostałych przypadkach. Systemy Oracle i SQL Server miały bardzo podobne wyniki, a najgorzej sprawdził się MySQL.

Przedstawiony przegląd literatury dowodzi, że wydajność systemów MySQL oraz PostgreSQL jest aktualnym tematem. Uzyskane w pracach innych wyniki pozwalają na zdefiniowanie tezy "PostgreSQL jest bardziej wydajny dla aplikacji desktopowych podczas małego obciążenia danymi, czyli do 1000 zapytań".

3. Obiekty badań

3.1. PostgreSQL

PostgreSQL jest szybkim, relacyjnym systemem zarządzania bazami danych. W testach porównawczych wyprzedza lub dorównuje osiągom wielu innych systemów przeznaczonych do zarządzania bazami danych, czy to otwartoźródłowych, czy też zastrzeżonych przez prawo [10].

Wspiera dużą część standardu SQL oraz oferuje wiele nowych rozwiązań takich jak złożone kwerendy, klucze obce, wyzwalacze a także inne oraz może zostać rozszerzony przez użytkownika m.in. o nowe typy danych, funkcje, czy operatory [11].

Początki tego systemu sięgają roku 1986, gdy rozpoczął się projekt POSGRES prowadzony przez profesora Michaela Stonebrakera na Uniwersytecie Kalifornijskim, gdzie rozwinął się z projektu Ingres, skąd wywodzi się jego nazwa oznaczająca w wolnym tłumaczeniu "następujący po Ingresie" (ang. post-Ingres). Z czasem projekt zmienił nazwę na Postgres95, a kolejno w roku 1996 na PostgreSQL, która to utrzymała się do dnia dzisiejszego i podkreśla zgodność ze standardem SQL [12].

PostgreSQL wspiera systemy operacyjne takie jak Linux, Windows oraz macOS, oraz jest zgodny ze standardem ACID (ang. Atomicity, Consistency, Isolation, Durability) dla transakcji zapewniając niepodzielność, spójność, izolację i trwałość danych [13].

3.2. MySQL

MySQL jest najpopularniejszym otwartoźródłowym systemem zarządzania bazami danych. Powstał w roku 1995, natomiast od 2010 jest własnością firmy Oracle i jest przez nią rozwijany oraz dystrybuowany w dwóch wersjach: wersji darmowej "The Community Edition", oraz wersji płatnej "Enterprise Edition", która to jest rozszerzona względem wersji darmowej. Wersja bezpłatna w zupełności wystarcza do zapewnienia niezawodności i bezpieczeństwa dla aplikacji [14].

MySQL w wersji 8.0 ma być nową generacją baz danych, dzięki obsłudze SQL jak i NoSQL z notacją JSON, a także wspólne wyrażenia tablicowe, które upraszczają i poprawiają przejrzystość kodu SQL [14].

4. Metoda badań

Do badań została użyta specjalnie na tę potrzebę zaprojektowana aplikacja w języku programowania Java. Do połączenia aplikacji z bazą danych opracowano oprogramowanie REST API które umożliwia na komunikację z serwerem przy pomocy protokołu HTTP oraz wysyłanie zapytań w notacji JSON. API zostało zaprojektowane tak, by można było zbadać operacje dodawania danych oraz ich pobierania z relacjami z inną encją, bądź bez relacji. Można również zbadać operacje aktualizacji danych i ich usuwania, w obu tych przypadkach bez żadnych relacji łączących.

Każdy test zostanie wykonany 10 razy dla 1, 100 i 1000 powtórzeń pętli obsługującej daną operację, dla każdego badanego systemu baz danych. Każde powtórzenie pętli wysyła do bazy jedno zapytanie w notacji JSON. Aby wyniki testów były miarodajne badania zostały przeprowadzone na pustych schematach baz danych, osobnych dla każdego zestawu operacji dla encji bez relacji, kolejno: dodawania, pobrania, aktualizacji i usuwania, a osobno dla encji z relacjami w tym przypadku scenariusze dodawania, a następnie pobierania danych. Po każdym takim zestawie operacji schemat bazy danych został usunięty, a jego miejsce zajmował nowy, pusty schemat. Dla wyników została obliczona średnia oraz odchylenie standardowe w zaokrągleniu do części dziesiętnej.

