

PARKING VIP (płatne 15zł/h)

/ / / / / / / / / /

PARKING

oo  
oo  
oo  
oo  
oo

# Systemy i sieci komputerowe (ISK)

## Pytania na egzamin

- [1. Metody uwierzytelniania użytkowników w systemach komputerowych - sposoby, wady, zalety](#)
- [2. Mechanizmy ochrony danych w systemach operacyjnych](#)
- [3. Wykorzystanie łańcucha bloków do zapewnienia niezmienności rejestrów danych](#)
- [4. Koncepcja sieci sterowanych programowo](#)
- [5. Metody i narzędzia wykorzystywane w opisywaniu procesów biznesowych](#)
- [6. Bezpieczeństwo komunikacji bezprzewodowej i transakcji sieciowych](#)
- [7. Analiza systemów informatycznych z użyciem sieci Petriego](#)
- [8. Weryfikacja modelowa z zastosowaniem logiki temporalnej](#)
- [9. Modelowanie sieci komputerowych z wykorzystaniem przepływów wieloskładnikowych](#)
- [10. Modelowanie i optymalizacji przeżywalnych sieci komputerowych](#)
- [11. Algorytmy rozwiązuje złożone problemy optymalizacyjne - typy, przykłady, wady i zalety.](#)
- [12. Diagnozowanie i monitoring sieci z wykorzystaniem systemu Linux](#)
- [13. Usługi katalogowe systemu Windows 200x Serwer](#)
- [14. Metody dostępu i protokoły komunikacyjne w systemach pamięci masowych](#)
- [15. Metody i narzędzia programistyczne stosowane w symulacji komputerowej](#)
- [16. Wybrany algorytm sztucznej inteligencji wykorzystywany w problemach związanych z Uczenie przez wzmacnianie](#)
- [17. Porównanie podejścia strukturalnego i obiektowego do tworzenia oprogramowania](#)
- [18. Omów zjawisko overfittingu oraz sposoby jego zapobiegania](#)
- [19. Etapy tworzenia systemów analityki biznesowej](#)
- [20. Ciągła integracja i automatyzacja procesu tworzenia oprogramowania.](#)

# 1. Metody uwierzytelniania użytkowników w systemach komputerowych - sposoby, wady, zalety

## Definicja uwierzytelniania



Politechnika Wrocławskiego

## Uwierzytelnianie i autoryzacja

### Uwierzytelnianie – *authentication*

- ▶ Okazanie systemowi komputerowemu „dowodu tożsamości”.
- ▶ Przekonanie programu lub systemu komputerowego o tożsamości osoby usiłującej uzyskać dostęp do systemu.  
Przykłady metod:
  - ▶ karta identyfikacyjna.
  - ▶ klucz sprzętowy.
  - ▶ hasło dostępowe znane tylko właścielowi.
  - ▶ udzielenie odpowiedzi na specjalnie zadane pytania.
  - ▶ analiza głosu lub sposobu pisania na klawiaturze.

### Autoryzacja

- ▶ Uzyskanie dostępu do systemu lub jego specyficznych usług
- ▶ Wiązana często z uwierzytelnieniem, ale jest osobnym, niezależnym procesem.
- ▶ Wymagana przy wykonaniu pewnych operacji, np. kasowania rekordów bazy, dostępie w trybie administratora, itp.

Uwierzytelnianie to proces używany przez firmy w celu zapewnienia, że tylko odpowiednie osoby, usługi i aplikacje z odpowiednimi uprawnieniami mogą uzyskiwać dostęp do firmowych zasobów. Jest to ważna część cyberbezpieczeństwa, ponieważ głównym priorytetem przestępcołów jest uzyskanie nieautoryzowanego dostępu do systemów. Robią to, kradnąc nazwy i hasła użytkowników, którzy mają dostęp. Proces uwierzytelniania obejmuje trzy podstawowe kroki:

- Identyfikacja: Użytkownicy określają swoją tożsamość zazwyczaj za pomocą nazwy użytkownika.
- Uwierzytelnianie: Zazwyczaj użytkownicy udowadniają, że są tym, za kogo się podają, wprowadzając hasło (które powinno być znane tylko dla danego użytkownika), ale w celu zwiększenia bezpieczeństwa wiele organizacji wymaga też potwierdzania tożsamości za pomocą czegoś, co użytkownik posiada (telefon lub token) albo co jest jego cechą (odcisk palca lub skan twarzy).

- Autoryzacja: System sprawdza, czy użytkownicy mają uprawnienia do systemu, do którego próbują uzyskać dostęp.

## Dlaczego uwierzytelnianie jest ważne?

Uwierzytelnianie jest ważne, ponieważ pomaga organizacjom chronić ich systemy, dane, sieci, witryny internetowe i aplikacje przed atakami. Pomaga też ono osobom fizycznym zachować poufność ich osobistych danych, umożliwiając im prowadzenie działań takich jak bankowość lub inwestowanie w trybie online przy mniejszym ryzyku. Gdy procesy uwierzytelniania są słabe, atakującemu łatwiej jest złamać zabezpieczenia konta, odgadując poszczególne hasła lub nakłaniając ludzi do przekazania ich danych uwierzytelniających. Może to prowadzić do następujących zagrożeń:

- Naruszenie zabezpieczeń lub eksfiltracja danych.
- Instalacja złośliwego oprogramowania, takiego jak oprogramowanie wymuszające okup.
- Niezgodność z regionalnymi lub branżowymi przepisami dotyczącymi prywatności danych.

## Jak działa uwierzytelnianie

W przypadku ludzi uwierzytelnianie obejmuje ustawienie nazwy użytkownika, hasła i innych metod uwierzytelniania, takich jak skanowanie twarzy, odcisk palca lub kod PIN. Aby chronić tożsamości, żadna z tych metod uwierzytelniania nie obejmuje bezpośredniego zapisywania w bazie danych usługi. Hasła są przekształcane w skróty (nie są szyfrowane), które są zapisywane w bazie danych. Gdy użytkownik wprowadza hasło, jest ono również przekształcane w skrót, a następnie skróty są porównywane. Jeśli oba skróty są zgodne, udzielany jest dostęp. W przypadku odcisków palca i skanów twarzy informacje są kodowane, szyfrowane i zapisywane na urządzeniu.

## Uwierzytelnianie wieloskładnikowe

Jednym z najlepszych sposobów ograniczenia naruszeń zabezpieczeń kont jest wymaganie dwóch lub więcej metod uwierzytelniania, które mogą obejmować dowolną z wcześniej wymienionych metod. Skutecznym najlepszym rozwiązaniem jest wymaganie dowolnych dwóch z poniższych elementów:

- Coś, co użytkownik zna, zwykle hasło.
- Coś, co użytkownik posiada, na przykład zaufane urządzenie, którego nie można łatwo zduplikować, takie jak telefon lub token sprzętowy.
- Coś, co jest cechą użytkownika, na przykład odcisk palca lub skan twarzy.

Na przykład wiele organizacji prosi o hasło (coś, co użytkownik zna), a dodatkowo wysyła wiadomość SMS z hasłem jednorazowym do zaufanego urządzenia (coś, co użytkownik posiada) przed zezwoleniem na dostęp.

## **Uwierzytelnianie dwuskładnikowe**

Uwierzytelnianie dwuskładnikowe to typ uwierzytelniania wieloskładnikowego, który wymaga dwóch form uwierzytelnienia.

## **Typy metod uwierzytelniania**

W nowoczesnym uwierzytelnianiu proces uwierzytelniania jest delegowany do zaufanego, oddzielnego systemu tożsamości, w przeciwieństwie do tradycyjnego uwierzytelniania, w którym każdy system samodzielnie weryfikuje tożsamości. Nastąpiła też zmiana typu stosowanych metod uwierzytelniania. Większość aplikacji wymaga podania nazwy użytkownika i hasła, ale w miarę jak przestępcy nabrały biegłości w kradzieży haseł, społeczność zajmująca się zabezpieczeniami opracowała kilka nowych metod pomagających chronić tożsamości.

## **Uwierzytelnianie oparte na hasłach**

Uwierzytelnianie oparte na hasłach jest najpowszechniejszą formą uwierzytelniania. Wiele aplikacji i usług wymaga od użytkowników tworzenia haseł zawierających kombinację cyfr, liter i symboli, aby zmniejszyć ryzyko odgadnięcia ich przez przestępca. Jednak hasła stwarzają też wyzwania związane z bezpieczeństwem i użytecznością. Ludziom trudno jest wymyślić i zapamiętać unikalne hasło do każdego konta online, dlatego często używają tych samych haseł ponownie. Atakujący stosują wiele taktyk, aby odgadnąć lub wykraść hasła albo nakłonić ludzi do ich nieumyślnego udostępnienia. Dlatego organizacje odchodzą od haseł na rzecz innych, bezpieczniejszych form uwierzytelniania.

## **Uwierzytelnianie oparte na certyfikatach**

Uwierzytelnianie oparte na certyfikatach to metoda korzystająca z szyfrowania, która umożliwia urządzeniom i osobom identyfikowanie się wobec innych urządzeń i systemów. Dwa typowe przykłady to karta inteligentna albo urządzenie pracownika wysyłające certyfikat cyfrowy do sieci lub serwera.

## **Uwierzytelnianie biometryczne**

W uwierzytelnianiu biometrycznym ludzie weryfikują swoją tożsamość za pomocą cech biologicznych. Na przykład wiele osób używa palca do logowania się na swoich telefonach, a niektóre komputery skanują twarz lub siatkówkę osoby, aby zweryfikować jej tożsamość. Dane biometryczne są też powiązane z konkretnym urządzeniem, więc atakujący nie mogą ich użyć bez uzyskania dostępu do tego urządzenia. Ten typ uwierzytelniania staje się coraz popularniejszy, ponieważ jest łatwy dla ludzi — nie muszą oni niczego zapamiętywać — oraz utrudnia kradzież przestępcom, co czyni go bezpieczniejszym niż hasła.

## **Uwierzytelnianie oparte na tokenach**

W uwierzytelnianiu opartym na tokenach zarówno urządzenie, jak i system generują co 30 sekund nowy unikalny numer nazywany jednorazowym tymczasowym kodem PIN (TOTP). Jeśli numery się zgadzają, system sprawdza, czy użytkownik posiada urządzenie.

### **Hasło jednorazowe**

Hasła jednorazowe to kody generowane dla określonego zdarzenia logowania, które wygasają wkrótce po ich wydaniu. Są one dostarczane za pośrednictwem wiadomości SMS, poczty elektronicznej lub tokenu sprzętowego.

### **Powiadomienia wypychane (push)**

Niektóre aplikacje i usługi używają powiadomień wypychanych do uwierzytelniania użytkowników. W takich przypadkach użytkownicy otrzymują na telefon wiadomość z prośbą o zatwierdzenie lub odrzucenie prośby o dostęp. Czasami ludzie przypadkowo zatwierdzają powiadomienia wypychane, mimo że próbują zalogować się do usług, które wysyłały powiadomienie, dlatego metoda ta jest czasami łączona z hasłami jednorazowymi. W przypadku haseł jednorazowych system generuje unikalny numer, który użytkownik musi wprowadzić. Dzięki temu uwierzytelnianie jest bardziej odporne na wyłudzanie informacji.

### **Uwierzytelnianie głosowe**

W przypadku uwierzytelniania głosowego osoba próbująca uzyskać dostęp do usługi otrzymuje połączenie telefoniczne z prośbą o wprowadzenie kodu lub identyfikację ustną.

### **Uwierzytelnianie a autoryzacja**

Chociaż terminów „uwierzytelnianie” i „autoryzacja” często używa się zamiennie, są to dwie powiązane, lecz odrębne koncepcje. **Uwierzytelnianie potwierdza, że logujący się użytkownik jest tym, za kogo się podaje, a autoryzacja potwierdza, że ma on odpowiednie uprawnienia dostępu do żądanych informacji.** Na przykład pracownik działu kadr może mieć dostęp do poufnych systemów, takich jak lista płac lub akta pracowników, których inni nie mogą zobaczyć. Zarówno uwierzytelnianie, jak i autoryzacja mają kluczowe znaczenie dla produktywności oraz ochrony poufnych danych, własności intelektualnej i prywatności.

Plusy i minusy rozwiązań:

Metoda	Plusy	Minusy
Powiadomienia wypychane (push)	Bezpieczeństwo Wygoda Szybkość Przyjazny użytkownikowi Łatwy w użyciu	Wymaga smartfona i połączenia z Internetem Użytkownik może przez pomyłkę zatwierdzić próbę fałszywego logowania

	Niska cena wdrożenia	Telefon może zostać skradziony lub zgubiony
Klucze bezpieczeństwa U2F/WebAuthn	Duże bezpieczeństwo Łączy dwa silne rozwiązania uwierzytelniające Niezwykle trudny do manipulacji Najlepsza obrona przed atakami typu man-in-the-middle	Wysokie koszty wdrożenia i utrzymania Użytkownicy muszą nosić ze sobą dodatkowy sprzęt Klucz bezpieczeństwa może zostać zgubiony lub skradziony
Link Email	Nie wymaga żadnego dodatkowego sprzętu ani oprogramowania	Nie jest uwierzytelnianiem wieloskładnikowym Podatny na ataki typu man-in-the-middle
Sprzętowe tokeny jednokrotnego użytku	Działa w trybie offline	Duże koszty Użytkownicy muszą nosić ze sobą dodatkowy sprzęt Token może zostać zgubiony lub skradziony
Tokeny jednokrotnego użytku przy użyciu oprogramowania	Działa w trybie offline	Wymaga smartfona Telefon może zostać zgubiony lub skradziony
Kod SMS	Działa w trybie offline Znany większości użytkowników Użytkownicy nie potrzebują smartfonów ani żadnego dodatkowego sprzętu ani oprogramowania	Kosztowny Podatny na ataki na karty SIM Telefon lub karta SIM może zostać zgubiona, skradziona lub uszkodzona
Kod QR	Znane niektórym użytkownikom	Wymaga smartfona i połączenia z Internetem Telefon może zostać zgubiony lub skradziony

Biometria (rozpoznawanie linii papilarnych i twarzy)	Silne uwierzytelnianie biometryczne	Możliwość złamania za pomocą ukrytego odcisku palca (ślady potu, oleju lub innych naturalnych wydzielin na skórze) lub zdjęcia w wysokiej rozdzielczości  Nie możesz zmienić swojego odcisku palca, więc gdy zostanie naruszony, zostanie naruszony na całe życie
Hasło	Nie wymaga żadnych zmian w obecnym systemie bezpieczeństwa	Łatwo zgadnąć Łatwo ukraść Łatwe do złamania Podatny na ataki phishingowe Podatny na ataki typu keylogging

## 2. Mechanizmy ochrony danych w systemach operacyjnych

**Uwierzytelnianie użytkownika:** System operacyjny wymaga od użytkowników uwierzytelnienia się przed uzyskaniem dostępu do systemu (autoryzacji). W tym celu powszechnie używane są nazwy użytkownika i hasła.

- Hasła są przechowywane w postaci zaszyfrowanej, funkcja szyfrująca to funkcja jednokierunkowa,
- wprowadzone hasło podczas logowania może być także szyfrowane tym samym kluczem symetrycznym i porównywane z hasłem przechowywanym
- Hasła można trzymać na urządzeniu przenośnym (np. pendrive) zaszyfrowane kluczem
- Można również wykorzystać hasła jednorazowe/kody SMS

W Unixie:

- Wszelkie restrykcje dotyczące dostępu do urządzeń czy programów wymuszane przez system operacyjny odnoszą się do numeru użytkownika i numeru grupy, a nie do ich nazw. Lista użytkowników znajduje się w pliku **/etc/passwd**
- Hasła użytkowników przeniesione są obecnie do pliku **/etc/shadow**
- plik **/etc/group** zawiera informacje o przynależności użytkowników do grup

**Kontrola dostępu:** System operacyjny wykorzystuje listy kontroli dostępu (ACL) do określenia, którzy użytkownicy lub procesy mają uprawnienia do dostępu do określonych zasobów lub wykonywania określonych działań.

- kontrola uprawnień jakie posiada dany użytkownik/grupa do obsługi danego katalogu/pliku (read, write, execute)

**Szyfrowanie:** System operacyjny może wykorzystywać szyfrowanie do ochrony wrażliwych danych i zapobiegania nieautoryzowanemu dostępowi.

- Zabezpieczenie przed niepowołanym dostępem, wykrywanie modyfikacji danych, bezpieczne nawet po przechwyceniu danych
- Szyfry strumieniowe - Przekształcają tekst jawny w szyfrogram bit po bicie;
- Szyfry blokowe - Przekształcają tekst jawny w szyfrogram blokami o ustalonej (dla danego algorytmu) długości.
- Systemy z kluczem prywatnym (symetrycznym) - Ten sam klucz jest używany zarówno do szyfrowania jak i deszyfrowania.
- Systemy z kluczem publicznym (asymetrycznym) - Korzystają z klucza publicznego w celu szyfrowania danych i klucza prywatnego w celu deszyfrowania.
- Zastosowania: SSH, VPN

**Firewall:** Firewall to urządzenie bądź oprogramowanie, które monitoruje i kontroluje przychodzący i wychodzący ruch sieciowy w oparciu o predefiniowane reguły bezpieczeństwa.

- blokuje nieautoryzowany ruch wchodzący z sieci publicznej do sieci prywatnej, jak i wyciek danych z sieci prywatnej do publicznej

- Posiada wbudowane mechanizmy skanowania ruchu sieciowego bazujące na zestawach określonych wcześniej i przyjętych zasad, składających się z bibliotek – reguł nakazów i zakazów. W momencie, gdy np. dane połączenie przychodzące nie spełnia wymogów określonych w regułach zapory, jest automatycznie blokowane.

**Oprogramowanie antywirusowe:** Oprogramowanie antywirusowe służy do ochrony systemu przed wirusami, złośliwym oprogramowaniem

- podobnie jak firewall, antywirus sprawdza pliki na maszynie komputera pod kątem złośliwego oprogramowania, przez które może dojść do wycieku danych z sieci wewnętrznej/komputera do sieci publicznej

**Aktualizacje i poprawki systemowe:** System operacyjny musi być na bieżąco aktualizowany za pomocą najnowszych poprawek i aktualizacji zabezpieczeń, aby zapobiec wykorzystaniu znanych luk w zabezpieczeniach.

- częste aktualizacje pozwalają na posiadanie jak najbardziej aktualnej bazy zagrożeń, takich jak najnowsze moduły detekcji wirusów dla programów antywirusowych
- “łatki” systemu operacyjnego lub oprogramowania, naprawiające niedoskonałości z poprzednich aktualizacji systemu, przez które system jest narażone na ataki

**Backup:** kopia zapasowa danych, najczęściej wybranych plików i folderów

- **Backup pełny** - kopiowanie wszystkich danych, niezależnie od czasu, kiedy były kopowane po raz ostatni. Czas wykonania kopii bezpieczeństwa jest długi.
- **Backup różnicowy (differential)** - tworzenie kopii zapasowej plików, które zostały zmodyfikowane po ostatnim pełnym backupie. Operacja kopiowania plików trwa stosunkowo krótko, wzrasta w skali tygodnia.
- **Backup przyrostowy (incremental)** - najszybszy sposób wykonania kopii bezpieczeństwa, kopowane są pliki, które zostały zmodyfikowane po ostatnim backupie pełnym lub przyrostowym.

