Distributed Systems

Course description sheet

Basic information

Field of study Computer Science **Major** - **Organisational unit** Faculty of Computer Science **Study level** First-cycle (engineer) programme **Form of study** Full-time studies **Profile** General academic	**Didactic cycle** 2022/2023 **Course code** WIINFS.li20.02816.22 **Lecture languages** Polish **Mandatoriness** Obligatory **Block** General Modules **Course related to scientific research** Yes

Course coordinator	Krzysztof Zieliński
Lecturer	Krzysztof Zieliński

Period Semester 6	**Method of verification of the learning outcomes** Exam **Activities and hours** Lectures: 30 Laboratory classes: 30	**Number of ECTS credits** 4

Course's learning outcomes

Code	Outcomes in terms of	Learning outcomes prescribed to a field of study	Methods of verification
Knowledge – Student knows and understands:			

Code	Outcomes in terms of	Learning outcomes prescribed to a field of study	Methods of verification
W1	Posiada szczegółową wiedzę w zakresie budowy i implementacji systemów działających w rozproszonym środowisku sieciowym.	INF1A_W04	Examination
W2	Ma elementarną wiedzę na temat cyklu życia systemów komputerowych	INF1A_W04	Examination
W3	Orientuje się w obecnym stanie oraz najnowszych trendach rozwojowych informatyki.	INF1A_W02	Examination
Skills – Student can:			
U1	Potrafi tworzyć efektywne aplikacje pracujące w rozproszonych systemach komputerowych.	INF1A_U07, INF1A_U10	Execution of laboratory classes
U2	Potrafi projektować i implementować oprogramowanie.	INF1A_U05	Execution of laboratory classes
U3	Potrafi ocenić istniejące oprogramowanie, wskazać możliwości jego poprawy i ulepszenia	INF1A_U10	Activity during classes, Execution of laboratory classes
Social competences – Student is ready to:			
K1	Potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania oraz adekwatnie zaplanować pracę.	INF1A_K03, INF1A_K04	Participation in a discussion, Execution of laboratory classes

Program content ensuring the achievement of the learning outcomes prescribed to the module

Moduł wprowadza studenta w podstawy systemów rozproszonych.

Student workload

Activity form	Average amount of hours* needed to complete each activity form
Lectures	30
Laboratory classes	30
Preparation for classes	20
Realization of independently performed tasks	30
Examination or final test/colloquium	2
Contact hours	5