Wszystkie badania zostały przeprowadzone na bazie danych opracowanej specjalnie na potrzeby aplikacji desktopowej służącej do zarządzania bazą filmów i seriali. Schemat bazy widnieje na rysunku 1. Do badań przeprowadzanych na encji bez relacji została użyta encja 'movie', natomiast do badań dla encji z relacjami zostały użyte tabele 'tvseries' oraz 'tvseassons'. Typ LONGTEXT przy wybranych kolumnach został użyty na wypadek dłuższego opisu, natomiast w badaniach nie został wykorzystany odpowiedni dla tego typu rozmiar danych i mógłby zostać on zastąpiony przez typ VARCHAR.

Do przeprowadzenia badań służył moduł testowy aplikacji przedstawiony na rysunku nr 2.

Aby wykonać wybrany scenariusz konieczne było zaznaczenie pola znajdującego się obok opisu, oraz wybranie liczby powtórzeń w górnej części widoku przedstawionego na rysunku 2. Możliwe jest także wybranie początkowego klucza głównego dla operacji pobierania, aktualizacji i usuwania, jednak wszystkie przeprowadzone badania zostały wykonane od klucza głównego o numerze 1.

Czasy wykonywania wybranego scenariusza zostały zmierzone w aplikacji desktopowej przy pomocy funkcji System.nanoTime() zwracającej czas uruchomionej Wirtualnej Maszyny Javy [15], której wartość została zapisana do zmiennej przed wykonaniem wybranego scenariusza, a po jego wykonaniu do nowej zmiennej została zapisana różnica aktualnej wartości zwróconej przez funkcję System.nanoTime() i zmiennej zawierającej zapisany czas sprzed wykonania scenariusza. Czasy zostały zwrócone w konsoli zintegrowanego środowiska programistycznego.

Badania zostały przeprowadzone na komputerze Asus ROG GL553 z 64 bitowym procesorem Intel Core i5-7300HQ 2.50 GHz, pamięcią RAM 16GB oraz 64 bitowym systemem operacyjnym Windows 10 Home w wersji 1909.