**RAID:** technologia wirtualizacji pamięci masowej, która łączy wiele fizycznych komponentów dyskowych w jedną lub więcej jednostek logicznych w celu redundancji danych, poprawy wydajności lub obu tych celów. Dane są dystrybuowane między dyskami na jeden z kilku sposobów, określanych jako poziomy RAID, w zależności od wymaganego poziomu nadmiarowości i wydajności. Każdy schemat lub poziom RAID zapewnia inną równowagę między kluczowymi celami: niezawodnością, dostępnością, wydajnością i pojemnością. Poziomy RAID wyższe niż RAID 0 zapewniają ochronę przed nieodwracalnymi błędami odczytu sektorów, a także przed awariami całych dysków fizycznych.

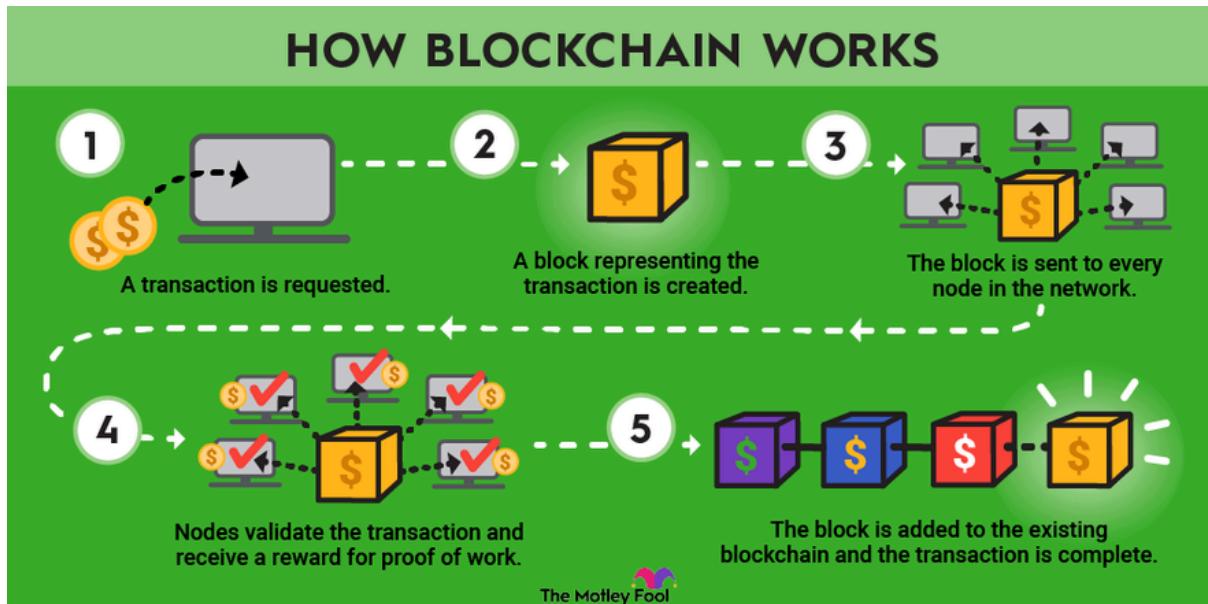
- RAID 0: striping (przeplot), Kilka dysków fizycznych tworzy jeden dysk logiczny przez „przeplot” sektorów, Awaria pojedynczego dysku powoduje z reguły utratę wszystkich danych, niezawodność niższa niż pojedynczego dysku
- RAID 1: mirroring, Zawartość dysków należących do pary jest zawsze identyczna (duplikacja zapisu w lustrze), Pojemność całości jest równa pojemności pojedynczego dysku, w razie awarii dysku dane nie giną
- RAID 0+1: lustro z przeplotem, zapewnia bezpieczeństwo danych i szybki dostęp
- RAID 5: macierz dyskowa (co najmniej 3 dyski, jeden dysk zawiera sumę kontrolną każdego sektora z pozostałych dysków), Podczas awarii jednego z dysków jego zawartość można odtworzyć na podstawie zawartości pozostałych i sumy kontrolnej

z dysku ostatniego, Awaria pojedynczego dysku powoduje degradację systemu do poziomu RAID 0.

- RAID 6: macierz dyskowa (co najmniej 4 dyski), podobne jak RAID 5, ale podwójne sumy kontrolne. Podczas awarii jednego lub dwóch dysków, ich zawartość można odtworzyć na podstawie zawartości pozostałych i sum kontrolnych. Awaria pojedynczego dysku nie powoduje degradacji systemu

### 3. Wykorzystanie łańcucha bloków do zapewnienia niezmienności rejestrów danych

**Blockchain** - łańcuch bloków jest technologią, która służy do przesyłania i przechowywania informacji o transakcjach internetowych. Informacje zostają ułożone w postaci następujących po sobie bloków danych. Jeden blok zawiera informacje o określonej liczbie transakcji, następnie po jego nasyceniu tworzy się kolejny blok danych.



**Bloki ułożone są w łańcuchu chronologicznie i powiązane ze sobą jednokierunkowo** tzn. nowszy blok wskazuje swojego poprzednika w łańcuchu. Otrzymujemy łańcuch, który jest nie modyfikowany i kryptograficznie bezpieczny, czyli nie da się wprowadzić zmian w już istniejących i zaakceptowanych blokach. Jeżeli do któregokolwiek węzła pojawi się nowa transakcja, jest ona rozgłoszana do wszystkich węzłów, następnie transakcje trafiają do bloków. Węzeł który jako pierwszy ogłosi nową prawidłową wersję łańcucha 'wygrywa'.

**Proof-of-Work** to algorytm osiągania konsensusu w łańcuchu bloków. Jego głównymi uczestnikami są górnicy. Używają oni urządzeń sprzętowych do przetwarzania transakcji użytkowników i dodawania ich do bloków. Te ostatnie są następnie włączane do łańcucha bloków. Gdy tworzony jest blok transakcji, zawarte w nim informacje są szyfrowane jako zestaw liczb i symboli – hash. Zadaniem górników jest znalezienie właściwego hasha poprzez obliczenia matematyczne. Pierwszy górnik, który to zrobi, będzie mógł włączyć blok transakcji do łańcucha bloków i otrzymać nagrodę w kryptowalucie.

Każdy blok składa się z:

- znacznika czasu,

- danych transakcji
- kryptograficznego skrótu z treści (haszu) (ang. hash) poprzedniego bloku, dzięki któremu formują one jednokierunkową listę — łańcuch bloków — w której nowo tworzone bloki powiązane są ze wszystkimi wcześniejszymi

#### Cechy systemu:

- Decentralizacja – w celu zatwierdzenia transakcji i dodania jej do łańcucha, wielu uczestników musi ją zatwierdzić zapewniając integralność danych.
- Dystrybucja – blockchain zbudowany jest o sieć rozproszoną. Wykorzystuje rejestr rozproszonych węzłów na wielu serwerach do udostępniania, rejestrowania i zatwierdzania transakcji.
- Niezmienność – po zatwierdzeniu transakcja jest rejestrowana tylko raz i szyfrowana za pomocą skrótu kryptograficznego. Każdy z bloków posiada skrót bloku poprzedniego oraz swój własny

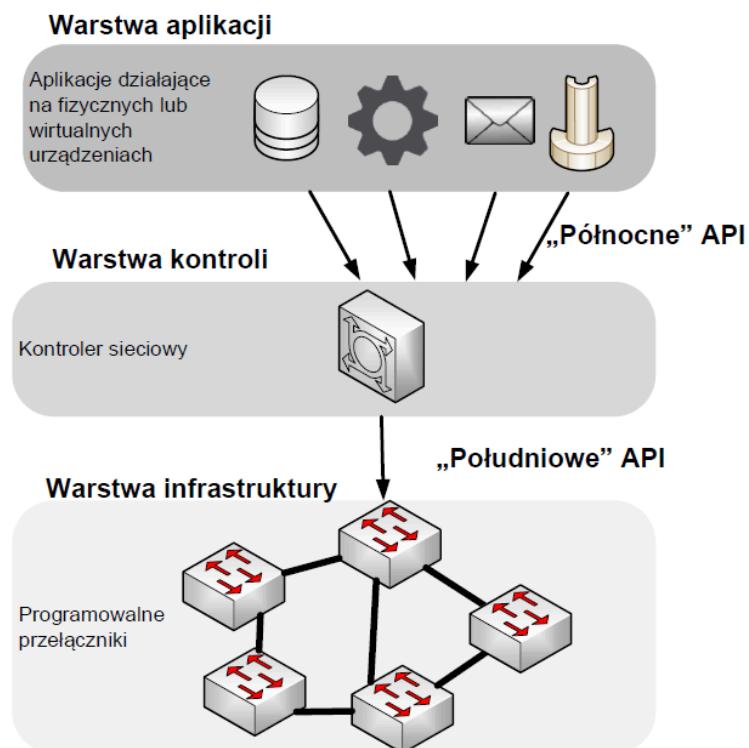
## 4. Koncepcja sieci sterowanych programowo

**Software-Defined Networking (SDN)**, czyli **koncepcja sieci sterowanych programowo** to nowy **paradygmat zarządzania siecią**, który stanowi przełom w tradycyjnym podejściu do projektowania, zarządzania i konfiguracji sieci komputerowych. Jego głównym celem jest **umożliwienie elastycznego i programowanego sterowania infrastrukturą sieciową**, co ma na celu usprawnienie zarządzania zasobami oraz umożliwienie innowacyjnych rozwiązań i eksperymentów także w dziedzinie automatyzacji sieci komputerowych. Architektura zaprojektowana w celu uczynienia sieci bardziej elastyczną i łatwiejszą w zarządzaniu.

Główne powody rozwoju koncepcji SDN:

- Zmiana charakterystyki ruchu w sieciach
- Wzrost popularności rozwiązań cloud computing i rozwój centrów danych
- Przetwarzanie Big Data

Podstawową ideą SDN jest tzw. **decoupling**, czyli **odseparowanie warstwy kontrolującej od warstwy przesyłowej**. W tradycyjnych sieciach te 2 warstwy są zazwyczaj zintegrowane w jednym urządzeniu, natomiast w SDN kontrola sprawowana jest w logicznie skoncentrowanym kontrolerze, który komunikuje się z urządzeniami przesyłowymi (np. przełącznikami) poprzez zdefiniowany interfejs, jak chociażby protokół **Openflow**. Logiczne odseparowanie i przykładową architekturę pokazano na rysunku (od prof. Walkowiaka):



Autorzy przeglądowych prac SDN przedstawiają jako obiecującą koncepcję, która ma zdecydowanie uprościć procesy i odseparować od mechanizmu przesyłania. Na podstawie literatury wyróżniają trzy główne warstwy SDN:

1. Aplikacje sieciowe (**Application Plane**) – odseparowana płaszczyzna, która korzystając z danych umożliwia osiągnięcie określonych celów, takich jak: mechanizmy bezpieczeństwa czy monitorowanie sieci. Aplikacje komunikują się z kontrolerem poprzez interfejs północny warstwy kontrolera.
2. Kontroler SDN (**Control Plane**) – płaszczyzna, która bezpośrednio manipuluje urządzeniami przesyłowymi w celu spełnienia określonych celów zadanych przez aplikacje. Używa interfejsu południowego do połączenia z płaszczyzną danych.
3. Warstwa infrastruktury (**Data Plane**) – część, która obsługuje wspólny protokół (np. OpenFlow) z kontrolerem i przetwarza rzeczywiste żądania i pakiety na podstawie konfiguracji przygotowanej przez kontroler.

Zrozumienie każdej z części jest istotne dla maksymalnego wykorzystania potencjalnych korzyści płynących z SDN, które jest rozumiane jako **filozofia zarządzania siecią**, mogąca mieć przewagę nad tradycyjnym podejściem poprzez **centralizację kontroli i możliwość programowania interfejsów sieciowych**. Wszystko to prowadzi do zwiększenia elastyczności sieci i zdolności do kreatywnego wprowadzania innowacji.

## 4.1. Modele SDN

W wykładzie wskazanych było kilka podejść do SDN:

- **Open SDN** – wykorzystanie OpenFlow do kontrolowania zachowania fizycznych i wirtualnych przełączników.
- **SDN by APIs** – sterowanie przepływem w sieci przez interfejsy API poszczególnych urządzeń.
- **Overlay SDN Model** – tworzenie wirtualnej sieci na istniejącej infrastrukturze sprzętowej; urządzenia końcowe przypisywane do dynamicznych tuneli, którym indywidualnie przydzielana jest przepustowość; sieć fizyczna pozostaje nietknięta.
- Hybrid SDN – model współistniejący z innymi protokołami.

## 4.2. Protokół OpenFlow

Protokół **OpenFlow** jest kluczową propozycją w kontekście Software-Defined Network i odgrywa **istotną rolę w realizacji idei odseparowania warstwy kontrolnej od warstwy przesyłowej**. OpenFlow definiuje standardowy interfejs komunikacyjny pomiędzy kontrolerem SDN a przełącznikami, co umożliwia zdalne programowanie zachowania sieciowych urządzeń przesyłowych. Ustala reguły przesyłania.

Standard ten umożliwia **przeniesienie kontroli nad przekazywaniem pakietów** z pojedynczych urządzeń do **logicznie scentralizowanego kontrolera SDN**. Dzięki temu, decyzje dotyczące przekazywania ruchu sieciowego są podejmowane **globalnie**, co zwiększa elastyczność i efektywność zarządzania siecią. Poza tym za jego pomocą kontroler SDN może zdalnie programować **reguły przekazywania pakietów** w przełącznikach. Umożliwia to szybkie i zdalne dostosowanie się do wymagań **bez konieczności ingerencji w fizyczną infrastrukturę sieciową**. Dzięki zdolności programowania reguł przekazywania pakietów, OpenFlow ułatwia badania i eksperymentowanie nad nowymi algorytmami, protokołami czy usługami sieciowymi. Centralny kontroler, mając spojrzenie globalne na sieć, może optymalizować przekazywanie ruchu, minimalizując przy tym zatory i dostosowując się do zmiennych warunków.

OpenFlow stanowi **otwarty standard**, a zatem jest on niezależny od dostawcy i umożliwia współpracę różnych urządzeń i kontrolerów. Jest często używanym protokołem do implementacji warstwy kontroli. Jedną ze znanych implementacji tego protokołu stanowi **ONOS (Open Network Operating System)**, czyli projekt open-source stworzony w celu dostarczenia elastycznego systemu operacyjnego dla sieci w kontekście SDN i właśnie protokołu Openflow.

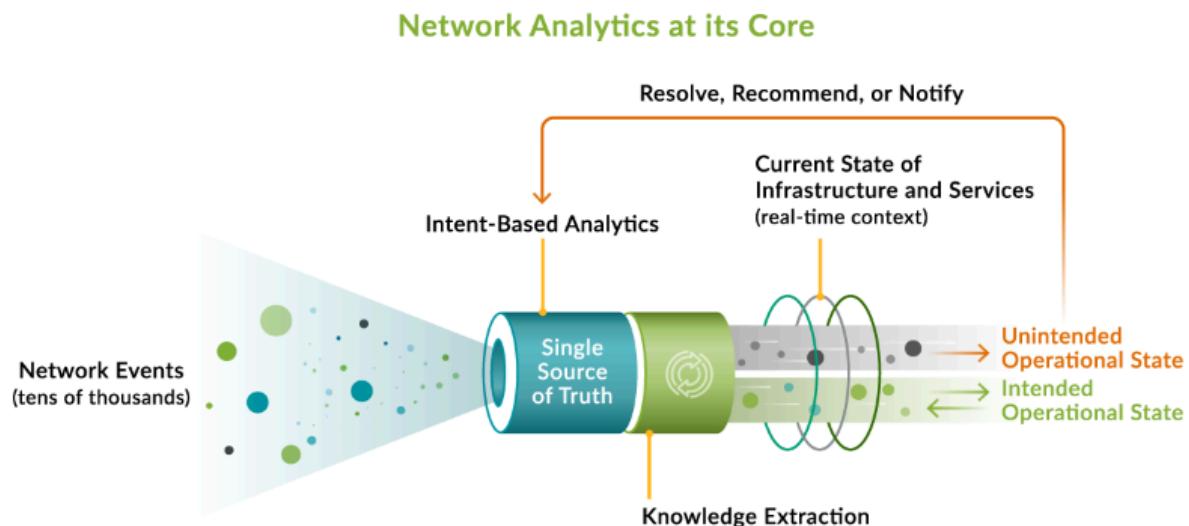
Protokół OpenFlow zapewnia sterownikowi SDN trzy typy informacji:

1. Wiadomości oparte na zdarzeniach: wysyłane przez przełącznik do kontrolera w przypadku zmiany łącza lub portu.
2. Statystyki przepływu: generowane przez przełącznik na podstawie przepływu ruchu.
3. Obudowane Pakiety: wysyłane przez przełącznik do kontrolera, ponieważ istnieje jawną akcję wysłania tego pakietu we wpisie tablicy przepływu lub ponieważ przełącznik potrzebuje informacji do ustanowienia nowego przepływu.

### 4.3. Intent-based network

Kolejnym ważnym aspektem rozwoju sieci sterowanych programowo jest koncepcja **Intent-based Networking (IBN)**. Jej głównym celem jest zmierzać ku bardziej intuicyjnemu i zautomatyzowanemu zarządzaniu sieciami poprzez **interpretację intencji użytkownika**. W tym kontekście, administratorzy sieci definiują **pożądane cele i wymagania**, a system automatycznie przekłada je na **konkretnie akcje konfiguracyjne**. Dzięki IBN, eliminuje się konieczność bezpośredniej interakcji z infrastrukturą sieciową na poziomie technicznym, co przyspiesza procesy oraz redukuje ryzyko błędów ludzkich.

Przykładowy proces bardziej autonomicznego procesu analizy intencji przedstawia Juniper niniejszą grafiką:



### 4.4. Automatyzacja konfiguracji i narzędzi programowe

Bardzo szerokie i dość ogólne pojęcie automatyzacja sieci komputerowych może być interpretowane na co najmniej kilka sposobów ze względu na kontekst, w którym dane narzędzia się analizuje.

W najbardziej ogólnym rozumieniu, kreowanym przez branżę, automatyzacja sieci komputerowych to **proces ułatwiający zarządzanie i konfigurowanie infrastruktury sieciowej za pomocą narzędzi, skryptów lub programów komputerowych**. Jej celem jest zminimalizowanie interwencji ludzkiej oraz skrócenie czasu i ryzyka związanego z ręcznymi operacjami konfiguracyjnymi. Automatyzacja ma upraszczać to, co powtarzalne, co może zostać wykonane w sposób automatyczny i co może chronić przed błędami związanymi z ręcznym wydawaniem komend konfiguracyjnych, przez co **idealnie wpisuje się w koncepcję sieci sterowanych programowo!**

Główni dostawcy rozwiązań sieciowych, jak na przykład Cisco, Juniper czy VMware zdają sobie sprawę z kluczowej roli, jaką odgrywa automatyzacja sieci komputerowych w dzisiejszych, ale i przyszłych dynamicznych środowiskach IT. Każdy z potentatów branży oferuje **własne podejście** i rozumienie tego procesu, dostarczając innowacyjne narzędzia, platformy i rozwiązania, które są dostosowane do różnorodnych potrzeb i wymagań klientów. Jest to jedna z głównych przyczyn różnic w rozumieniu tego zagadnienia – klient. Porównanie sposobów, w jakie ci liderzy myślą i wdrażają swoje rozwiązania w kontekście sieci komputerowych, rzuca światło na zupełnie nowe pomysły, które będą kształtać przyszłość zarządzania sieciami.