Student workload	**Hours** 117
Workload involving teacher	**Hours** 60

* hour means 45 minutes

Program content

No.	Program content	Course's learning outcomes	Activities
1.	# *Gniazda (4 godz.). Cechy protokołów TCP/UDP.* Zastosowanie portów, definicja gniazda i asocjacji. Kolejność bajtów w przesyle danych liczbowych (Big endian / Little endian). Podstawy programowania gniazd w językach Java i Python. Przykład aplikacji klient-serwer z użyciem protokołów TCP oraz UDP. Zaprojektowanie i implementacja aplikacji do komunikacji sieciowej z użyciem gniazd TCP oraz UDP. # *Komunikacja grupowa (4 godz.).*Definicja problemu. Koncepcja wirtualnej synchroniczności. Problem porządkowania wiadomości. JGroups jako przykład realizacji wirtualnej synchroniczności. Implementacja przykładowej aplikacji. # *Samodzielne prace praktyczne (2 godz.).* - sprawdzenie wiadomości. # *Message Oriented Middleware (MOM) (4 godz.).* Modele komunikacji - synchroniczna oraz asynchroniczna. Cechy komunikacji opartej o wiadomości. RabbitMQ jako przykład realizacji MOM. Cechy i elementy składowe, nawiązywanie połączeń, deklaracja kolejek, publikowanie oraz konsumowanie wiadomości. Niezawodność oraz load-balancing, metody routingu wiadomości. Zaprojektowanie oraz implementacja zaawansowanej aplikacji wykorzystującej poznane mechanizmy RabbitMQ. # *Technologie middleware (4 godz.).* Przeźroczystość w kontekście technologii middleware. OO middleware i RPC middleware. Języki definiowania interfejsów technologii middleware. Technologie Zeroc Ice, Apache Thrift i gRPC - podstawowe informacje, porównanie i prosta aplikacja klient-serwer. Zaprojektowanie, implementacja i demonstracja zaawansowanej aplikacji wykorzystującej middleware. # *Samodzielne prace praktyczne (2 godz.).* - sprawdzenie wiadomości. # *Systemy reaktywne (4 godz.).* Cechy systemów reaktywnych według The Reactive Manifesto. Akka jako przykład platformy do budowy systemów reaktywnych. Cechy platformy Akka. Cechy, hierarchia, cykl życia aktorów. Referencje oraz ścieżki do aktorów, transparentność lokalizacji. Strategie obsługi błędów. Strumienie reaktywne z użyciem Akka Streams API. Zaprojekotwanie oraz implementacja zaawansowanej aplikacji reaktywnej z użyciem Akka. # *Środowisko Zookeeper (4 godz.).* Zookeeper jako przykład środowiska umożliwiającego koordynowanie rozproszonych aplikacji/usług. Pojęcie Znode. Rodzaje Znode - efemeryczny i sekwencyjny. Mechanizm powiadomień. Replikacja stanu serwerów Zookeeper - pojęcia lider i kworum. Gwarancje spójności. Testy działania serwera Zookeeper na pojedynczym urządzeniu oraz w środowisku zreplikowanym. Zaprojektowanie i implementacja przykładowej aplikacji wykorzystującej mechanizm obserwatorów i umożliwiającej sprawdzenie aktualnej ilości potomków oraz uruchomienie zewnętrznej aplikacji. # *Samodzielne prace praktyczne (2 godz.).* - sprawdzenie wiadomości.	U1, U2, U3, K1	Laboratory classes

No.	Program content	Course's learning outcomes	Activities
2.	# *Systemy rozproszone – definicja, pojęcia podstawowe, usługi bazowe w systemach rozproszonych (2 godz.)* Omówienie podstawowych definicji i własności systemów rozproszonych. Przegląd usług transparentności w systemach rozproszonych. # *Pojęcia systemu operacyjnego rozproszonego, sieciowego oraz warstwy pośredniczącej (2 godz.)* Przedstawienie kategorii systemów operacyjnych dla systemów rozproszonych oraz zasad ich budowy. Wprowadzenie pojęcia warstwy pośredniczącej omówienie podstawowych usług zapewnianych przez warstwy pośredniczące. Przykłady warstw pośredniczących. # *Mechanizmy komunikacji w systemach rozproszonych - zdalne wywołanie procedury, zdalne wywołanie metody, przesyłanie komunikatów. Migracja - rodzaje oraz znaczenie w budowie środowisk przetwarzania rozproszonego (2 godz.)* Porównanie różnych mechanizmów komunikacji w systemach rozproszonych. Sposoby implementacji zdalnego wywołania procedury. Model imperatywny a model obiektowy zdalnego wywołania procedury. Techniki przesyłania komunikatów. # *Warstwy pośredniczące obiektowe – OOM, oraz zorientowane na wiadomości – MOM (2 godz.)* Charakterystyka obiektowego przetwarzania rozproszonego. Obiektowe warstwy pośredniczące oparte na mechanizmie zdalnego wywołania metod. Wady i zalety warstw pośredniczących opartych na mechanizmie zdalnego wywołania metod. Przykłady: CORBA, RMI. Charakterystyka warstw pośredniczących zorientowanych na przesyłanie komunikatów. Przykłady MOM: JMS, Active MQ. # *Model Actora - AKKA (2 godz.)* # *Systemy reaktywne, reaktywne przetwarzanie strumieniowe,zasady budowy skalowalych systemów rozproszonych(2 godz.)* # *Kanoniczne problemy systemów rozproszonych – pojęcie czasu, zegary logiczne Lamporta, zegary wektorowe (2 godz.)* Wprowadzenie do kanonicznych problemów systemów rozproszonych. Modele czasu i ich znaczenie w systemach rozproszonych. Protokół NTP. Zegary logiczne – zegar Lamporta. Zegary wektorowe i ich zastosowania. # *Modele spójności w systemach rozproszonych (2 godz.)* Pojęcie spójności danych. Modele spójności: spójność ścisła, spójność sekwencyjna, spójność linearyzowana, spójność przyczynowa, spójność FIFO. # *Replikacja oraz protokoły zarządzanie replikami danych (2 godz.)* Pojęcie globalnego stanu systemu i algorytmy jego wyznaczania. Sposoby wyznaczenia stanu zakończenia obliczenia rozproszonego. Replikacja danych i jej znaczenie dla efektywności i niezawodności dla pracy systemów rozproszonych. Algorytmy utrzymywania konsystentności replik. # *Algorytmu elekcji oraz synchronizacji w środowisku rozproszonym (2 godz.)* Rodzaje algorytmów elekcji: Bully, Ring. Zastosowanie algorytmów elekcji. Algorytmy synchronizacji i wzajemnego wkluczania w systemach rozproszonych, Zookeeper. # *Zagadnienia konstrukcji systemów rozproszonych odpornych na uszkodzenia. Semantyka błędów. Protokoły odtwarzania spójnego stanu systemu rozproszonego (2 godz.)* Omówienie sposobów replikacji systemów rozproszonych. Klastryzacja systemów rozproszonych i zasady jej konfiguracji. # *Problemy uzgadniania stanu w warunkach występowania uszkodzeń – uzgodnienie Bizantyjskie (2 godz.)* Omówienie źródeł problemów związanych z ustaleniem konsystentnego stanu rozproszonego. # *Mechanizmy bezpieczeństwa w systemach rozproszonych (2 godz.)* Omówienie zagrożeń bezpieczeństwa występujących w systemach rozproszonych oraz źródeł ich występowania. # *Teoria CAP i jej znaczenie w budowie aplikacji rozproszonych dużej skali(2 godz.)*	W1, W2, W3	Lectures