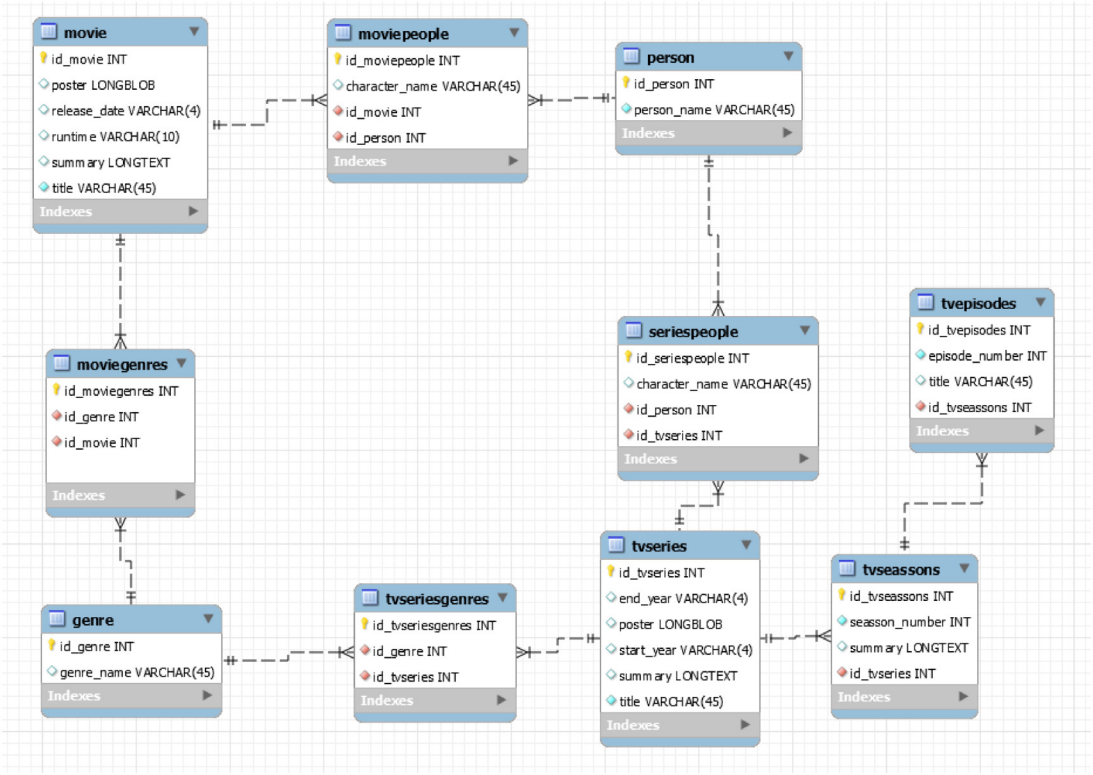
5. Wyniki badań

5.1. Operacja dodania danych

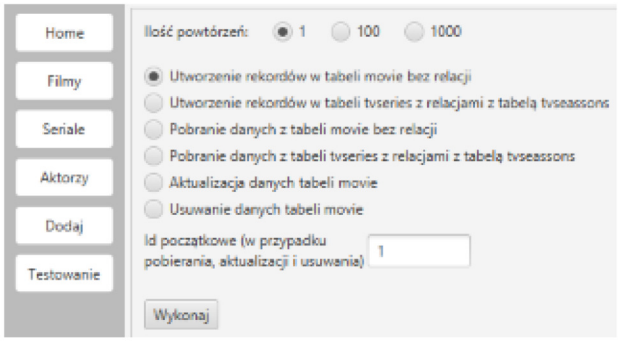
Dla operacji dodawania danych zostały przygotowane dwa scenariusze. Pierwszy przewidujący dodanie danych do encji bez żadnych relacji, a drugi dodanie danych do dwóch encji, które zostaną połączone relacją.

W tabeli 1 przedstawiono średnie czasy oraz odchylenie standardowe dla operacji dodania danych bez relacji dla poszczególnych baz danych. Można tu zaobserwować, że w przypadku PostgreSQL czasy odchylenia standardowego są niższe. Tendencja ta utrzymuje się przy kolejnych scenariuszach.

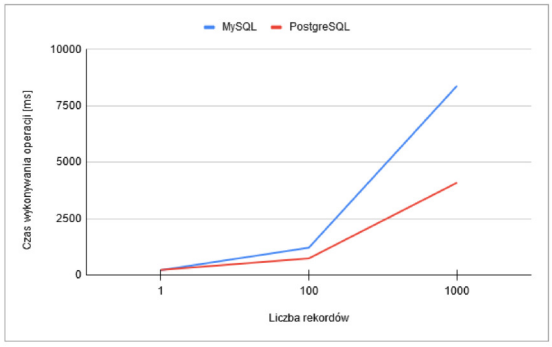
Na rysunku 3 przedstawiono wykres obrazujący średnie czasy wykonania operacji dodawania bez relacji dla poszczególnych baz danych. Widać na nim rosnącą wraz z ilością danych przewagę systemu PostgreSQL, dla której czas przy 100 i 1000 powtórzeń jest o połowę krótszy.



Rysunek 1: Diagram EER (ang. Enhanced entity-relationship).



Rysunek 2: Moduł testowania bazy danych w aplikacji desktopowej.



Rysunek 3: Średni czas operacji dodawania danych bez relacji.

Tabela 1: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji dodania danych bez relacji w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	214,2 ±27, 4	226,2 ±9, 2
100	1211 ±197, 1	736,8 ±29, 8
1000	8390,5 ±158, 9	4095.7 ±83, 2

W przypadku operacji dodawania danych z relacją sytuacja jest podobna, jak w przypadku dodawania danych bez relacji (tabela 2).

Wykres przedstawiony na rysunku 4 obrazuje, że czasy dodania danych z relacjami są większe niż w poprzednim przypadku. Różnica pomiędzy badanymi systemami baz danych w pewnym momencie jest prawie dwukrotna, na korzyść systemu PostgreSQL.

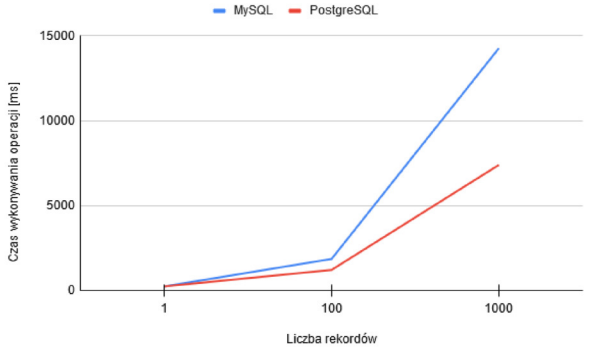
Tabela 2: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji dodania danych z relacją w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	240,4 ±3, 9	243,5 ±7, 2
100	1856,3 ±83, 2	1211 ±44
1000	14272,7 ±1014, 4	7399 ±54, 2

5.2. Operacja pobierania danych

Podobnie do operacji dodawania danych w tym przypadku również zbadano dwa scenariusze. Pierwszy przewidujący pobieranie danych bez relacji, a drugi z relacjami.