**Nie da się wskazać jednolitego podejścia w zakresie automatyzacji sieci komputerowych!**

Obecnie kreowane rozwiązania są mocno **vendor-specific**, czyli charakterystyczne dla danego producenta, dla danej klasy obsługiwanej sprzętu. Fakt ten zdecydowanie definiuje podejście inżynierów do zagadnienia automatyzacji. Coraz bardziej rośnie potrzeba poszukiwania **cech wspólnych**, spójności stosowanych standardów i poszukiwania ścieżek wspólnej konfiguracji między urządzeniami różnych dostawców.

## **I TO BĘDZIE GŁÓWNE WYZWANIE W REALIZACJI KONCEPCJI SDN!**

Istnieje kilka środowisk i narzędzi do implementacji automatyzacji, które wpisują się w sieci sterowane programowo i wyróżnia się przede wszystkim:

- **Ansible** – Umożliwia definiowanie konfiguracji urządzeń sieciowych w formie plików tekstowych (deklaratywne podejście do konfiguracji), co pozwala na łatwą reprodukcję i zarządzanie konfiguracją w różnych środowiskach.
- **Puppet** – Podobnie jak Ansible, umozliwia deklaratywne określanie konfiguracji systemów i aplikacji, co pozwala na utrzymanie spójnego stanu w środowiskach informatycznych, ponieważ narzędzie samo dokona konfiguracji. Puppet, w przeciwieństwie do Ansible, działa w architekturze klient-serwer, co oznacza, że wymaga zainstalowanego oprogramowania klienta (agent) na zarządzanych maszynach.
- **Narzędzia języka Python** – Python oferuje wiele bibliotek i narzędzi do automatyzacji zadań. Największym zainteresowaniem cieszą się biblioteki **Napalm**, **Netmiko** oraz **Paramiko**.
- **Jenkins** – jest narzędziem do ułatwiania **procesów CI/CD** (ang. Continuous Integration/ Continuous Deployment). Pozwala na budowanie, testowanie i wdrażanie konfiguracji.
- I wiele innych...

# 5. Metody i narzędzia wykorzystywane w opisywaniu procesów biznesowych

Proces biznesowy - powiązane ze sobą zadania, które prowadzą do osiągnięcia wyznaczonego efektu

Typy procesów:

- Zarządczy – kieruje działaniem systemu (np. zarządzanie strategiczne)
- Operacyjny – istota biznesu, źródło wartości dodanej (np. produkcja)
- Pomocniczy – wspiera procesy główne (np. księgowość)

## 1. Metody opisywania procesów biznesowych:

**1.1 Notacja BPMN (Business Process Model and Notation)** – graficzna notacja wykorzystywana do modelowania procesów biznesowych

BPMN składa się z kilku podstawowych elementów, które są używane do modelowania procesów biznesowych:

Obiekty Przepływu (Flow Objects)

- Zdarzenia:
  - Zdarzenia początkowe (Start Events): Oznaczają początek procesu.
  - Zdarzenia pośrednie (Intermediate Events): Wskazują na coś, co dzieje się w trakcie trwania procesu.
  - Zdarzenia końcowe (End Events): Oznaczają koniec procesu.
- Aktywności (Activities):
  - Zadania (Tasks): Podstawowa jednostka pracy, wykonywana przez uczestnika procesu.
  - Podprocesy (Sub-processes): Grupowanie zadań, które można rozwijać i ukrywać dla zwiększenia czytelności diagramu.
  - Procesy wywoływane (Call Activities): Odrębne procesy wywoływane przez proces główny.
- Bramki (Gateways):
  - Bramki decyzyjne (Exclusive Gateway - XOR): Używane do przedstawienia decyzji, gdzie tylko jedna ścieżka może być wybrana.
  - Bramki równoległe (Parallel Gateway - AND): Umożliwiają równoległe wykonywanie ścieżek.

- Bramki inkluzywne (Inclusive Gateway - OR): Umożliwiają wybór jednej lub kilku ścieżek.
- Bramki kompleksowe (Complex Gateway): Używane do bardziej złożonych warunków decyzyjnych.

### Obiekty Łączące (Connecting Objects)

- Przepływy sekwencyjne (Sequence Flows): Łączą zdarzenia, aktywności i bramki, określając kolejność wykonywania działań.
- Przepływy wiadomości (Message Flows): Przedstawiają przepływ komunikatów między różnymi uczestnikami procesu.
- Asocjacje (Associations): Łączą obiekty przepływu z obiektami artefaktów.

### Paski Pływania (Swimlanes)

- Baseny (Pools): Reprezentują głównych uczestników procesu, takich jak różne organizacje lub jednostki organizacyjne.
- Tory (Lanes): Podział basenów na mniejsze jednostki, które reprezentują role, zespoły lub systemy.

### Artefakty (Artifacts)

- Obiekty danych (Data Objects): Przedstawiają informacje wykorzystywane lub generowane przez proces.
- Magazyny danych (Data Stores): Przedstawiają miejsca przechowywania danych wykorzystywanych przez proces.
- Notatki (Annotations): Umożliwiają dodawanie dodatkowych informacji tekstowych do diagramu.

**1.2 UML (Unified Modeling Language)**- standardowy język modelowania używany do specyfikacji, wizualizacji, projektowania i dokumentacji systemów informatycznych

UML oferuje szeroki wachlarz diagramów, które można podzielić na dwie główne kategorie:

#### Diagramy Strukturalne

- Diagram klas (Class Diagram): Przedstawia klasy w systemie oraz relacje między nimi. Diagramy klas są podstawowym narzędziem do modelowania struktury systemów obiektowych.
- Diagram obiektów (Object Diagram): Reprezentuje instancje klas w określonym momencie. Używane do wizualizacji stanu systemu.
- Diagram komponentów (Component Diagram): Pokazuje fizyczne komponenty systemu oraz zależności między nimi. Jest używany do modelowania architektury systemów.
- Diagram wdrożenia (Deployment Diagram): Przedstawia fizyczne wdrożenie artefaktów systemu na węzłach, takich jak serwery czy urządzenia sieciowe.
- Diagram pakietów (Package Diagram): Grupuje elementy modelu UML w logiczne pakiety, co pomaga w organizacji dużych systemów.

## Diagramy Behawioralne

- Diagram przypadków użycia (Use Case Diagram): Reprezentuje interakcje między aktorami (użytkownikami lub systemami) a systemem, definiując funkcjonalność, którą system ma dostarczyć.
- Diagram aktywności (Activity Diagram): Modeluje przepływ pracy lub sekwencję działań w procesie biznesowym lub systemie.
- Diagram stanów (State Machine Diagram): Opisuje różne stany obiektu oraz przejścia między tymi stanami w odpowiedzi na zdarzenia.
- Diagram sekwencji (Sequence Diagram): Przedstawia interakcje między obiektemi w formie sekwencji wiadomości wymienianych w określonym czasie.
- Diagram komunikacji (Communication Diagram): Podobny do diagramu sekwencji, ale skupia się na strukturze komunikacji między obiektemi.
- Diagram interakcji (Interaction Overview Diagram): Kombinuje elementy diagramu aktywności i diagramu interakcji, przedstawiając ogólny obraz przepływu kontroli.
- Diagram czasowy (Timing Diagram): Pokazuje zmiany stanu obiektów w zależności od czasu.

**1.3 Diagramy Przepływu Danych (DFD, ang. Data Flow Diagrams)** - narzędzie do modelowania procesów, koncentruje się na przepływie informacji w systemie. DFD są używane do przedstawienia systemów informatycznych w sposób, który jest łatwo zrozumiałym dla analityków biznesowych oraz programistów.

### Podstawowe Elementy DFD:

- Procesy - Procesy są centralnym elementem DFD i reprezentują działania przetwarzające dane. Są zazwyczaj przedstawiane jako okręgi lub prostokąty z zaokrąglonymi narożnikami.
- Przepływy danych - linie z strzałkami, które wskazują kierunek przepływu informacji między elementami diagramu. Opisują, jakie dane są przesyłane i między którymi komponentami systemu.
- Magazyny danych - reprezentują miejsca przechowywania danych w systemie, takie jak bazy danych czy pliki. Są przedstawiane jako otwarte prostokąty lub dwa równoległe linie.
- Zewnętrzne jednostki - elementy poza systemem, które wchodzą w interakcję z systemem poprzez dostarczanie lub odbieranie danych. Są przedstawiane jako prostokąty.

**Poziomy DFD** - DFD są często tworzone na różnych poziomach szczegółowości, co umożliwia stopniowe ujawnianie szczegółów systemu, wyróżnia się poziomy:

- Diagram kontekstowy (Context Diagram) - najwyższy poziom DFD, przedstawiający cały system jako pojedynczy proces oraz jego interakcje z zewnętrznymi jednostkami. Umożliwia zrozumienie ogólnego kontekstu działania systemu.
- Diagram poziomu 0 (Level 0 DFD) - Rozwija diagram kontekstowy, dzieląc główny proces na podprocesy i pokazując przepływy danych między nimi oraz magazynami danych.

- Diagramy poziomu 1, 2, ... (Level 1, 2, ... DFD) - Dalsze rozwinięcie poszczególnych procesów z diagramu poziomu 0, pokazując jeszcze bardziej szczegółowe podprocesy i przepływy danych.

## **2. Narzędzia Wykorzystywane w Opisywaniu Procesów Biznesowych**

Microsoft Visio - narzędzie do tworzenia diagramów, które obsługuje różne typy diagramów procesów biznesowych, w tym schematy blokowe, DFD oraz BPMN.

Lucidchart - narzędzie online do tworzenia diagramów i współpracy w czasie rzeczywistym. Obsługuje różne typy diagramów, takie jak BPMN, UML i schematy blokowe, a także umożliwia współpracę między zespołami.

Bizagi - narzędzie do modelowania i automatyzacji procesów biznesowych, które oferuje intuicyjny interfejs do tworzenia diagramów BPMN. Bizagi umożliwia również symulację procesów oraz integrację z systemami IT.

Visual Paradigm - wszechstronne narzędzie do modelowania, analizy i projektowania systemów informatycznych oraz procesów biznesowych. Obsługuje różne notacje, takie jak UML, BPMN, ERD, i wiele innych, oferując funkcje takie jak generowanie kodu, inżynierię odwrotną oraz współpracę zespołową w czasie rzeczywistym.

# 6. Bezpieczeństwo komunikacji bezprzewodowej i transakcji sieciowych

**Bezpieczeństwo komunikacji bezprzewodowej** odnosi się do ochrony danych przesyłanych za pośrednictwem sieci bezprzewodowych, takich jak Wi-Fi, Bluetooth, oraz sieci komórkowe, przed nieautoryzowanym dostępem, podsłuchem, manipulacją oraz innymi formami ataków. Zabezpieczenia obejmują różne techniki, takie jak szyfrowanie, autoryzacja, uwierzytelnianie, oraz monitorowanie ruchu sieciowego, mające na celu zapewnienie poufności, integralności i dostępności danych przesyłanych bezprzewodowo.

**Bezpieczeństwo transakcji sieciowych** to ochrona danych osobowych i finansowych przesyłanych podczas transakcji online przed nieautoryzowanym dostępem, oszustwami, kradzieżą tożsamości oraz innymi zagrożeniami cybernetycznymi. Obejmuje stosowanie protokołów szyfrowania, takich jak SSL/TLS, tokenizację danych płatniczych, uwierzytelnianie wieloskładnikowe oraz systemy wykrywania i zapobiegania oszustwom, które mają na celu zapewnienie bezpiecznego przetwarzania i przechowywania danych w trakcie i po zakończeniu transakcji.

(z chatu) zabezpieczenia ze szczegółami:

## 1. Szyfrowanie:

- **WEP, WPA, WPA2, WPA3:** Standardy szyfrowania stosowane w sieciach Wi-Fi, mające na celu ochronę danych przed nieautoryzowanym dostępem. WPA3 jest najnowszym i najbezpieczniejszym standardem.
- **SSL/TLS:** Protokoły zabezpieczające połączenia internetowe i szyfrujące dane przesyłane między przeglądarką a serwerem.

## 2. Autoryzacja i uwierzytelnianie:

- **Hasła:** Użycie silnych, unikalnych haseł do zabezpieczania dostępu do sieci bezprzewodowych.
- **Certyfikaty cyfrowe:** Stosowanie certyfikatów do uwierzytelniania użytkowników i urządzeń.
- **Dwuskładnikowe uwierzytelnianie (2FA):** Dodatkowa warstwa zabezpieczeń, która wymaga dwóch różnych form weryfikacji tożsamości użytkownika.

## 3. Integralność danych:

- **Kody autoryzacji wiadomości (MAC):** Stosowane do zapewnienia, że dane nie zostały zmodyfikowane w trakcie transmisji.
- **Hashe:** Algorytmy haszujące, takie jak SHA-256, które pomagają w weryfikacji integralności danych.

## 4. Zarządzanie siecią:

- **Segmentacja sieci:** Podział sieci na mniejsze segmenty, aby ograniczyć ryzyko w przypadku naruszenia bezpieczeństwa.
- **Monitorowanie i analiza ruchu:** Stosowanie narzędzi do monitorowania sieci i wykrywania nieprawidłowości w ruchu sieciowym.

#### 5. Ochrona przed zagrożeniami:

- **Zapory sieciowe (firewall):** Filtrują ruch sieciowy, blokując potencjalnie niebezpieczne połączenia.
- **Systemy wykrywania i zapobiegania włamaniom (IDS/IPS):** Narzędzia do identyfikacji i neutralizacji zagrożeń w sieci.
- **Antywirusy i antymalware:** Ochrona przed złośliwym oprogramowaniem.

Do bezpieczeństwa transakcji można dodać:

#### 1. Uwierzytelnianie i autoryzacja:

- **Hasła i PIN-y:** Podstawowe metody zabezpieczania kont użytkowników.
- **Biometria:** Stosowanie odcisków palców, skanów twarzy lub tęczówki do weryfikacji tożsamości użytkownika.
- **Dwuskładnikowe uwierzytelnianie (2FA):** Wymaganie dodatkowego dowodu tożsamości oprócz hasła.

#### 2. Bezpieczne protokoły płatności:

- **3D Secure:** Protokół stosowany przez wiele kart kredytowych do dodatkowej weryfikacji transakcji online.
- **Tokenizacja:** Proces zastępowania wrażliwych danych płatniczych unikalnym tokenem, który może być używany tylko raz.

#### 3. Monitorowanie i wykrywanie oszustw:

- **Analiza zachowań użytkowników:** Wykrywanie nietypowych aktywności, które mogą wskazywać na próbę oszustwa.
- **Systemy wykrywania oszustw (FDS):** Narzędzia automatycznie identyfikujące i blokujące podejrzane transakcje.

Ataki i zagrożenia występujące w komunikacji bezprzewodowej i transakcjach sieciowych:

#### **Man in the Middle**

- Przechwycenie niezbędnych do wykonania transakcji danych
- Kradzież środków lub tożsamości
- Przechwycenie komunikacji i przekonanie użytkownika, że strona, na którą się loguje (spreparowana) jest stroną na którą chce się zalogować
- Za pomocą otrzymanych danych można uzyskać dostęp do konta użytkownika

- Aby się uchronić – sprawdzać certyfikaty stron, nie wykonywać transakcji w sieci publicznej
- W bankach wprowadzono ochronę w postaci kodów SMS

### **Spoofing MAC**

- Jedną z metod weryfikacji urządzeń w sieciach bezprzewodowych jest weryfikacja po adresie fizycznym karty sieciowej MAC
- Podслушаивание transmisji w sieciach bezprzewodowych jest proste, więc można odczytać adres MAC
- Następnie podmienić własny adres na jeden z nich
- Aby się uchronić – oprócz weryfikacji adresów MAC używać zabezpieczeń hasłem, nadawać dostęp do zasobów dla użytkowników z dostępem do lokalnej sieci

### **Sniffing transmisji**

- Podsłuchanie transmisji nieszyfrowanych
- Uzyskanie danych (login/hasło), przez co można się podszywać pod osobę
- Aby zrobić atak wystarczy oprogramowanie (np. Wireshark)
- Aby się uchronić – skonfigurować sieć aby wymuszała szyfrowanie danych
- Nawiązywać komunikację w wyższych warstwach w sposób szyfrowany (SSL)

### **Evil Twin**

- Podstawienie urządzenia (np. WiFi Pineapple) które symuluje Access Point
- Jeśli będzie wysyłać mocniejszy sygnał od prawdziwego nastąpi przełączenie bez informowania użytkownika
- Ruch z komputera przechodzi wtedy przez fałszywy AccessPoint
- Gdy w telefonie zostaje włączone Wifi telefon wyszukuje dostępne wokół niego sieci oraz odpytuje czy są zapamiętaną przez nie siecią – a urządzenie WiFi Pineapple zawsze odpowie – TAK :) – telefon łączy się z fałszywym Access Pointem
- Aby się uchronić – łączyć ze sprawdzonymi sieciami, wyłączać wifi w telefonie

## **Złamanie zabezpieczeń WEP/WPA**

- Należy używać lepszych zabezpieczeń , np. szyfrowania WPA2 (używające algorytmu AES)

*(AES - algorytm szyfrowania blokowego, szyfruje dane w blokach o stałej długości 128 bitów, używając kluczy kryptograficznych o długości 128, 192 lub 256 bitów. Jest uznawany za bardzo bezpieczny i wydajny, dlatego jest szeroko stosowany w różnych aplikacjach do ochrony danych)*

## **Modyfikowanie pakietów**

- Jeśli transmisja zostanie przechwycona to możliwe jest zmodyfikowanie adresu z którego została wysłana wiadomość
- Dlatego możliwe staje się przechwycenie odpowiedzi serwera
- Aby się uchronić konieczne jest zapewnienie integralności pakietów i jej weryfikacja

## **Przechwytywanie sygnału bluetooth:**

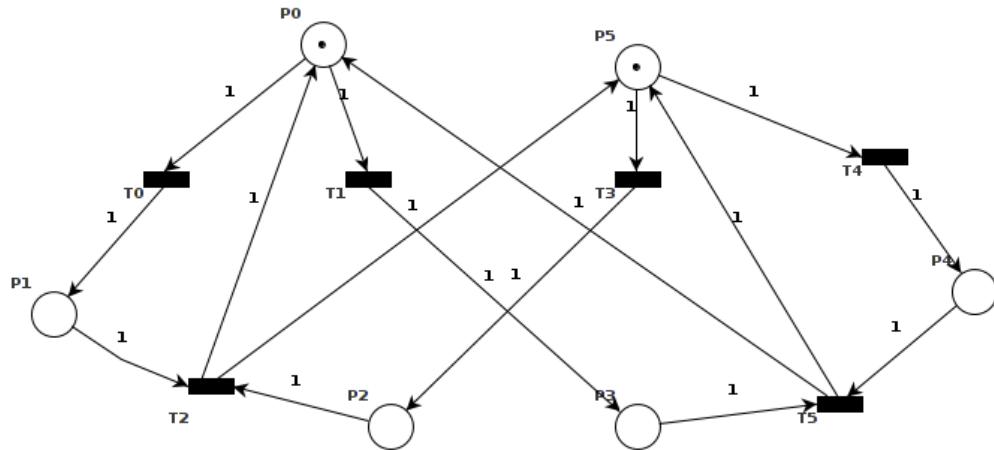
- Klawiatury bezprzewodowe – atakujący może dysponować urządzeniem które będzie przechwytywało wciśnięte przyciski
- Po odpowiedniej analizie pozwala to na wyodrębnienie loginów i haseł

## **Social engineering**

- Jest to sposób na pozyskanie informacji istotnych z punktu bezpieczeństwa
- Polega na manipulowaniu ludzką lekkomyślnością w celu osiągnięcia korzyści, np. fałszywe e-maile, podglądanie wpisywanych z klawiatury haseł

## 7. Analiza systemów informatycznych z użyciem sieci Petriego

Dobra, definicję to każdy se może przeczytać na wikipedii, nie będę tego przeklejał jak debil. To zagadnienie było na miasi na jednych, lub 2 labach. Popatrz na rysunek i sobie przypomnij co tam się działo.