Extended information/Additional elements

Teaching methods and techniques :

Lectures

Activities	Methods of verification	Credit conditions
Lectures	Examination	
Lab. classes	Activity during classes, Participation in a discussion, Execution of laboratory classes	

Additional info

Wykłady z przedmiotu będą prowadzone w sposob zdalny z wykorzystaniem platformy Webex/Teams.
Pozostałe zajęcia będą odbywać się w salach. Dotyczy to także zaliczeń i egzaminów odbywajacych się w sesjach egzaminacyjnych.

Conditions and the manner of completing each form of classes, including the rules of making retakes, as well as the conditions for admission to the exam

Warunkiem zaliczenia zajęć laboratoryjnych jest zrealizowanie wszystkich zadań oraz zaliczenie kolokwium. Brak zaliczenia laboratorium nie uniemożliwia przystąpienia do egzaminu.

Method of determining the final grade

Ocena końcowa jest oceną z laboratorium z uwzględnieniem terminu uzyskania zaliczenia: za każdy dodatkowy termin jest obniżana o pół stopnia. Ocena z laboratorium bierze pod uwagę aktywność na zajęciach i przygotowanie do nich, oceny z zadań domowych i kolokwium. Oceny są wyznaczane zgodnie z regulaminem studiów AGH.

Manner and mode of making up for the backlog caused by a student justified absence from classes

Studenci nieobecni na zajęciach laboratoryjnych muszą samodzielnie nadrobić zaległości, oddanie zadania jest możliwe w dodatkowych terminach.

Prerequisites and additional requirements

Znajomość podstaw systemów operacyjnych oraz komunikacji sieciowej.