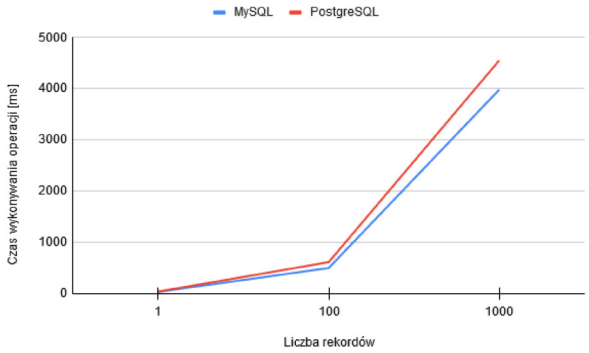
W przypadku pobierania danych bez relacji niewielką przewagę zyskał system zarządzania bazami danych MySQL, a czasy wykonywania operacji są do siebie zbliżone dla obu systemów co można zaobserwować w tabeli 3 i na rysunku 5.



Rysunek 4: Średni czas operacji dodawania danych z relacją.

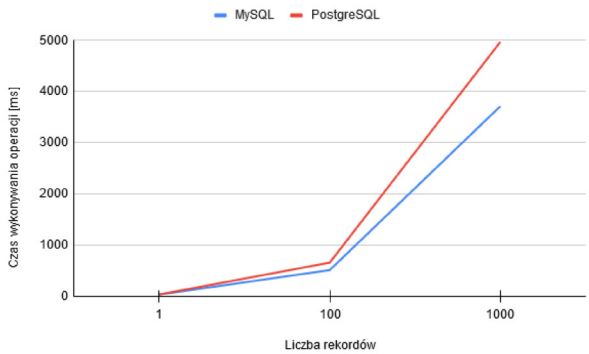
Tabela 3: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji pobierania danych bez relacji w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	32,7 ±1, 8	37,5 ±2, 15
100	498,1 ±31, 3	612,5 ±28, 3
1000	3982,3 ±174, 2	4553,9 ±51, 3



Rysunek 5: Średni czas operacji pobierania danych bez relacji.

W przypadku pobierania danych z relacją sytuacja jest podobna jak w przypadku pobierania danych bez relacji, gdzie przewagę miał system MySQL, natomiast tym razem różnica w czasach jest nieco większa. Wyniki przedstawiono w tabeli 4 i na rysunku 6.



Rysunek 6: Średni czas operacji pobierania danych z relacją.

Tabela 4: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji pobierania danych z relacją w milisekundach.

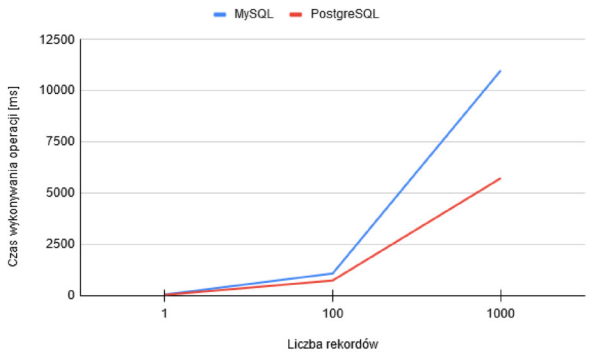
L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	35,2 ±2, 9	37 ±2, 2
100	513,1 ±38, 1	658,2 ±26, 4
1000	3713,4 ±131, 6	4969,7 ±60

5.3. Operacja aktualizacji danych

Operacja aktualizacji danych, której wyniki zostały przedstawione w tabeli 5 i na rysunku 7. Tak jak w przypadku operacji dodawania danych, operacja ta wykonuje się szybciej w przypadku systemu PostgreSQL.

Tabela 5: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji aktualizacji danych bez relacji w milisekundach.

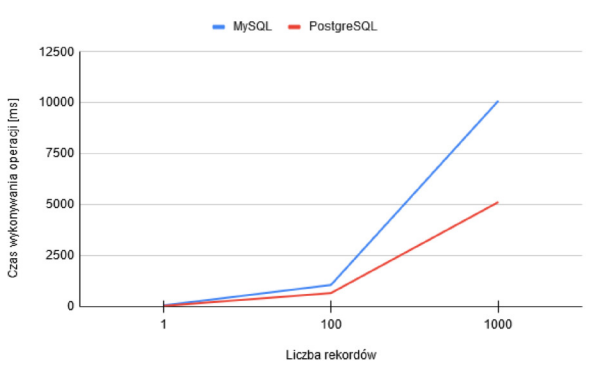
L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	49,4 ±14, 5	31,8 ±2, 9
100	1080,7 ±52, 4	739,7 ±23
1000	11002 ±178, 5	5738,3 ±168, 8



Rysunek 7: Średni czas operacji aktualizacji danych bez relacji.