Pamiętasz? No ja też tak średnio. Najpewniej obrona będzie polegać tak, że losujesz temat i masz o nim opowiadać. To lecimy po kolej co warto powiedzieć.

Czym są sieci petriego?

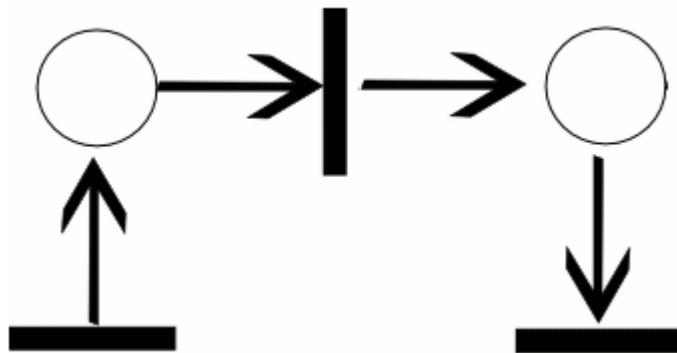
Sieci petriego są narzędziem służącym do modelowania komunikacji z automatami, na dzień dzisiejszy są wykorzystywane do modelowania systemów współbieżnych, dyskretnych, synchronizacji procesów i podobnych.

Jak działają?

Zanim powiemy jak działają, to musimy wiedzieć z czego się składają:

1. miejsca (białe okręgi)
2. przejścia (czarne belki)
3. krawędzie skierowane (strzałki od miejsca do przejścia, lub przejścia do miejsca)
4. żetony (kropki w okręgach)

Teraz jak te sieci działają, tak na chłopski rozum?

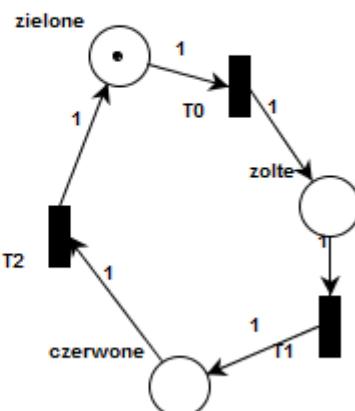


Ten gif świetnie to tłumaczy.

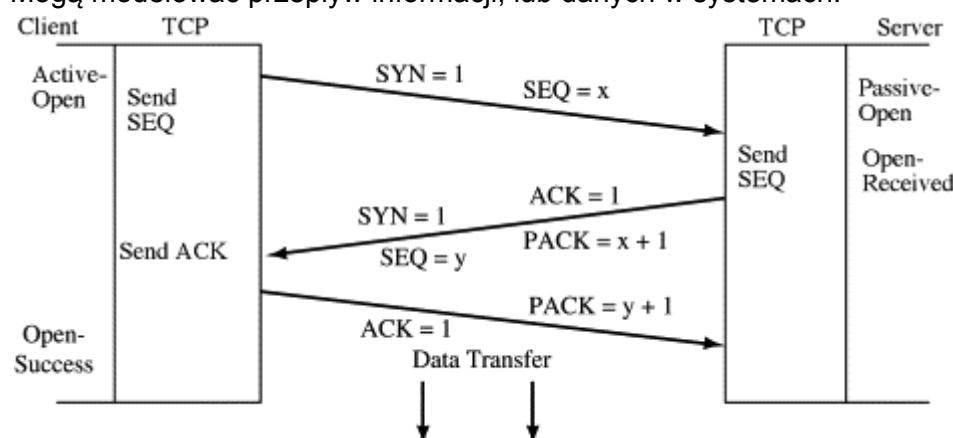
1. Po sieci poruszają się żetony
2. Żeton się porusza przez przejścia
3. przejście jest "aktywne", gdy miejsca które wchodzą do przejścia mają wystarczająco żetonów - można wykonać przejście
4. przejście jest "nieaktywne", gdy przynajmniej jedno miejsce wchodzące w przejście ma za mało żetonów - nie można wykonać przejścia
5. krawędź może mieć wagę (domyślnie 1, może być 2, 3, 4 itp. )
6. Jeśli krawędź ma wagę 2 i wchodzi w przejście - potrzeba 2 żetony w miejscu, które wchodzi w przejście, żeby było aktywne
7. Jeśli krawędź ma wagę 2 i wychodzi z przejścia - w miejscu gdzie prowadzi pojawią się 2 żetony

To by było tyle z teorii jak to działa na sucho, ale o co w tym w zasadzie chodzi? Co opisuje sieć?

W zasadzie to co chcemy, jest to język, a w językach można powiedzieć wiele różnych rzeczy. Sieci petriego mogą opisywać automat stanów, tylko, że gorzej niż graf przejść. Przykładowo niżej sieć petriego opisująca światła uliczne.

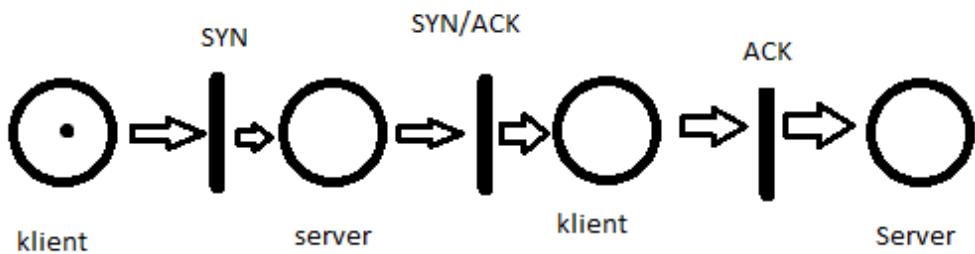


Mogą modelować przepływ informacji, lub danych w systemach.



Na rysunku jest schemat 3 way handshake klasyczne nawiązanie połączenia po TCP.  
Można to zamodelować przy użyciu sieci petriego, gdzie każda strzałka to byłoby przesłanie

informacji między serwerem a klientem i vice versa. Miejsca natomiast to byłyby kolejne stany.



Proszę. Właśnie zamodelowaliśmy 3 way handshake przy pomocy sieci petriego. Źeton to moment w którym znajduje się handshake w scenariuszu poprawnego nawiązania połączenia. Tylko po co to robić?

Jeśli chcesz poćwiczyć dodaj przejście tak, żeby sieć obsługiwała też odpowiedzi RST i zamykanie komunikacji.

## FAQ

### **Skąd się biorą żetony w sieci?**

Może być sieć, która ma obieg zamknięty. Wtedy żetony są tam wsadzane przy tworzeniu sieci i sobie latają w obiegu. Mogą być też przejścia bez wejścia, czyli nie ma miejsc, które wchodzą w przejście. Wtedy jest ono zawsze aktywne i gdy je wykonujemy pojawiają się żetony w sieci,

### **Czy zawsze krawędzie łączą miejsca i przejścia?**

Tak. Krawędź musi być między miejscem a przejściem, albo na odwrót. Nigdy cokolwiek innego. (*memotechnika - sieci petriego są hetero, bo zawsze kółko łączy się z belką a belka z kółkiem, nigdy kółko z kółkiem i nigdy belka z belką*)

### **Co w takim razie modelują tak konkretnie?**

Jajco, to co sobie wymyślisz to zamodelujesz. Baw się

### **Co oznacza, że kropka jest w kółku?**

Oznacza to, że żeton jest w danym miejscu i nic więcej. To co to oznacza musisz sobie sam wymyślić, bo nie oznacza to nic konkretnego

### **Czy w takim razie jeśli kropka jest w danym miejscu, to znaczy, że automat jest w jakimś stanie?**

Może tak być, ale możesz też sobie założyć coś innego. Te sieci są zupełnie nieintuicyjne

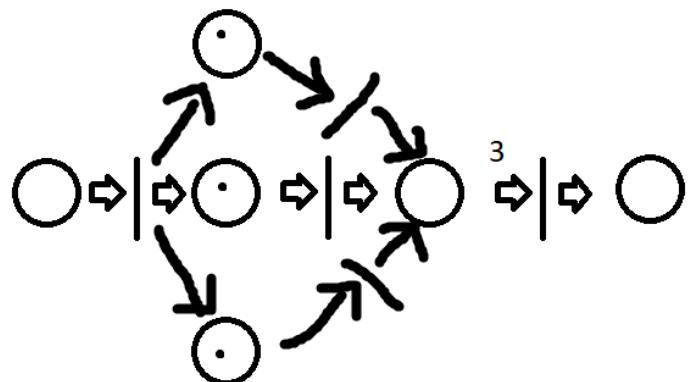
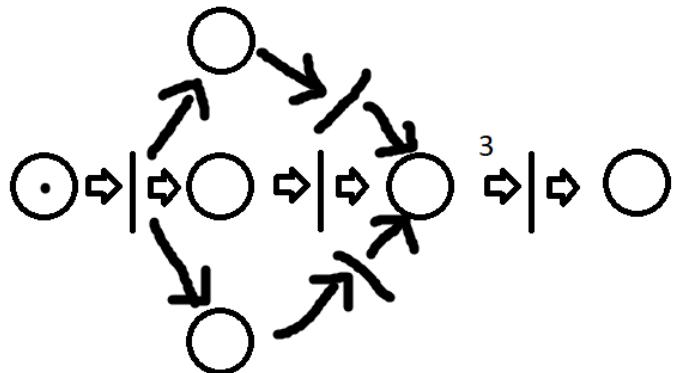
### **Jakie jest w takim razie najlepsze zastosowanie sieci petriego?**

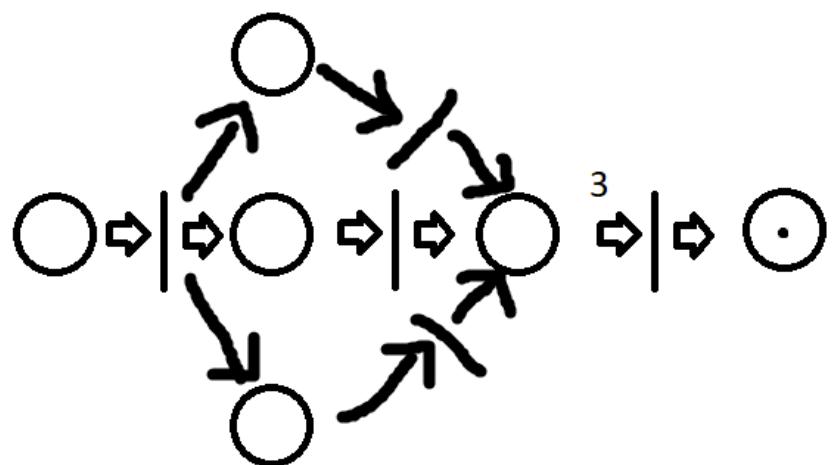
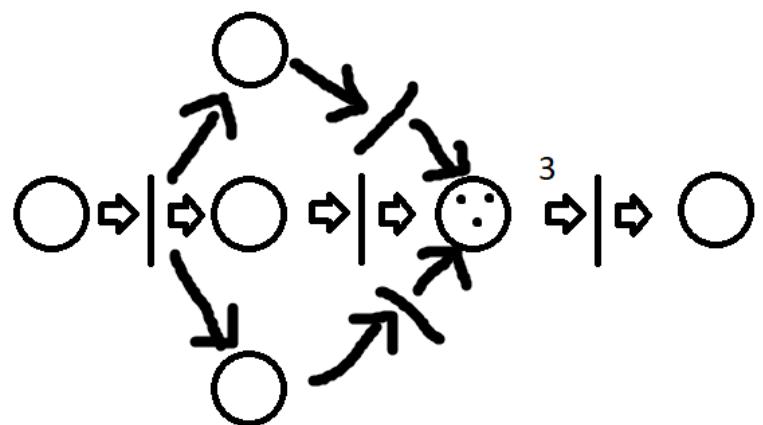
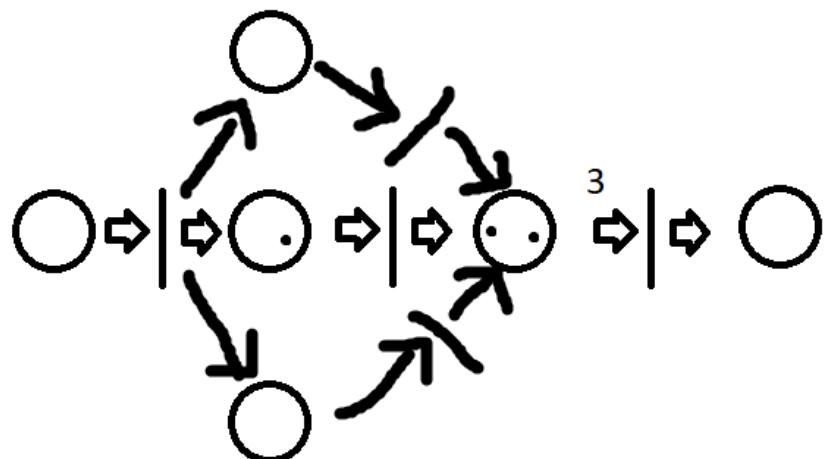
Moim zdaniem to łamigłówki logiczne. Coś jak sudoku, albo krzyżówka. Można zrobić w symulatorze sieć petriego i tam sobie kliknąć, żeby doprowadzić do jakiegoś stanu. Całkiem fajne i ciekawe, ale chyba nic więcej z tego nie będzie.

### **Co mam powiedzieć jak się zapytają jak wykorzystać sieci petriego w przemyśle?**

Możesz opisać, któryś z przykładów wyżej. Możesz też powiedzieć o modelowaniu wielowątkowych programów i semaforów

Poniżej przykład sieci petriego dla programu, który pozwala wykonać ostatnią funkcję, gdy 3 uswierbieżnione funkcje się wykonają.





## 8. Weryfikacja modelowa z zastosowaniem logiki temporalnej

Sama **logika** to nauka argumentowania, matematyczny opis rzeczywistości, jedna z głównych dziedzin matematyki, obok teorii zbiorów oraz arytmetyki. **Logika modalna** to klasyczna logika zdań określona matematycznie, przy pomocy formuł oraz relacji takich jak koniunkcja, negacja, alternatywa czy implikacja.

Natomiast **logika temporalna**, to rozszerzenie logiki tradycyjnej, modalnej o symbole określające upływ czasu. Pozwala ona wnioskować z uwzględnieniem czasu, przypisuje wartości prawda/fałsz do wyrażeń logiki modalnej. Umieszcza te wartości w strukturze czasowej.

### Czas liniowy, rozgałęziony i równoległy

**Liniowy:**  $(\forall t_0, t_1) (t_0 < t_1 \vee t_0 = t_1 \vee t_0 > t_1)$



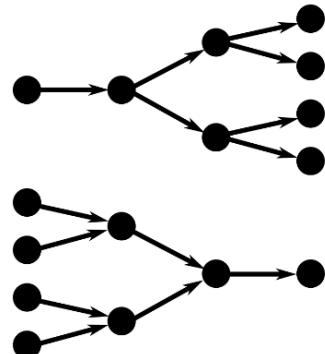
— istnieje tylko jeden wariant przepływu czasu.

### Rozgałęziony:

— istnieją różne warianty przepływu czasu, mające wspólną część ze sobą:

- lewostronna liniowość:

$(\forall t_0, t_1, t_2) (t_1 < t_0 \wedge t_2 < t_0 \Rightarrow t_1 < t_2 \vee t_1 = t_2 \vee t_1 > t_2)$



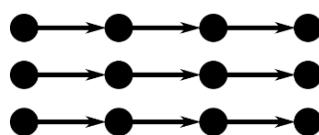
- prawostronna liniowość:

$(\forall t_0, t_1, t_2) (t_0 < t_1 \wedge t_0 < t_2 \Rightarrow t_1 < t_2 \vee t_1 = t_2 \vee t_1 > t_2)$

### Równoległy:

— istnieją różne warianty przepływu czasu, nie mające ze sobą żadnej wspólnej części;

— czas jest jednocześnie lewostronne i prawostronne liniowy.



Dla wyjaśnienia, **czas** (według wykładu z MIASI) to nieprzestrzenne kontinuum, w którym zdarzenia zachodzą w nieodwracalnej kolejności od przeszłości, przez teraźniejszość do przyszłości. Jest coś takiego jak czas:

- ciągły, gdzie dla 2 momentów istnieje pewien moment między innymi
- dyskretny - gdzie są takie 2 momenty, że nie istnieje żaden moment między nimi

ale też

- liniowy - istnieje tylko 1 wariant przepływu czasowego.
- rozgałęziony - istnieją różne warianty przepływu które mają części wspólne.
- równoległy - istnieją różne warianty przepływu które NIE mają części wspólnej, czas jest jednocześnie lewo i prawostronnie liniowy.

Żeby nie było za dobrze to czas też może być domknięty, więc mamy coś takiego jak czas skończony i nieskończony, z czego ten 1 może być zamknięty lewo, prawo lub obustronnie.

Czas jest też:

- punktowy — struktura czasowa składa się tylko z punktów.
- przedziałowy — struktura czasowa składa się tylko z przedziałów

#### **Przykładowe rodzaje logiki temporalnej (pierwsze 2 najważniejsze):**

- **LTL** - Linear Temporal Logic, jak nazwa wskazuje liniowa struktura czasu, jest lewostronnie skończona, czyli od pewnego momentu teraźniejszości. Nie istnieją ścieżki upływu czasu i współbieżnie wykonywane czynności (z racji na liniowość). Czas jest dyskretny, liniowy oraz punktowy. LTL można zastosować do opisu i analizy deterministycznych algorytmów, czy właściwości niedeterministycznych algorytmów, ale też tam, gdzie nie ma różnych wariantów przepływu czasu.
- **CTL** - Computation Tree Logic, tutaj mamy do czynienia z drzewem, więc rozgałęziony wariant czasowy (lewostronnie liniowy), podobnie jak w przypadku LTL lewostronnie skończony - nieograniczony od „terażniejszego” momentu do przyszłości. Czas jest dyskretny i punktowy. CTL można zastosować do opisu i analizy deterministycznych oraz niedeterministycznych algorytmów oraz właściwości algorytmów, i tam gdzie mogą być różne warianty przepływu czasu.
- **RTCTL** - Real-Time Computation Tree Logic, jest to wersja CTL, gdzie wartości czasu dane są ilościowo jako stałe. Pozwala na weryfikację systemów czasu rzeczywistego, gdzie dana operacja nie tylko musi być wykonana, ale też są na nią ograniczone ramy czasowe

Logikę temporalną **wykorzystuje się** głównie w modelowaniu i weryfikacji systemów:

- wspólnie - składają się z procesów wykonywanych jednocześnie, mających dostęp do wspólnych danych, potencjalnie współpracujących ze sobą.
- reaktywnych - działają na zasadzie akcja-reakcja, wobec pojawiających się okoliczności, przykładowo silnik w maszynie lub ruch lotniczy.