Rules of participation in given classes, indicating whether student presence at the lecture is obligatory

Wykład: Studenci uczestniczą w zajęciach poznając kolejne treści nauczania zgodnie z syllabusem przedmiotu. Studenci winni na bieżąco zadawać pytania i wyjaśniać wątpliwości. Rejestracja audiowizualna wykładu wymaga zgody prowadzącego.
Ćwiczenia laboratoryjne: Studenci wykonują ćwiczenia laboratoryjne zgodnie z materiałami udostępnionymi przez prowadzącego. Student jest zobowiązany do przygotowania się w przedmiocie wykonywanego ćwiczenia, co może zostać zweryfikowane kolokwium w formie ustnej lub pisemnej. Zaliczenie zajęć odbywa się na podstawie zaprezentowania rozwiązania postawionego problemu. Zaliczenie modułu jest możliwe po zaliczeniu wszystkich zajęć laboratoryjnych.

Literature

Obligatory

1. # Nurkiewicz Tomasz, Reactive Programming with RxJava, O'Reilly Media, Inc, USA, 2016
2. # Raymond Roestenburg,Rob Bakker, Rob Williams, Akka in Action, O'Reilly Media, Inc, USA, 2014
3. # A. Tanenbaum, S. Maatern, “Systemy rozproszone. Zasady I paradygmaty”, Klasyka Informatyki, WNT,2006
4. # M. Henning, S. Vinowski, „Advanced CORBA Programming in C++” , Addison-Wesley, 1999
5. # R. Johson, J. Hoeller, A. Anderson, T, Risberg, C. Sampaleanu, ”Spring Framework. Profesjonalne tworzenie oprogramowania w Javie”, Helion, 2006
6. # G. F. Coulouris, „Systemy rozproszone. Podstawy i projektowanie”, WNT, 1999
7. # W. Grosso, „Java RMI – Design and Building Distributed Systems”, O’Reilly, 2002

Scientific research and publications

Publications

1. # Przemyslaw Dadel, Krzysztof Zielinski:Evolution of Reactive Streams API for Context-Aware Mobile Applications. Computing and Informatics 35(4): 852-869 (2016)
2. # Marek Psiuk, Daniel Zmuda, Krzysztof Zielinski: Distributed OSGi built over message-oriented middleware. Softw., Pract. Exper. 43(1): 1-31 (2013)
3. # Pierre de Leusse, Bartosz Kwolek, Krzysztof Zielinski: A Middleware Infrastructure for Multi-rule-Engine Distributed Systems. ServiceWave 2010: 231-232
4. # Pierre de Leusse, Bartosz Kwolek, Krzysztof Zielinski: A common interface for multi-rule-engine distributed systems. RuleML Challenge 2010

Learning outcomes prescribed to a field of study

Code	Content
INF1A_K03	Potrafi współdziałać i pracować w grupie, przyjmując w niej różne role, ma świadomość odpowiedzialności za pracę własną i za wspólnie realizowane zadania.
INF1A_K04	Potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania oraz adekwatnie zaplanować pracę.
INF1A_U05	Potrafi wykorzystać poznane metody i modele do tworzenia programów o charakterze użytkowym, a także potrafi adekwatnie wykorzystać znane algorytmy i struktury danych w budowie systemu komputerowego.
INF1A_U07	Potrafi ocenić, dobrać i stosować właściwe metody i narzędzia stosowane przy realizacji zadań związanych z budową systemu komputerowego, potrafi ocenić przydatność i korzystać z dostępnych bibliotek oraz komponentów oprogramowania.
INF1A_U10	Potrafi porównać rozwiązania istniejących systemów komputerowych ze względu na zadane kryteria użytkowe i ekonomiczne oraz wskazać możliwości ich ulepszenia
INF1A_W02	Ma szczegółową wiedzę w zakresie podstaw algorytmiki, struktur danych oraz złożoności obliczeniowej, a także w zakresie podstaw teoretycznych budowy wybranych narzędzi i systemów informatycznych.
INF1A_W04	Ma szczegółową wiedzę w zakresie technik i narzędzi implementacyjnych uwzględniających wybrane aspekty budowy oprogramowania, w szczególności systemów baz danych, aplikacji działających w środowiskach sieciowych i budowy interfejsu użytkownika.