5.4. Operacja usuwania danych

Operacja usuwania danych, dla której wyniki zostały przedstawione w tabeli 6 oraz na wykresie na rysunku 8 prezentuje wyniki podobne do operacji aktualizacji danych. Ponownie lepsze czasy osiągnął tu system PostgreSQL.



Rysunek 8: Średni czas operacji usuwania danych bez relacji.

Tabela 6: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji usuwania danych bez relacji w milisekundach.

L. rekordów	MySQL	PostgreSQL
1	$56{,}9 \pm 20{,}8$	$33{,}3 \pm 2{,}2$
100	$1060{,}4 \pm 61{,}5$	$663{,}6 \pm 23{,}4$
1000	$10095{,}2 \pm 203{,}8$	$5127{,}4 \pm 88{,}3$

6. Wnioski

W niniejszym artykule przedstawiono badania porównawcze wydajności PostgreSQL i MySQL na podstawie aplikacji desktopowej, napisanej w języku Java. Na potrzeby artykułu stworzono także dwie aplikacje RESTAPI służące do połączenia aplikacji z wybranymi systemami zarządzania bazą danych. Na podstawie przedstawionych wyników badań można wyciągnąć wnioski:

1. W przypadku operacji dodania, aktualizacji i usuwania danych z bazy danych system bazodanowy PostgreSQL okazał się blisko dwukrotnie wydajniejszy.
2. W przypadku operacji pobierania danych z bazy różnice były znacząco mniejsze w porównaniu do innych operacji, lecz lepsze wyniki uzyskał system MySQL.
3. Duże różnice w czasach wykonywania poszczególnych operacji na korzyść systemu PostgreSQL mogą wynikać z jego lepszego przystosowania do notacji JSON [16].
4. Postawiona teza "PostgreSQL jest bardziej wydajny dla aplikacji desktopowej podczas małego obciążenia danymi, czyli do 1000 zapytań" została udowodniona.

Literatura

[1] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, D. Pecherle. Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage, Applied Sciences 10(23) (2020) 8524–8545.

[2] R. Klewek, W. Truskowski, M. Skublewska-Paszkowska, Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji, Journal of Computer Science Institute 16 (2020) 279–284.

[3] B. Nejman, B. Pańczyk, Wydajność pracy z bazami danych aplikacji tworzonych w Django, Journal of Computer Science Institute 11 (2019) 82–85.

[4] K. Krocz, O. Kizun, M. Skublewska-Paszkowska, Analiza porównawcza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2, Journal of Computer Science Institute 14 (2020) 1–7.

[5] C. J. Date, An Introduction to Database Systems, Pearson Education, 2004.

[6] Historia baz danych, https://www.quickbase.com/articles/timeline-of-database-history, [11.09.2020].

[7] Historia relacyjnych baz danych, https://twobithistory.org/2017/12/29/codd-relational-model.html, [12.09.2020].

[8] C. Asiminidis, G. Kokkonis, S. Kontogiannis, Database systems Performance Evaluation for IoT Applications, International Journal of Database Management Systems (IJDMS) 10(6) (2018) 1–14.

[9] Md. I. Hossain, S. Mahmud, T. D. Santa, Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis, Daffodil International University Dhaka, Bangladesh, 2019.

[10] R. Obe, L. Hsu, PostgreSQL: Up and Running. A practical guide to the advanced open source database, Third edition, O'Reilly Media, 2018.

[11] Dokumentacja PostgreSQL, https://www.postgresql.org/docs/11/intro-whatis.html, [11.09.2020].

[12] Dokumentacja PostgreSQL: Historia, https://www.postgresql.org/docs/8.4/history.html, [11.09.2020].

[13] Artykuł porównujący PostgreSQL i MySQL, https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql/, [11.09.2020].

[14] E. Vanier, B. Shah, T. Malepati, Advanced MySQL 8, Packt Publishing Ltd., 2019.

[15] Dokumentacja java: Class System, https://docs.oracle.com/javase/7/docs/api/java/lang/System.html, [18.09.2020].

[16] Blog enterprisedb.com, https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features, [20.09.2020].