Najważniejsze zadania weryfikacji:

- osiągalność (pożądany stan zostanie w końcu osiągnięty)
- bezpieczeństwo (gwarancja, iż stan nieprawidłowy nigdy nie zostanie osiągnięty)

**Operatory temporalne** pozwalają przypisywać wyrażeniom wartość prawda / fałsz w strukturze czasowej. Podstawowe operatory temporalne:

- U ("dopóki") - pUq znaczy, że q jest prawdziwe w pewnym momencie; jeśli to przyszły moment, to w każdym wcześniejszym momencie p jest prawdziwe.

- X ("następnie") - Xp znaczy, że p jest prawdziwe w następnym momencie
- G ("zawsze") - Gp znaczy, że p jest prawdziwe w każdym momencie
- F ("kiedyś") - Fp znaczy, że p jest prawdziwe w pewnym momencie

W CTL oprócz operatorów temporalnych wykorzystuje się też operatory ścieżkowe. Kolejność priorytetów operatorów i spójników logicznych jest taka sama jak dla logiki LTL, z tym że każdy operator temporalny stanowi nierożłączną parę z jednym operatorem ścieżkowym. Operatory ścieżkowe pozwalają na użycie operatorów temporalnych dla:

- A — każdej możliwej ścieżki (przyszłego upływu czasu),  
E — pewnej możliwej ścieżki (przyszłego upływu czasu).

Tutaj jeszcze bym dorzucił z tych dalszych wykładów coś (w końcu jest weryfikacja modelowa w pytaniu):

## Modelowa weryfikacja systemu

### Modelowa weryfikacja systemu

#### Dane wejściowe:

- formalny model funkcji systemu dany jako automat;
- własności, które muszą być spełnione przez system, dane jako formuły logiki temporalnej.

#### Dane wyjściowe:

- odpowiedź, czy system spełnia te własności.

Czy każda możliwa sekwencja stanów systemu (programu) spełnia te własności?

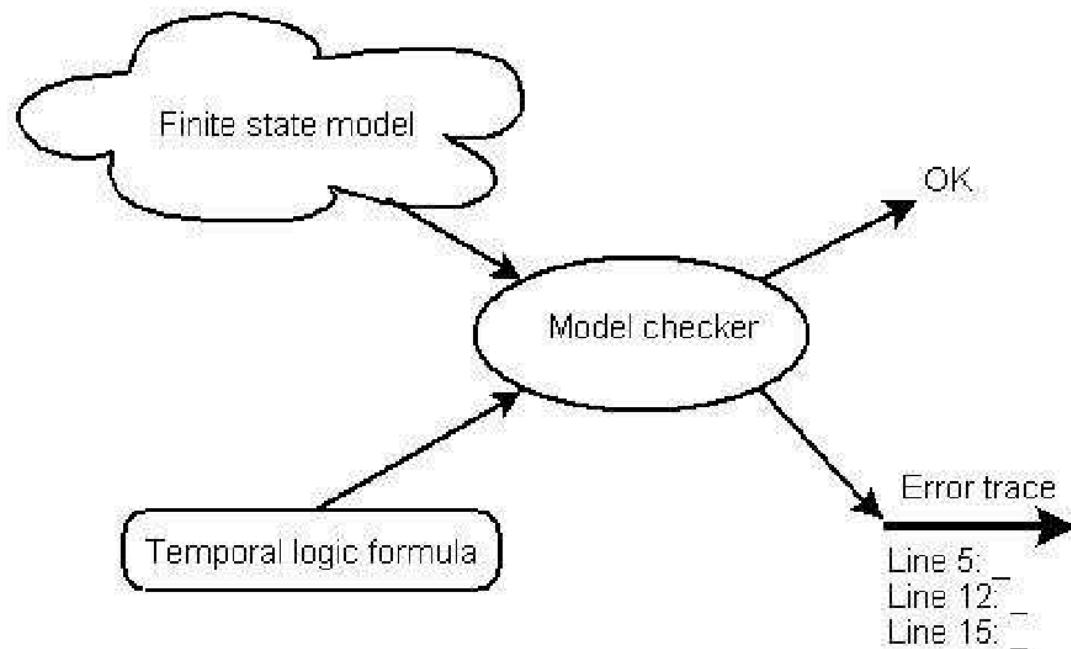
Czy któraś możliwa sekwencja stanów systemu (programu) spełnia te własności?

19

**Weryfikacja modelowa** - (ang. model checking) to technika automatycznego sprawdzania poprawności modelu systemu względem określonych własności formalnych wyrażonych w logice temporalnej. Proces weryfikacji modelowej składa się z kilku kroków:

1. Modelowanie systemu: Tworzenie formalnego modelu systemu, zazwyczaj jako system stanów lub automaty skończone. Jest to formalny opis systemu, skończenie stanowe automaty. Jest wykorzystana do tego Struktura Kripkego, czyli typ niedeterministycznego skończonego automatu.
2. Specyfikacja własności: Definiowanie właściwości, które system ma spełniać, przy użyciu formuł logiki temporalnej.

3. Sprawdzanie modeli. Automatyczne sprawdzenie, czy model spełnia określone własności za pomocą narzędzi do weryfikacji modelowej (np. SPIN, NuSMV, UPPAAL).



Cytat kogoś o wiele mądrzejszego apropos sprawdzania systemów informatycznych (w tym z użyciem właśnie logiki颞orarnej), dokładnie Djikstry, tego od algorytmów:

Sytuacja programisty jest podobna do sytuacji matematyka, który rozwija teorię i dowodzi twierdze. [...] Nigdy nie może on gwarantować, że dowód jest poprawny, co najwyżej może powiedzieć: „Nie znalazłem żadnych błędów”. [...] Jest to tak niezmiernie wiarygodne, że analogia może służyć jako wielkie źródło inspiracji. [...]

Nawet przy założeniu bezbłędnie działającej maszyny powinniśmy sobie zadawać pytania: „Gdy automatyczny komputer wytwarza wyniki, dlaczego ufamy im, jeśli tak jest?”, a następnie: „Jakimi środkami możemy wzmacnić nasze przeświadczenie, że wytworzone wyniki są rzeczywiście wynikami zamierzonymi?”

# 9. Modelowanie sieci komputerowych z wykorzystaniem przepływów wieloskładnikowych

[oparte na *opracowanko-Piotr-Z-2.pdf* i wykładach PSK]

## O optymalizacji i modelowaniu

Modelowanie jest jednym z trzech etapów projektowania sieci:

1. Sformułowanie problemu
2. **Modelowanie**
3. Optymalizacja

Aby umożliwić optymalizację, najpierw należy sformułować model matematyczny rozważanego problemu. Ale po co w ogóle optymalizować (wydaje mi się, że mogą o to zapytać, a jak nie to można tym trochę lać wodę).

Z wykładu PSK prof. Walkowiaka o celach optymalizacji (wykład 1, slajd 39):

- Oszczędności finansowe
- Poprawienie działania systemu
- Większa efektywności pracy
- Poprawienie niezawodności
- Poprawienie bezpieczeństwa
- Zmniejszenie zużycia zasobów

Do przeprowadzenia optymalizacji konieczne są ograniczenia, więc ponownie z wykładu tym razem jakie są przykładowe ograniczenia (wykład 1, slajd 45):

- Koszt (np. Pojemność)
- Opóźnienie
- Wydajność
- Niezawodność
- Bezpieczeństwo
- Zużycie zasobów

Teraz o samym modelowaniu (wykład 1, slajd 48):

- Model zawiera: zmienne, stałe, funkcję kryterialną i ograniczenia
- Wielkość modelu (tj. liczba zmiennych i ograniczeń) powinna być możliwie najmniejsza
- Ponieważ najskuteczniejsze metody optymalizacji są opracowywane dla problemów liniowych, często nieliniowa funkcja celu i ograniczenia są przybliżone przy użyciu funkcji liniowych

## O problemach optymalizacyjnych

Jak już wiemy czym jest modelowanie, to teraz trochę o rodzajach problemów optymalizacyjnych.

Oczywiste jest, że można je podzielić na problemy na te **z ograniczeniami i bez ograniczeń**.

Mniej oczywisty jest dalszy podział (wykład 2a, slajd 4):

- **Liniowy** – funkcja kryterialna oraz ograniczenia są funkcjami liniowymi, zmienne są ciągłe
- **Nieliniowy** – funkcja kryterialna nie jest funkcją liniową
- **Wypukły** – funkcja kryterialna oraz ograniczenia są funkcjami wypukłymi
- **Całkowitoliczbowy** – zmienne decyzyjne przyjmować wyłącznie wartości całkowite

Skoro już była wzmianka o problemach liniowych, to warto wspomnieć o programowaniu liniowym (wykład 2a, slajd 14):

**Programowanie liniowe LP** (ang. *Linear Programming*) to klasa problemów programowania matematycznego, w której wszystkie warunki ograniczające oraz funkcja celu mają postać liniową.

Warto też zapamiętać, że najpopularniejszą metodą rozwiązywania zadań programowania liniowego jest metoda **Simplex**.

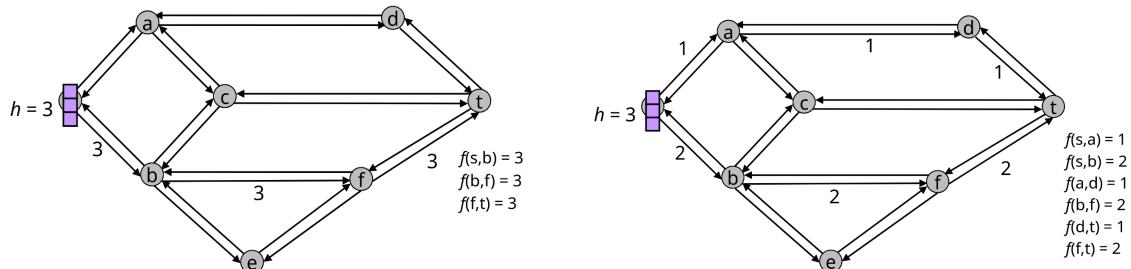
## O modelowaniu sieci i jej składnikach

Sieci komputerowe modelujemy jako grafy składające się z **wierzchołków** (węzły sieci, routery, switche, serwery i inne urządzenia końcowe), **krawędzi** (warstwa fizyczna, kable, światłowód, radio itp.) oraz **wag** przypisanych do krawędzi (maksymalna dopuszczalna przepustowość połączenia). Dodatkowe ograniczenia mogą zostać przypisane do wierzchołków (jaki całkowity ruch dany węzeł jest w stanie obsłużyć), ale do prostych modeli wystarczą wagi dla krawędzi.

Powyższe modeluje nam jednak jedynie **topologię sieci**. Dla pełniejszego zamodelowania działania sieci komputerowej niezbędne jest uwzględnienie w modelu również **przepływów** (pakietów, bitów) w sieci.

### Przepływy jednoskładnikowe

Do zrozumienia czym są przepływy wieloskładnikowe, korzystne może być zrozumienie przepływów jednoskładnikowych. Wykorzystajmy do tego przykłady ze slajdów (wykład 2b, slajdy 7 i 8):



Co prawda nie mam możliwości pokazania niesamowitych animacji profesora, ale możemy sobie wyobrazić kwadraciki poruszające się po krawędziach. Widoczne cyfry oznaczają, ile kwadracików porusza się po danej krawędzi. Widać, że składnik może poruszać się **bez rozgałęzień** (przykład po lewej stronie), jak i **z rozgałęzieniami** (przykład po prawej stronie). Obecność rozgałęzień **nie decyduje** o tym czy przepływ jest wieloskładnikowy! To jest dalej jeden składnik z jednym węzłem źródłowym (tam gdzie widoczne są kwadraciki) oraz jednym węzłem końcowym (węzeł  $t$ ).

W sumie w tym miejscu warte może być wymienienie cech przepływów z rozgałęzieniami i bez:

Z rozgałęzieniami:

- Żądanie można podzielić na mniejsze pakiety
- Można je (mniejsze pakiety) przesyłać z węzła docelowego do węzła źródłowego różnymi trasami (tutaj w opracowaniu *opracowanko-Piotr-Z-2.pdf* był błąd, bo te mniejsze pakiety nie są osobnymi składnikami).
- Wykorzystywane np. w protokole IP

Bez rozgałęzienia:

- Całe żądanie musi być przesyłane jedną trasą
- Wykorzystywane w protokołach zorientowanych połączeniowo, np. MPLS, FR, sieci optyczne
- Problem całkowitoliczbowy, problem NP. Jest to dodatkowe ograniczenie.

## Przepływy wieloskładnikowe

Jak to zostało przed momentem opisane, **składnik** to zbiór pakietów mający ten sam węzeł źródłowy i ten sam węzeł docelowy. Możemy z tego wywnioskować, że w przepływach wieloskładnikowych będziemy operować z sytuacją, gdzie mamy wiele takich zbiorów, i każdy z nich może mieć różne węzły źródłowe oraz docelowe. Natomiast nic nie stoi na przeszkodzie, żeby ktoś z nich współdzieliły któryś z nich.

Profesor Walkowiak definiuje **przepływ wieloskładnikowy** (ang. multicommodity flow) jako **średni przepływów pakietów** w sieci komputerowej w przyjętej jednostce czasu (wykład 2b, slajd 15).

Poniżej są przykłady zastosowania przepływów wieloskładnikowych (wykład 2b, slajd 43):

Typ sieci	Węzły	Łuki	Przepływ
Sieć informacyjna	Ludzie	Linie transmisyjne	wiadomości
Sieć komputerowa	Komputery, routery	Linie transmisyjne	dane
Sieć kolejowa	Dworce, skrzyżowania	Tory	Pociągi
Sieć dostawcza	Fabryki, magazyny	Drogi, tory, itp..	Ciążarówki, wagony

## Notacje węzeł-łuk i łuk-ścieżka

Tutaj nie jestem pewien, czy jest to specyficzne dla przepływów wieloskładnikowych, ale było i w wykładach i starym opracowaniu. Szczero wątpię, żeby mieli pytać o ten cały matematyczny bełkot, więc daję poprawione krótkie opisy ze starego opracowania.

### Węzeł-łuk

- Indeksowane są węzły sieci i łącza (łuki)
- Dla żądania musi być spełniony warunek: z węzła źródłowego wypływa całość żądania i wpływa ono całkowicie do węzła docelowego, dla każdego innego węzła suma ruchu wchodzącego i wychodzącego to 0

### Łuk-ścieżka

- Indeksowane są ścieżki kandydujące
- Potrzebna macierz zmiennych, która określa czy łuk należy do ścieżki
- Całe żądanie przesyłane jest jedną ścieżką (przepływy nierozgałęziające się) lub suma przepływających po ścieżkach danych jest równa rozmiarowi żądania (przepływy rozgałęziające się)

# 10. Modelowanie i optymalizacji przeżywalnych sieci komputerowych

**Przeżywalność** można zdefiniować jako **zdolność sieci do zapewniania i utrzymania akceptowalnego poziomu usług w obliczu różnych usterek i wyzwań w normalnym działaniu**. Przeżywalna sieć komputerowa powinna być zaprojektowana tak, aby utrzymać podstawowe operacje i usługi w obliczu różnych zakłóceń, takich jak awarie sprzętu, błędy oprogramowania, cyberataki, klęski żywiołowe i inne zagrożenia. W sieci nieprzeżywalnej wady i błędy prowadzą do awarii usługi czyli odchylenia działania usługi od oczekiwanej funkcjonowania systemu. Przy przepustowości sieci rzędu Gb/s lub Tb/s nawet krótka awaria spowoduje znaczną utratę danych. Przy dzisiejszej zależności od sieci komputerowych awaria może spowodować konsekwencje finansowe, polityczne, społeczne, zdrowotne.

Warto wspomnieć także o **sieciach samonaprawialnych**, które **posiadają zdolność do samodzielnego wykrycia uszkodzenia i przekonfigurowania własnych zasobów** w taki sposób, aby uszkodzenie w jak najmniejszym stopniu wpłynęło na jakość działania sieci.

## Jakie problemy mogą wystąpić w sieci?

- **Zagrożenia pasywne - niezależne od działań człowieka**, przykładowo huragany, trzęsienia ziemi, tsunami, powodzie i powszechnie przerwy w dostawie prądu.
- **Zagrożenia aktywne - spowodowane przez świadome działanie człowieka** przykładowo wirusy, terroryzm, vandalizm, inżynieria społeczna, atak na sieć zasilającą, backdoor w oprogramowaniu urządzeń sieciowych, DoS i DDoS.

## Jak modelować sieci przeżywalne?

Koncepcje reakcji na awarię w sieciach przeżywalnych:

- **Protekcja** - stosowana zazwyczaj w sposób rozproszony, bez centralnego sterowania. Zakłada, że zapasowe zasoby sieci przygotowane na wypadek awarii są przydzielane przed wystąpieniem awarii. **W momencie awarii następuje tylko przełączenie ruchu sieciowego na wcześniej przygotowane zapasowe zasoby**, metoda działa bardzo szybko.
- **Odtworzenie** - stosowane w sposób rozproszony lub skonsolidowany. Cechuje się brakiem wstępnie zarezerwowanych zasobów zapasowych. **Po wystąpieniu awarii dynamicznie próbuje odzyskać łączność w sieci wykorzystując wolne zasoby sieci**. Czas działania jest dość duży, ale brak wstępnej rezerwacji zapewnia elastyczność.  
Odtworzenia mogą być **globalne i lokalne**.

Podstawowym sposobem zapewniania przeżywalności jest **redundancja** - nadmiarowość:

- **Sprzętowa** - duplikowane są urządzenia sieciowe, serwery, zasilanie itd.
- **Połączeń** - dodawane są nadmiarowe łącza w sieci w celu zapewnienia alternatywnych tras przesyłania danych.

- **Programowa** - dodawane są mechanizmy wspomagania procesu powrotu sieci do normalnej pracy po awarii.

Bardziej zaawansowane metody zapewniania przeżywalności:

- **Automatyczna ochrona przez przełączanie** (ang. automatic protection switching - APS) - używa dedykowanych elementów chroniących podstawowe elementy sieci.
  - 1+1 - jeden działający system roboczy i całkowicie rezerwowy system zapasowy, w którym kopiowany jest sygnał podstawowej linii.
  - 1:1 - APS podobny do 1+1, ale transmitowany sygnał nie jest kopiowany jednocześnie do obu systemów, w przypadku awarii potrzebne jest przełączenie na system zapasowy.
  - 1:N - N systemów roboczych jest chronionych jednym systemem zapasowym.
- **Samonaprawialne pierścienie** (ang. self-healing rings).
- **Algorytmy z powielaniem** (ang. flooding algorithms).
- **Ochrona przez ścieżki wirtualne** (ang. virtual path protection switching).
- **P-cykle** (ang. p-cycles) - specjalne struktury w sieci wykorzystujące jednocześnie zalety topologii pierścienia i topologii siatki.
- **Ochrona przez ścieżki zapasowe**.

**Do problemu niezawodności można podejść szukając minimalnego zbioru węzłów, łuków, których usunięcie przerywa łączność (minimaksowe) lub szukać elementów sieci, które są najbardziej narażone na uszkodzenie (statystyczne).**

Awarie opisuje się przez stany awaryjne, które są pewnymi wektorami dostępności elementów w sieci.

## Optymalizacja sieci przeżywalnych

Do optymalizacji sieci przeżywalnych stosuje się te same metody, jak w klasycznych problemach, przykładowo: Simplex, branch and bound, Lagrangean relaxation, algorytmy ewolucyjne.

## Dodatek - opis ogólnikami

Czym cechuje się sieć przeżywalna?

- **Odporność na uszkodzenia** - kompensacja losowych, nieskorelowanych awarii komponentów.
- **Tolerancja zakłóceń** - nieprzewidywalnie długie opóźnienia, wyzwania związane z energią.
- **Tolerancja ruchu** - tolerowanie nieprzewidywalnego obciążenia bez znaczącego spadku w jakości usług.
- **Odzyskiwanie po awarii** - szybkie przywracanie operacji, kopie zapasowe danych, zautomatyzowane procesy odzyskiwania.

Jak modelować sieć przeżywalną?

- Budowanie sieci z elementów odpornych na awarie - parametr MTBF (ang. Mean Time Between Failures).
- Redundancja elementów sieci.
- Mechanizmy wykrywania anomalii, strategii zaradczych i odzysku.
- Projektowanie sieci na podstawie przewidywanego ruchu.
- Stosowanie architektury rozproszonej aby zapobiec punktowym awariom.
- Równoważenie obciążenia.
- Wirtualizacja w tym SDN.
- Mechanizmy korzystania z sieci innych firm w przypadku katastrofy (choć to bardziej telco).

## Dodatek - ogólnikowy opis standardów

Przykłady większych projektów/standardów (dość niszowe i specyficzne, ich znajomość nie będzie niezbędna, podrzucam bardziej jako przykłady, które można wykorzystać do lania wody):

- Advanced Networked Systems Architecture (ANSA) - strategia zarządzania niezawodnością składająca się z ośmiu etapów: ograniczanie awarii, wykrywanie awarii, diagnoza awarii, rekonfiguracja, odzyskiwanie, restart, naprawa i reintegracja. Rama ANSA definiuje regiony oczekiwania w dwuwymiarowej przestrzeni wartości względem czasu, aby opisać akceptowalną usługę.
- T1 od Working Group of Alliance for Telecommunications Industry Solutions (ATIS) - wielopoziomowa rama dla przeżywalności sieci z czterema warstwami: fizyczną, składającą się z infrastruktury o geograficznej różnorodności; systemową, składającą się z węzłów i łączy z przełączaniem ochronnym; logiczną, składającą się z pojemności na warstwie systemowej; i usługową, z dynamicznym trasowaniem i rekonfiguracją dla przeżywalności. Ten framework określa przerwy w usługach za pomocą trzech liter: U (niesprawność, odwrotny metryka niezawodności), D (czas trwania), E (zakres na różnych parametrach, w tym obszarze geograficznym, populacji i usługach) nasilenia tej trójki są mapowane do przestrzeni trójwymiarowej klasyfikowane jako drobne, poważne lub katastrofalne.
- CMU-CERT - czterostopniowa strategia składająca się z trzech "R": resistance (tradycyjne zabezpieczenia, różnorodność, redundancja, specjalizacja, weryfikacja zaufania i obserwowanie własności stochastycznych), recognition (analiza redundancji i testowanie, monitorowanie włamań i zachowania systemu, monitorowanie integralności) recovery (redundancja, zróżnicowane lokalizacje zasobów informacyjnych, planowanie awaryjne i zespoły reagowania) i jako czwarte adaptacja oraz ewolucja.

# 11. Algorytmy rozwiązujące złożone problemy optymalizacyjne - typy, przykłady, wady i zalety.

Większość na podstawie [1]

## Typy:

- **Algorytmy dokładne** - znajdują najlepsze rozwiązanie problemu kosztem potencjalnie nieakceptownego czasu pracy - przykłady: **przeszukiwanie pełne, Branch and bound, programowanie matematyczne, programowanie dynamiczne**
- **Algorytmy Heurystyczne** - algorytm niedający (w ogólnym przypadku) gwarancji znalezienia rozwiązania optymalnego, umożliwiający jednak znalezienie rozwiązania dość dobrego w rozsądny czasie. [2]
  - **Metaheurystyki** - heurystyka wyższego poziomu, algorytmy tego typu nie rozwiązują bezpośrednio żadnego problemu, a jedynie podają sposób na utworzenie odpowiedniego algorytmu heurystycznego - przykłady: **przeszukiwanie lokalne, algorytm genetyczny, symulowane wyżarzanie, Tabu Search.** [3]
  - **Heurystyki konstruujące** - typ algorytmów optymalizacji, które wykorzystują informacje zebrane dotychczas przez algorytm lub dane dostarczone z góry, aby wspomóc podjęcie decyzji, który potencjalne rozwiązanie powinno być testowane oraz jak stworzyć kolejne potencjalne rozwiązanie. Heurystyka jest zależna od rozwiązywanego problemu. [1]
  - **Heurystyki wyspecjalizowane dla danego problemu** [1]
- **Algorytmy Aproksymacyjne** - algorytmy służące do znajdowania przybliżonych rozwiązań problemów optymalizacyjnych. Istotą algorytmu aproksymacyjnego, tym co odróżnia go od algorytmu heurystycznego, jest związana z każdym takim algorymem informacja o koszcie zwracanego rozwiązania w stosunku do rozwiązania optymalnego. Mianowicie koszt rozwiązania zwróconego przez algorytm aproksymacyjny jest nie większy albo nie mniejszy od rozwiązania optymalnego pomnożonego przez pewną stałą. [4]

## Przykłady:

### Dokładne:

[5] **Przeszukiwanie pełne (exhaustive search, brute force)** - sprawdzenie wszystkich odpowiedzi w celu znalezienia optymalnej.

Zalety: prosta implementacja, gwarancja znalezienia optymalnej odpowiedzi

Wady: wymaga sprawdzenia wszystkich odpowiedzi, potencjalnie najwolniejsza możliwa metoda

### [1, 6] **Branch and bound (Metoda podziału i ograniczeń)**

Głównym założeniem metody podziału i ograniczeń (branch-and-bound) jest uporządkowane przeszukiwanie drzewa reprezentującego przestrzeń rozwiązań problemu oraz możliwość

odcięcia gałęzi, które nie zawierają „dobrych” rozwiązań. Dzięki temu nie przeglądamy całego drzewa i możemy dzięki temu dzielić je i przeglądać tylko obiecujące obszary.

Ta metoda składa się z dwóch podstawowych procedur:

- **rozgałęzianie (branching )** — dzielenie zbioru rozwiązań reprezentowanego przez węzeł na rozłączne podzbiory, reprezentowane przez następców tego węzła
- **ograniczanie (bounding)** — pomijanie w przeszukiwaniu tych gałęzi drzewa, o których wiadomo, że nie zawierają optymalnego rozwiązania w swoich liściach

Ogólny schemat postępowania wygląda następująco:

1. zbiór wszystkich dopuszczalnych rozwiązań jest dzielony na mniejsze podzbiory
2. dla każdego otrzymanego podzbioru obliczamy ograniczenie wartości funkcji celu
3. z dalszych podziałów eliminujemy podzbiory o ograniczeniu przekraczającym wartość funkcji celu

Metody przeszukiwania:

1. **przeszukiwanie wszerz (breadth-first search)** — polega na odwiedzeniu wszystkich osiągalnych wierzchołków z wierzchołka początkowego
2. **przeszukiwanie w głąb (depth-first search)** — polega na badaniu wszystkich krawędzi wychodzących z podanego wierzchołka
3. **najpierw najlepszy (best-first search)** — wybierane są te drogi, które wydają się prowadzić do najlepszego rozwiązania

Zalety: Potencjalnie szybsza metoda znalezienia optymalnego rozwiązania

Wady: Kompletnie zależny od prawidłowej estymacji górnych i dolnych granic gałęzi przestrzeni. Bez nich algorytm jest efektywnie przeszukiwaniem pełnym.

### **Heurystyczne:**

[1] **Przeszukiwanie lokalne** - Algorytm przeszukiwania lokalnego porusza się od rozwiązania do rozwiązania sąsiedniego w przestrzeni rozwiązań. Koniec przeszukiwania ustalany jest przez kryterium końca, np. znalezienie optimum, czas itp.

Cechy sąsiedztwa:

- ograniczenie na rozmiar: dla każdego  $x$ ,  $N(x)$  zawiera co najmniej jedno rozwiązanie różnie od  $x$ ; zbiór sąsiednich rozwiązań nie może być równy liczebnością przestrzeni wszystkich rozwiązań;
- podobieństwo sąsiadów: kolejne rozwiązanie nie powinno różnić się zbyt wiele tak, aby nie wymagało konstruowania zupełnie nowego rozwiązania;
- równouprawnienie: niezależnie od wyboru rozwiązania początkowego, powinno być możliwe osiągnięcie każdego rozwiązania należącego do pełnej przestrzeni rozwiązań

Strategie przeszukiwania:

- **Algorytm zachłanny (greedy)** - Algorytm wybiera pierwsze przejrzane rozwiązanie sąsiednie, które jest lepsze niż aktualne.
- **Algorytm stromy (steepest)** - Algorytm wybiera najlepsze rozwiązanie spośród swoich sąsiadów, które jest lepsze niż aktualne

Zalety:

- proste w implementacji
- stosunkowo szybkie w działaniu
- stosowalne dla wielu problemów optymalizacji

Wady:

- wynik ich działania jest zależny od rozwiązania startowego
- bardzo duże ryzyko zatrzymania w optimum lokalnym
- znalezione rozwiązanie jest zwykle rozwiązaniem przybliżonym danego problemu
- ustalone sąsiedztwo

**JESZCZE DODAM PRZYKŁADY PÓŹNIEJ**

- [1] [http://aq96820.home.amu.edu.pl/wp-content/uploads/2019/05/optymalizacja\\_si.pdf](http://aq96820.home.amu.edu.pl/wp-content/uploads/2019/05/optymalizacja_si.pdf)
- [2] [http://algorytmy.ency.pl/artykul/algorytmy\\_heurystyczne](http://algorytmy.ency.pl/artykul/algorytmy_heurystyczne)
- [3] <https://pl.wikipedia.org/wiki/Metaheurystyka>
- [4] [https://pl.wikipedia.org/wiki/Algorytm\\_aproksymacyjny](https://pl.wikipedia.org/wiki/Algorytm_aproksymacyjny)
- [5] <https://jakubnowosad.com/ahod/05-exhaustive-search.html#5>
- [6] [https://en.wikipedia.org/wiki/Branch\\_and\\_bound](https://en.wikipedia.org/wiki/Branch_and_bound)

## 12. Diagnozowanie i monitoring sieci z wykorzystaniem systemu Linux

Linux jest systemem opensource, co oznacza, że można go dostosować do specyficznych potrzeb sieciowych. Istnieje wiele narzędzi dedykowanych do monitorowania i diagnostyki sieci, które są stale rozwijane:

- **ping**
  - Wysyła żądania ICMP *echo request* do docelowego hosta i czeka na odpowiedzi *echo reply*. W ten sposób mierzy czas, jaki zajmuje podróż pakietów tam i z powrotem.
- **traceroute**
  - Identyfikuje trasę jaką pokonuje pakiet z jednego hosta do drugiego. Pomaga w diagnozowaniu problemów z routingu i identyfikacji wąskich gardeł. Stopniowo wysyła pakiety z coraz to większą wartością TTL (Time to Live), która zmniejszana jest o jeden przez każdy router po drodze. Gdy osiągnie 0 router odrzuca pakiet i wysyła ICMP *time exceeded* z powrotem do nadawcy.
- **iperf / netperf**
  - wykonywanie pomiarów wydajności sieciowej poprzez generowanie ruchu testowego w trybie klient-serwer,
  - pomiar przepustowości, badanie opóźnień, testowanie stabilności połączenia, identyfikacja przeciążeń, wąskich gardeł,
  - testowanie wydajności aplikacji sieciowych np. serwery plików, serwery baz danych, serwery WWW, symulacje specyficznych obciążen sieciowych, testowanie QoS, benchmarking
- **nslookup / resolvectl / dig**
  - diagnostyka DNS, sprawdzenie przypisanych adresów IP i innych rekordów dla konkretnych domen,
  - zarządzanie ustawieniami DNS w systemd
  - wykonywanie zaawansowanych zapytań DNS
- **ifconfig**
  - tradycyjne narzędzie w systemach Unix do konfiguracji i zarządzania interfejsami sieciowymi. Pozwala na wyświetlanie informacji o interfejsach w tym adresacje IP, MAC, rodzaj interfejsu, MTU, liczba odebranych i wysłanych pakietów, długość kolejki pakietów itp.
- **ip**
  - Oferuje bardziej zaawansowane funkcje niż ifconfig w tym dodatkowo:
    - konfiguracje trasowania (routing),
    - zarządzanie regułami routingu,
    - konfiguracja tuneli,
    - zarządzanie VLAN,
    - konfiguracja multicast,
    - zasady QoS,
    - zarządzanie mostami
    - monitorowanie zdarzeń sieciowych w czasie rzeczywistym (ip monitor)

- **netstat**
  - pokazuje listę wszystkich aktywnych połączeń TCP, UDP, ICMP oraz UNIX sockets. Dla każdego połączenia wyświetlane adresy, numery portów oraz stan połączenia,
  - dostarcza statystyki protokołów sieciowych i aktualną tablicę routingu systemu,
  - pokazuje wszystkie porty, na których aplikacje nasłuchują na połączenia przychodzące
  - Można zidentyfikować, które połączenia są aktywne, jakie porty są używane i czy są jakieś nieoczekiwane połączenia. Łatwe narzędzie do szybkiego audytu sieci i wykrywania potencjalnie nieautoryzowanych połączeń lub usług.
- **ss**
  - pokazuje informacje o wykorzystaniu gniazd sieciowych, rozszerza funkcjonalność netstat,
  - informacje o rodzajach gniazd TCP/UDP/DCCP/Raw/Unix
  - informacje o procesach, użytkownikach, pamięci, statystyce, ruchu, parametrach protokołów TCP/UDP,
  - umożliwia zaawansowane filtrowanie wyników na podstawie kryteriów.
- **tcpdump**
  - monitoruje wszystkie pakiety sieciowe przechodzące przez wybrany interfejs lub całą sieć,
  - przechwytuje pakiety różnych protokołów i umożliwia definiowanie zaawansowanych filtrów, pozwalając na selektywne przechwytywanie,
  - może działać w trybie przechwytyującym wszystkie pakiety, nie tylko te trafiające do systemu lub z niego
- **wireshark**
  - jedno z najbardziej zaawansowanych narzędzi do analizy sieci, dostępne na różnych platformach, w tym Linux,
  - działa jak tcpdump i obejmuje szeroki zakres protokołów, w tym również HTTP, DNS, SSL, a więc różne warstwy modelu OSI,
  - oferuje intuicyjny interfejs graficzny, wyświetlając pakiety i ich zawartość w czytelnej formie tabelarycznej.
- **nmap**
  - skanowanie sieci, analiza urządzeń i usług, szczegółowe badanie infrastruktury sieciowej, identyfikacja urządzeń,
  - skanowanie otwartych portów na hostach lub całej sieci,
  - rozpoznanie działających usług np. wersje oprogramowania, typy serwerów WWW, bazy danych, FTP, SSH itp.
  - określenie rodzaju i wersji systemu operacyjnego,
  - używane przy audytach, testach penetracyjnych, diagnostyce sieciowej.
- **arp**
  - wyświetlenie oraz manipulowanie tablicą Address Resolution Protocol, tzn mapowanie między adresami IP, a adresami MAC,
  - brak wpisów w tablicy lub błędy skutkują problemem z komunikacją w sieci
- **conntrack**
  - mechanizm śledzenia stanu połączeń w jądrze Linux, monitoruje i zarządza stanem połączeń sieciowych, istotne dla zarządzania ruchem i NAT,

- utrzymywana tabela stanów połączeń wraz z metadanymi, działa w ścisłej integracji z iptables, co pozwala na dynamiczne zarządzanie regułami filtrowania pakietów w trybie połączeniowym
- **iptables / ebtables**
  - zarządzanie regułami firewall na poziomie warstw L3/L2, kontrola ruchu i zarządzanie, filtrowanie oraz przekierowywanie ruchu,
  - wgląd w blokowane, akceptowane i przekierowywane pakiety w zależności od reguł,
- **/proc/net**
  - wirtualny system plików udostępniany przez jądro, który dostarcza informacji o bieżącym stanie sieci. Zawiera surowe, lecz szczegółowe statystyki często interpretowane przez opisane wcześniej narzędzia.
- **ethtool**
  - zarządzanie i diagnostyka interfejsów Ethernet, używane głównie do uzyskiwania informacji oraz ustawiania parametrów fizycznych,
  - wyświetlanie informacji o interfejsie (MAC, MTU, stan, prędkość i tryb pracy np 1000Mbit full duplex, obsługiwane tryby transmisji, statystyki błędów),
  - regulacja offload, prędkości, kontroli przepływu, funkcji energooszczędnich
- **iw**
  - umożliwia skanowanie sieci Wi-Fi w zasięgu danego interfejsu, można wyświetlić dostępne sieci, ich SSID, kanały, siłę sygnału, jakość sygnału, szyfrowanie oraz inne parametry,
  - pozwala na uzyskanie szczegółowych informacji o konkretnym radiu
    - obsługiwane tryby pracy
    - obsługiwane częstotliwości i kanały (2GHz, 5GHz, 6GHz)
    - ustawienia mocy transmisji,
    - obsługiwane standardy (802.11)
  - przełączanie interfejsu Wi-Fi w tryb monitorowania, który jest niezbędny do przechwytywania pakietów oraz analizowania ruchu.
- **wavemon**
  - interaktywne środowisko CMD, które umożliwia szybkie wyświetlanie kluczowych parametrów sieci Wi-Fi,
  - przedstawia poziom sygnału w czasie rzeczywistym, w postaci wykresu lub wskaźnika,
  - monitoruje jakość połączenia, w tym fluktuacje sygnału, opóźnienia transmisji itp

## 13. Usługi katalogowe systemu Windows 200x Serwer

Usługi katalogowe systemu Windows Server, znane jako Active Directory (AD), to **hierarchiczna struktura** używana do **przechowywania informacji o zasobach sieciowych**, takich jak użytkownicy, komputery, drukarki i inne urządzenia. Active Directory jest centralnym punktem zarządzania zasobami sieciowymi w systemach Windows 2000 Server i późniejszych.

W gruncie rzeczy jest to obszerna, **hierarchiczna baza danych** o obiektach w sieci wraz z mechanizmami dostępu do tej bazy. Technicznie jest to implementacja protokołu sieciowego warstwy aplikacji **LDAP (ang. Lightweight Directory Access Protocol)**. Owa implementacja najlepiej nadaje się do zarządzania komputerami z systemem **Windows Pro** (dla firm) lub Linux/Unix po odpowiedniej konfiguracji. Umożliwia grupowanie komputerów w **domeny Windows** i scentralizowaną administrację dzięki przechowywanym danym sieciowym (użytkownicy, drukarki, serwery czy komputery). Każda grupa składa się z obiektów, które przy pomocy atrybutów opisują konkretny byt. Można dzięki temu uzyskać **spójność nazw, opisów, lokalizacji, praw dostępu, zarządzania czy zabezpieczeń**.

Jest to potężne i skalowalne narzędzie, które powinno ułatwiać pracę administratora w środowiskach o dużej skali. Daje możliwość delegowania zadań administracyjnych do poszczególnych użytkowników lub grup, co pozwala na lepszy podział obowiązków. Na całość usług związanych z Active Directory składa się aż 5 elementów:

- AD Domain Services
- AD Certificate Services
- AD Lightweight Directory Services
- AD Rights Management Services
- AD Federation Services

Usługi katalogowe poza uproszczeniem i scentralizowaniem konfiguracji, pomagają także zachować mechanizmy bezpieczeństwa jak uwierzytelnianie i autoryzacja, które chronią dane i zasoby przed nieuprawnionym dostępem. Istnieje także możliwość replikacji danych katalogowych między kontrolerami domeny, co zwiększa niezawodność i dostępność danych. Active Directory wspiera także definiowanie szerokich polityk bezpieczeństwa w sieci, które pomagają chronić dane i zasoby przedsiębiorstwa.

Żeby lepiej zrozumieć dalsze rozważania co dają nam usługi katalogowe, należy wyjaśnić jeszcze kilka pojęć:

### Magazyn danych

Magazyn danych (Data Store) w Active Directory to miejsce, gdzie przechowywane są **wszystkie informacje o zasobach sieciowych**. Dane te są organizowane w strukturę hierarchiczną i przechowywane w bazie danych zlokalizowanej na kontrolerach domeny.

Active Directory używa bazy danych Jet (extensible storage engine - ESE) do przechowywania danych, które są fizycznie zapisane w pliku 'NTDS.dit'.

## Kontroler domeny

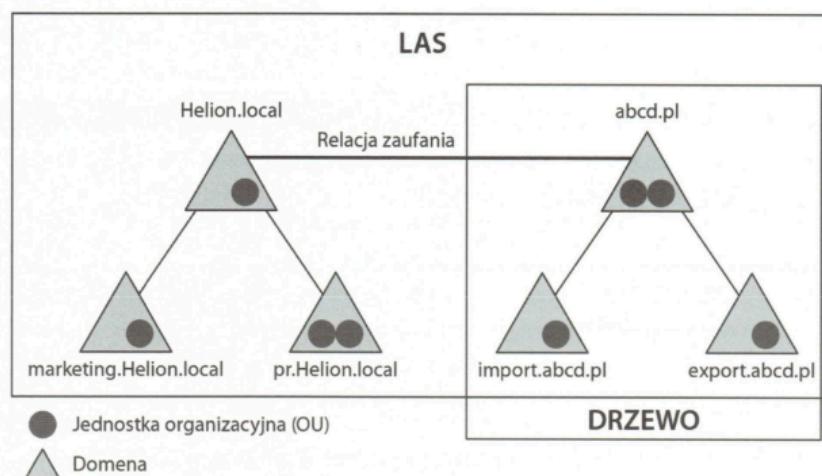
Kontroler domeny (Domain Controller) to **serwer**, który odpowiada za autoryzację użytkowników, przechowywanie danych katalogowych oraz egzekwowanie zasad bezpieczeństwa w domenie. Kontrolery domeny zarządzają replikacją danych Active Directory i zapewniają dostępność informacji na temat zasobów sieciowych. W przypadku awarii jednego z kontrolerów domeny, inne kontrolery przejmują jego rolę, co zwiększa niezawodność systemu. Wyróżnić możemy kontrolery typu **Global Catalog** (katalog globalny), a także kontrolery tylko do odczytu - **Read-Only Domain Controller** oraz odczytu i zapisu – **Writeable Domain Controller**.

## Domena

Domena (Domain) to podstawowa jednostka strukturalna w Active Directory, która **grupuje** użytkowników, komputery i inne zasoby. **Domena posiada unikalną nazwę DNS** (Domain Name System) i jest **zarządzana przez jeden lub więcej kontrolerów domeny**.

## Las

Las (Forest) to **najwyższy poziom hierarchii Active Directory**, który może zawierać **jedną lub więcej domen**. Las jest **kolekcją domen**, które dzielą wspólną schemat katalogową, konfigurację oraz relacje zaufania. Domeny w lesie mogą współdzielić zasoby i zapewniają spójne zarządzanie politykami i uprawnieniami. Las jest również autonomiczną jednostką administracyjną, co oznacza, że **wszystkie domeny w lesie są zarządzane wspólnie**. Pierwsza domena, która zostanie utworzona w lesie, będzie tak zwaną domeną główną lasu, a cały las przyjmie nazwę taką jak domena główna. Jeśli przykładowo tworzymy nową domenę w nowym lesie i nazwiemy ją test.local to cały las przyjmie taką nazwę.

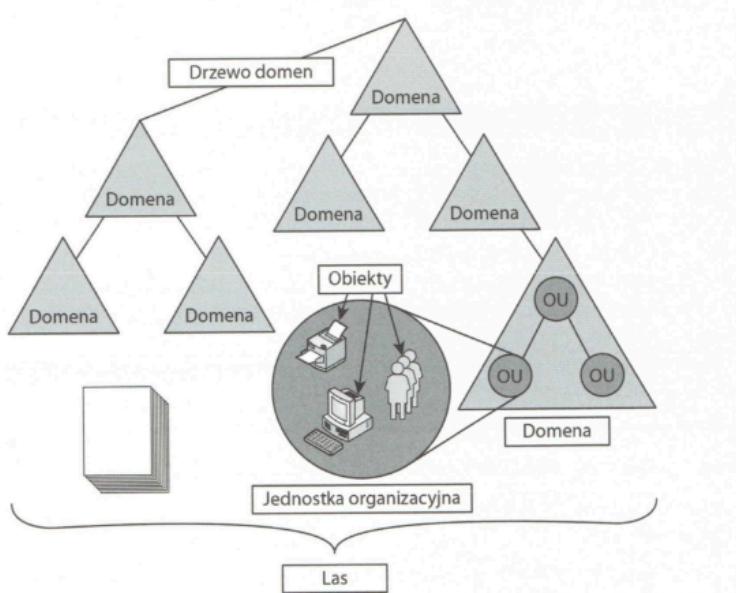


Rysunek 2.2. Struktura lasu domen

## Drzewo

Drzewo (Tree) to **hierarchiczna struktura domen w ramach lasu**, które mają **wspólną przestrzeń nazw DNS**. Drzewo składa się z domen nadzędnych i podrzędnych, które są ze sobą połączone hierarchicznie. Domeny w drzewie dzielą się przestrzenią nazw i zaufaniami, co umożliwia łatwiejsze zarządzanie zasobami.

Drzewo można także określić jako: jedna domena, albo kilka domen **pracujących pod tą samą przestrzenią nazw DNS**.



Rysunek 2.1. Struktura domeny

## Jednostka organizacyjna

Jednostka organizacyjna (Organizational Unit, OU) to **kontener w ramach domeny**, który służy do organizowania i zarządzania obiekty, takimi jak użytkownicy, grupy, komputery i inne jednostki organizacyjne. OU pozwala administratorom na delegowanie uprawnień administracyjnych i stosowanie polityk grupowych w bardziej granularny sposób. Dzięki OU można logicznie **podzielić zasoby sieciowe na podstawie struktury organizacyjnej firmy**, co ułatwia zarządzanie i administrację.

## Polityki grupowe

Polityki grupowe (Group Policy) to **mechanizmy zarządzania**, które umożliwiają administratorom **definiowanie i egzekwowanie konfiguracji** dla użytkowników i komputerów w domenie. Polityki grupowe mogą kontrolować szeroki zakres ustawień, od konfiguracji systemu operacyjnego, poprzez instalację oprogramowania, aż po ustawienia zabezpieczeń. Polityki grupowe są definiowane na poziomie domeny, jednostki organizacyjnej lub lokalnie na komputerze.

## Relacje zaufania

Relacje zaufania (Trust Relationships) umożliwiają użytkownikom z jednej domeny **uzyskiwanie dostępu do zasobów w innej domenie**. Relacje zaufania mogą być **jednokierunkowe lub dwukierunkowe** i są tworzone pomiędzy domenami w tym samym lesie lub pomiędzy różnymi lasami. Dzięki relacjom zaufania, zasoby mogą być udostępniane w sposób kontrolowany i bezpieczny.

## Kilka użytecznych przypadków administracji

To co chyba najistotniejsze z wykorzystania usług katalogowych:

- Na wszystkich komputerach włączonych w domenę można **logować się za pomocą tych samych kont**, które są przechowywane na serwerze pełniącym rolę kontrolera.
- Przy pomocy konta domenowego można także zapewniać **dostęp do zasobów udostępnionych w domenie**, czyli do katalogów sieciowych, drukarek itp.
- **Uproszczenie administracji** dzięki przypisywaniu uprawnień do **grup** zamiast indywidualnych użytkowników.
- **Aktualizowanie oprogramowania i polityka bezpieczeństwa**
- **Centralna konfiguracja ustawień systemu Windows / Konfiguracja przydziałów dyskowych.**
- Dodawanie **folderów udostępnionych** dla obiektów usługi domenowej
- **Odwzorowanie struktury organizacji** poprzez hierarchię jednostek organizacyjnych, które stanowią drzewo i którym można przypisywać konkretne **obiekty z uprawnieniami GPO (Group Policy Object)**.
- **Delegowanie kontroli administracyjnej** innym użytkownikom.
- **Dziedziczenie uprawnień, nadawanie uprawnień specjalnych** (których jest mnóstwo w zależności od przypadku użycia).
- Możliwość stworzenia **profilu mobilnego**, który umożliwi dostęp do dokumentów, pulpitu czy ustawień konkretnego użytkownika.
- Przypisywanie licznych obiektów GPO do grup zabezpieczeń.

## Sposoby logowania do komputera w domenie

Możemy rozróżnić od siebie użytkowników lokalnych i użytkowników domenowych. Pierwszy typ użytkowników są to konta stworzone lokalnie i dają możliwość użytkowania zasobów maszyny. Drugi typ natomiast jest stworzony i przechowywany na kontrolerze domeny w Active Directory. Umożliwia on logowanie do domeny i użytkowanie zasobów komputerów w domenie.

# 14. Metody dostępu i protokoły komunikacyjne w systemach pamięci masowych

**Pamięć masowa** — pamięć trwała, umożliwiająca przechowywanie dużych ilości danych w sposób trwały i możliwy do odczytu maszynowego przez długi czas. W odróżnieniu od pamięci operacyjnej nie pozwala na adresowanie pojedynczych bajtów, a jej czas dostępu jest wielokrotnie dłuższy.

W pamięciach masowych możemy wymienić 4 główne typy nośników pamięci:

- **Magnetyczne dyski twarde** — najpopularniejsze urządzenie pamięci masowej o dużej pojemności, przechowujące dane na okrągłym dysku z powłoką ferromagnetyczną. Umożliwiają losowy dostęp read/write.
- **Dyski SSD (flash)** - przechowują dane w pamięci opartej na półprzewodnikach, posiadają niskie zapotrzebowanie mocy oraz wysoką przepustowość.
- **Napędy taśm magnetycznych** - niedrogie rozwiązanie do długoterminowego przetrzymywania danych, przechowuje dane na cienkiej folii z tworzywa sztucznego z powłoką magnetyczną.
- **Napędy dysków optycznych** — nośniki CD, DVD, BD. Również niedrogie rozwiązanie do długoterminowego przetrzymywania danych. Dane przetrzymywane są na dysku z poliwęglanu z powłoką magnetyczną.

Pamięć masowa może być podłączona bezpośrednio lub przez SAN. Łączność komputerów z systemami pamięci masowych jest możliwa dzięki:

- fizycznym komponentom:
  - HBA (ang. Host Bus Adapter), porty i kable
- protokołom interfejsów definiujących formaty komunikacji między urządzeniami:
  - IDE/ATA, SCSI i FC

Systemy pamięci masowych operują się na 2 głównych architekturach:

- **serwerowo-centryczna** — architektura pamięci masowych, gdzie każdy serwer posiadał własne pamięci masowe. Przykładowo każdy z działów jak, dział sprzedaży, R&D czy księgowości.
- **informacyjno-centryczna** — architektura pamięci masowych, gdzie posiadamy pojedyncze źródło danych dostępne z pomocą przykładowo SAN (ang. Storage Area Network). Gdzie wiele serwerów może być podłączonych do sieci SAN, która umożliwia dostęp do skonsolidowanej pamięci masowej.

**Protokoły komunikacyjne** wykorzystywane w systemach pamięci masowej:

- **FTP**
  - Dostęp do plików przez sieć TCP/IP.
  - Wykorzystuje połączenie przez protokół TCP.
  - Wymagany klient - np. FileZilla.
  - 21 - port klienta.
  - 20 - port serwera.
- **IDE/ATA**
  - Popularny, starszy format interfejsu służący do podłączania dysków twardych i napędów optycznych.

- W większości zastąpiony przez SATA.
  - Wersja Ultra DMA/133 ATA obsługuje przepustowość 133 MB/s.
- **Serial ATA (SATA)**
  - Popularny interfejs do obsługi pamięci masowych.
  - Jest szeregową wersją specyfikacji IDE/ATA używaną zwykle do połączeń wewnętrznych.
  - Stosowany głównie w zastosowaniach konsumenckich.
  - Zapewnia szybkość przesyłania danych do 16 GB/s (standard 3.2).
- **SCSI**
  - Standard magistrali danych pozwalający na łączenie wielu urządzeń w jedną magistralę.
  - Pozwala na łączeniu kilku serwerów, dysków i innych urządzeń pamięci masowej, dzięki czemu możliwe jest podłączenie jednego dysku do kilku serwerów i przesyłanie danych pomiędzy dyskami bez udziału komputera.
  - Obsługuje do 16 urządzeń na 1 magistrali.
  - Wersja Ultra-640 zapewnia szybkość przesyłania do 640 MB/s.
- **SAS**
  - Następca równoległego SCSI, szeregowy protokół punkt-punkt.
  - Używany do podłączania napędów, głównie dysków do serwerów.
  - Posiada częściową kompatybilność ze standardem SATA, pozwalając na działanie dysków SATA z kontrolerami SAS (jednak w drugą stronę takiego wsparcia nie ma).
  - Zapewnia szybkość przesyłania do 12 GB/s (SAS 3.0).
- **FC (ang. Fibre Channel)**
  - Popularny, szeroko stosowany protokół szybkiej transmisji danych w sieciach SAN.
  - Najczęściej pracuje na światłowodzie, jednak może również działać na kablach miedzianych, zapewnia szeregową transmisję danych.
- **IP:**
  - Wykorzystuję istniejącą sieć opartą na protokole IP.
  - Przykłady protokołów to iSCSI oraz FCIP.

#### **Metody dostępu** do pamięci masowej:

- **dostęp blokowy** - (ang. Block Storage) zapewnia blokowy dostęp do pamięci masowej. Przyjmuje dowolne dane, takie jak plik czy wpis w bazie danych, i dzieli je na bloki o różnych rozmiarach. System pamięci masowej dostępu blokowego przechowuje następnie blok danych w podstawowej pamięci fizycznej w sposób zoptymalizowany pod kątem szybkiego dostępu i wyszukiwania. Dostęp ten jest preferowany, gdy wymagane jest szybkie, wydajne i niezawodne przesyłanie danych. Może być budowany na podstawie architekturę skalowaną wertykalnie lub horyzontalnie. Można o nim myśleć jak o bardziej bezpośrednim dostępie w postaci strumienia do danych. Deweloperzy często wdrażają pamięć blokową jako sieć pamięci masowej (SAN). Sieć **SAN**- sieć pamięci masowej, obszar sieci zapewniający systemom komputerowym dostęp do zasobów pamięci masowej. Zwykle odległa pamięć masowa stanowi centralną przestrzeń składowania danych współdzieloną poprzez SAN przez wiele hostów w środowisku rozproszonym. Cechuje się:
  - Z reguły jest osobną siecią, niedostępna z głównej sieci LAN

- Dostęp do danych następuje poprzez serwery włączone w SAN
- Tego typu sieci oferują dostęp tylko na poziomie blokowym, jednak istnieją systemy plików budowane w oparciu o tą sieć, które oferują dostęp na poziomie plików
- **dostęp plików** — (ang. File Storage) posiada dodatkową warstwę składającą się z systemu plików. Przechowuje dane w hierarchicznej strukturze plików i folderów. W środowiskach sieciowych pamięć masowa oparta na plikach często korzysta z technologii pamięci masowej podłączonej do sieci (NAS). NAS umożliwia użytkownikom dostęp do danych w pamięci sieciowej w podobny sposób, jak do lokalnego dysku twardego. Przechowywanie plików jest przyjazne dla użytkownika i pozwala użytkownikom zarządzać kontrolą udostępniania plików.
- **dostęp obiektowy** — (ang Object Storage). Przechowuje dane w postaci obiektów w płaskiej przestrzeni adresowej na podstawie ich zawartości i atrybutów, a nie nazwy i lokalizacji. Obiekt zawiera dane użytkownika, powiązane metadane i atrybuty zdefiniowane przez użytkownika. Obiekty są jednoznacznie identyfikowane przez identyfikator obiektu - object ID. Technologia przechowująca dane w nieustrukturyzowanym formacie zwany obiektami i zarządzająca nimi. Najlepiej sprawdza się w przypadku dużych ilości nieustrukturyzowanych danych, zwłaszcza gdy trwałość, nieograniczone miejsce na dane, skalowalność i złożone zarządzanie metadanymi są istotnymi czynnikami wpływającymi na ogólną wydajność.

## Pamięć ujednoliciona (Unified Storage)

### Unified Storage

Pojedyncza, zintegrowane (konwergentna) infrastruktura pamięci masowej, która konsoliduje dostęp blokowy (iSCSI, FC, FCoE), plikowy (CIFS, NFS) i obiektowy (REST, SOAP).

- Wdrożenie ujednoliconej pamięci masowej zapewnia następujące korzyści:
  - Zmniejsza koszty inwestycyjne i operacyjne
  - Zarządzane przez pojedynczy interfejs zarządzania
  - Zwiększa wykorzystanie pamięci

# 15. Metody i narzędzia programistyczne stosowane w symulacji komputerowej

**Symulacja komputerowa** — program czy proces modelowania matematycznego przeprowadzany na komputerze, którego zadaniem jest **przewidywanie zachowania lub wyniku układu rzeczywistego, lub fizycznego**. Symulacja komputerowa jako narzędzie naukowe wykorzystywana jest między innymi w meteorologii, fizyce jądrowej, fizyce cząstek elementarnych, inżynierii materiałowej czy mechanice płynów.

Model składa się z równań używanych do uchwycenia zachowania systemu. Natomiast symulacja komputerowa to **faktyczne uruchomienie programu wykonującego algorytmy rozwiązuające te równania, często w sposób przybliżony**. Symulacja jest zatem **procesem uruchamiania modelu**.

W najwęższym znaczeniu symulacja komputerowa to program uruchamiany na komputerze, w którym stosuje się metody krok po kroku w celu zbadania przybliżonego zachowania modelu matematycznego. Zwykle jest to model systemu ze świata rzeczywistego (choć może to być system wyimaginowany lub hipotetyczny). Mówiąc o „symulacji komputerowej” w najwęższym znaczeniu, powinniśmy mówić o konkretnej implementacji algorytmu na konkretnym komputerze cyfrowym, napisanej w określonym języku, przy użyciu określonego kompilatora itp. Mówiąc szerzej, możemy myśleć o symulacji komputerowej jako o kompleksowej metodzie badania systemów.

Modele wykorzystywane do symulacji komputerowych można **klasyfikować według kilku niezależnych par atrybutów**, a wybór techniki modelowania symulacyjnego zależy od konkretnego rozważanego systemu i zamierzonych wyników symulacji, do rozważanych mogą należeć:

- **przewidywanie zdarzeń:**
  - stochastyczne (losowe) — wykorzystują generatorów liczb losowych do modelowania zdarzeń losowych
  - deterministyczne — wynik jest powtarzalny i zależy tylko od danych wejściowych i ewentualnych interakcji ze światem zewnętrznym
- **upływ czasu**
  - dyskretny — czas zwiększa się stałymi przyrostami, a krok czasowy dobiera się optymalnie ze względu na zasobowość systemu, jego wydajność i charakter symulowanego obiektu i/lub zjawiska
  - ciągły — czas zwiększa się stałymi przyrostami, jak w symulacji z czasem dyskretnym, lecz wartości próbek sygnałów są interpolowane dla chwil pośrednich pomiędzy momentami odczytu.
- **liczba użytych komponentów**
  - lokalne — przetwarzanie odbywa się na pojedynczym komputerze
  - rozproszone — wiele komputerów/maszyn, połączonych w sieci lokalnej lub zewnętrznej
- **format danych wyjściowych:**
  - statyczne — wynikiem jest zbiór danych, np. statyczny obraz
  - dynamiczne — wynikiem jest proces przebiegający w czasie np. animacja.

Główne wykorzystuje się 4 metody:

- **Symulacja Monte Carlo** — Ta technika statystyczna wykorzystuje losowość do rozwiązywania problemów probabilistycznych, przy czym losowanie dokonywane jest zgodnie z rozkładem, który musi być znany. Typowym przykładem może być modelowanie wyniku zderzenia cząstki o wysokiej energii z jądrem złożonym, gdzie każdy akt zderzenia elementarnego (z pojedynczym nukleonem jądra) modelowany jest oddzielnie poprzez losowanie liczby, rodzaju, kąta emisji, energii itp. cząstek wtórnych emitowanych w wyniku takiego zderzenia. Następnym etapem jest modelowanie losu każdej z cząstek wtórnych (w wyniku kolejnego losowania prawdopodobieństwa oddziaływanego lub wyjścia z jądra). Kontynuując taką procedurę, można otrzymać pełny opis „sztucznie generowanego” procesu złożonego. Po zebraniu dostatecznie dużej liczby takich informacji można zestawić ich charakterystyki z obserwowanymi wynikami doświadczalnymi, potwierdzając lub negując słuszność poczynionych w całej procedurze założeń.

TLDR; koncentruje się na występowaniu określonych zdarzeń i symuluje, jak system może zachowywać się w czasie. Znajduje zastosowanie w różnych systemach, takich jak systemy stochastyczne, ale jest kosztowny obliczeniowo.

- **Discrete event simulation (DES)** — symulacja dyskretna. W tego typu symulacji symulator utrzymuje kolejkę zdarzeń posortowaną według symulowanego czasu ich wystąpienia, badany proces jest modelowany jako ciąg zdarzeń, przy czym zmiany stanu modelu pojawiają się w określonych punktach czasu. Symulator odczytuje kolejkę i wyzwala nowe zdarzenia w miarę przetwarzania każdego zdarzenia. Nie jest istotne przeprowadzanie symulacji w czasie rzeczywistym. Zadaniem modelującego jest określenie zmiennych stanu, które są istotne z punktu widzenia celu badania oraz zdarzeń, które powodują ich zmiany, a także logicznych powiązań między nimi

TLDR; koncentruje się na wystąpieniu określonych zdarzeń i symuluje zachowanie systemu w określonym czasie. Dokładnie i skutecznie modeluje systemy z małą liczbą zdarzeń, ale nie jest pomocne w modelowaniu systemów z wieloma zdarzeniami.

- **Ciągła symulacja dynamiczna** (ang Continuous Dynamic Simulation) — wykonuje numeryczne rozwiązywanie równań różniczkowo-algebraicznych lub równań różniczkowych (częściowych lub zwyczajnych). Okresowo program symulacyjny rozwiązuje wszystkie równania i wykorzystuje liczby do zmiany stanu i wyniku symulacji. Zastosowania obejmują symulatory lotu, gry symulacyjne budowy i zarządzania, modelowanie procesów chemicznych i symulacje obwodów elektrycznych. Pierwotnie tego rodzaju symulacje faktycznie wdrażano na komputerach analogowych, gdzie równania różniczkowe można było bezpośrednio reprezentować za pomocą różnych komponentów elektrycznych, takich jak wzmacniacze operacyjne.

TLDR; koncentruje się na interakcjach i pętlach sprzężenia zwrotnego między różnymi komponentami systemu w celu symulacji zachowania złożonych systemów

w czasie

- **Agent-based simulation (ASB)** — specjalny rodzaj dyskretnej symulacji, która nie opiera się na modelu z podstawowym równaniem, ale mimo to można ją formalnie przedstawić. W symulacji agentowej poszczególne byty (takie jak cząsteczki, komórki, drzewa lub konsumenci) w modelu są reprezentowane bezpośrednio (a nie przez ich gęstość lub stężenie) i posiadają stan wewnętrzny oraz zestaw zachowań lub reguł, które określają, w jaki sposób stan agenta jest aktualizowany z jednego kroku czasowego do następnego.

TLDR; symuluje zachowanie różnych agentów w systemie. Z łatwością umożliwia modelowanie złożonych systemów z wieloma oddziałującymi agentami, ale jest trudny do sprawdzenia i sparametryzowania.

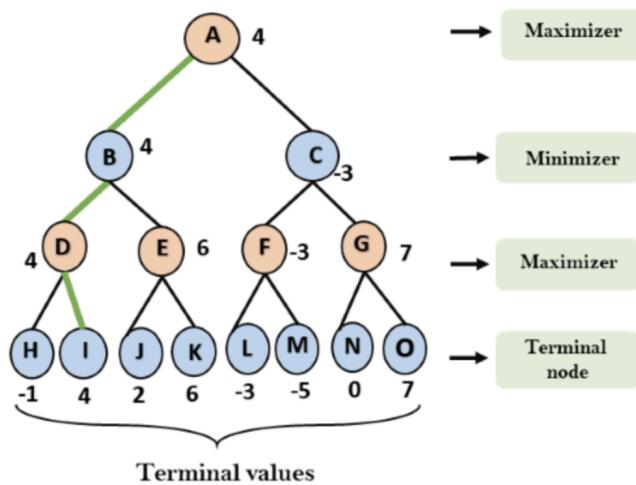
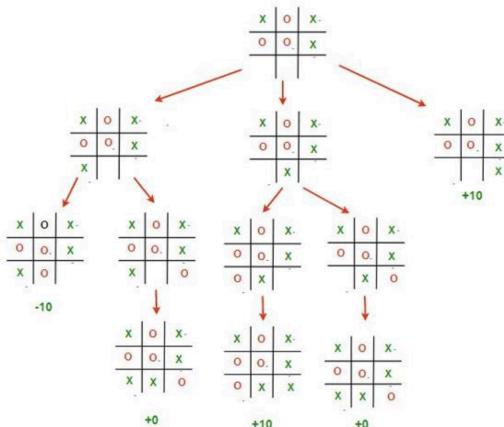
Program symulacyjny może być napisany w dowolnym języku programowania. W ograniczonym zakresie można do symulacji wykorzystać również arkusz kalkulacyjny (ex. Excel). Zostały jednak stworzone specjalne **narzędzia przeznaczone do wykonywania symulacji komputerowych**. Obecnie stosowane są:

- język programowania GPSS — język programowania stosowany do symulacji komputerowych. Najczęstszym zastosowaniem tego języka są symulacje systemów kolejkowych, takich jak np. kas w supermarketach czy linie produkcyjne w fabrykach.
- język python
- SciLab — pakiet naukowy, stworzony do badań matematycznych i posiada w sobie setki funkcji matematycznych, którymi możemy operować zarówno na liczbach, jak i na bardziej zaawansowanych strukturach jak wektory czy macierze.
- FlexSim — pakiet oprogramowania do symulacji zdarzeń dyskretnych. Narzędzia symulacyjne pozwalają na optymalizację obecnych i planowanych procesów, identyfikację i ograniczenie strat, redukcję kosztów oraz zwiększenie przychodów.
- Matlab Simulink — środowisko oparte na diagramie blokowym używane do projektowania systemów z modelami wielodomenowymi, przeprowadzania symulacji przed przejściem na sprzęt i wdrażania bez pisania kodu.
- Matlab — program komputerowy będący interaktywnym środowiskiem do wykonywania obliczeń naukowych i inżynierskich, oraz do tworzenia symulacji komputerowych.
- AnyLogic — wielometodowe narzędzie służące do modelowania oraz symulacji procesów równoległych, które mogą być stosowane do analizy poprawności programów równoległych i rozproszonych.
- Jakikolwiek inne które się podobają.

# 16. Wybrany algorytm sztucznej inteligencji wykorzystywany w problemach związanych z projektowaniem gier komputerowych

## Minmax

Polega na minimalizacji maksymalnego zysku dla przeciwnika oraz maksymalizacji minimalnego zysku dla gracza. Algorytm minimax jest rekurencyjny. Przy każdej rekurencji próbuje obliczyć prawidłową wartość bieżącej pozycji na planszy, analizując każdy możliwy ruch z bieżącej pozycji na planszy. Dla każdego ruchu oblicza wynikową pozycję na planszy i wykonuje rekurencję, aby znaleźć wartość tej pozycji. Aby wyszukiwanie nie trwało wiecznie (w przypadku, gdy drzewo jest bardzo głębokie), algorytm ma maksymalną głębokość wyszukiwania. Jeśli bieżąca pozycja tablicy znajduje się na maksymalnej głębokości, algorytm wywołuje funkcję oceny statycznej i zwraca wynik. Jeśli algorytm rozważa pozycję, w której bieżący gracz ma się poruszyć, zwraca najwyższą wartość, jaką widział; w przeciwnym razie zwraca najniższą. W ten sposób na przemian wykonywane są kroki minimalizacji i maksymalizacji. Jeśli głębokość wyszukiwania wynosi zero, zapisywany jest również najlepszy znaleziony ruch. Będzie to ruch, który należy wykonać.



## Algorytm alfa-beta

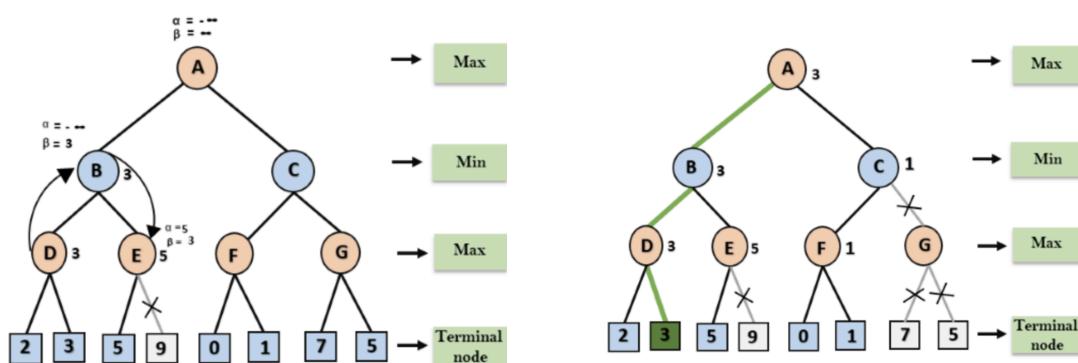
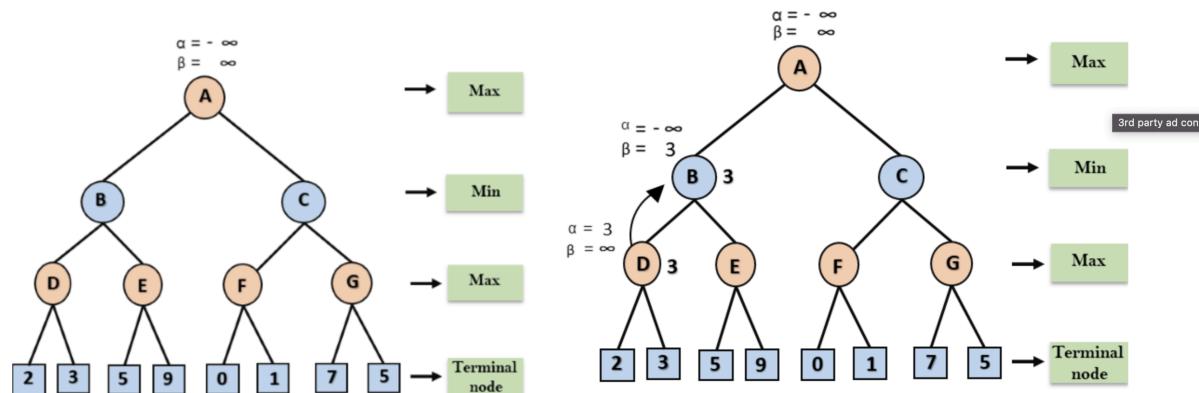
Przycinanie alfa-beta jest zmodyfikowaną wersją algorytmu minimax. Jest to technika optymalizacji algorytmu minimax. Liczba stanów gry, które musi zbadać minimax, jest wykładnicza w głębokości drzewa. Istnieje technika, dzięki której bez sprawdzania każdego węzła drzewa gry możemy obliczyć poprawną decyzję minimaksową, a technika ta nazywana jest przycinaniem. Wiąże się to z dwoma parametrami progowymi alfa i beta dla przyszłej ekspansji, więc nazywa się to przycinaniem alfa-beta. Jest on również nazywany algorytmem alfa-beta. Przycinanie alfa-beta może być stosowane na dowolnej głębokości drzewa, a czasami nie tylko przycina liście drzewa, ale także całe poddrzewo.

Dwa parametry można zdefiniować jako:

- Alfa: Najlepszy (o najwyższej wartości) wybór, jaki znaleźliśmy do tej pory w dowolnym punkcie wzdłuż ścieżki Maximizera. Początkowa wartość alfa wynosi  $-\infty$ .
- Beta: Najlepszy (o najniższej wartości) wybór, jaki do tej pory znaleźliśmy w dowolnym punkcie na ścieżce Minimalizatora. Początkowa wartość beta wynosi  $+\infty$ .

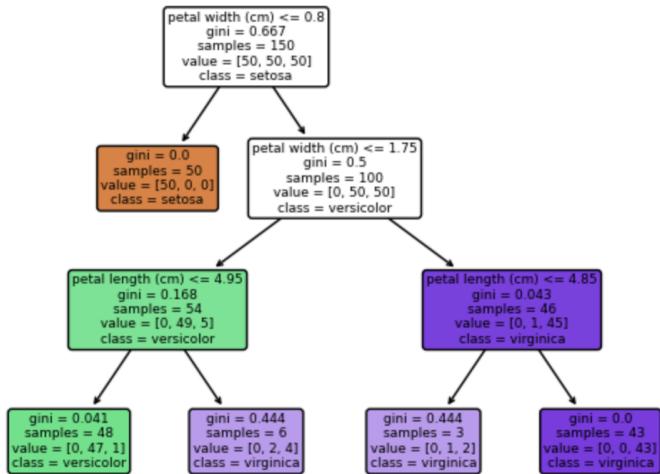
Przycinanie alfa-beta do standardowego algorytmu minimax zwraca ten sam ruch, co standardowy algorytm, ale usuwa wszystkie węzły, które tak naprawdę nie wpływają na ostateczną decyzję, ale spowalniają algorytm. Przycinając te węzły, algorytm staje się szybszy.

$\alpha > \beta$



## Drzewa Decyzyjne

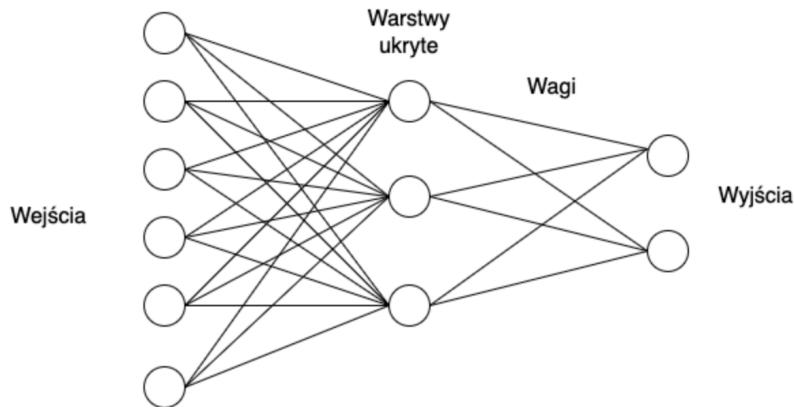
Drzewa decyzyjne są strukturą hierarchiczną. Każde drzewo zawiera: korzeń, gałęzie i liście, które zawierają informacje o tym, która klasa jest najbardziej prawdopodobna. Proces budowania drzewa opiera się o wybranie korzenia i dokonywanie dalszych podziałów. Dzielenie opiera się o wykorzystanie funkcji określającej jakość podziału. W implementacji, którą można znaleźć w module scikit-learn, dostępne są kryteria: gini, entropy, log\_loss. W celu zobrazowania struktury drzew decyzyjnych - przykład przedstawiony na rysunku:.



## Sieci Neuronowe

Sieci neuronowe to zespół neuronów. Pojedynczy neuron można rozpatrywać jako model regresji liniowej z funkcją aktywacji, składający się z danych wejściowych, wag, przesunięć i danych wyjściowych. Wagi pozwalają określić znaczenie każdej próbki, dlatego wejścia są przez nie mnożone, a potem sumowane, decydują o znaczeniu każdego wejścia dla wyjścia neuronu. Dodatkowo, każda warstwa może mieć tzw. bias, który pomaga w dopasowaniu funkcji aktywacji neuronów.

- Warstwa wejściowa: Przyjmuje dane wejściowe i przekazuje je do warstw ukrytych.
- Warstwy ukryte: Neurony w tych warstwach wykonują obliczenia na podstawie danych wejściowych, które są przekazywane z warstwy wejściowej.
- Warstwa wyjściowa: Generuje wynik końcowy na podstawie obliczeń wykonanych przez neurony w warstwach ukrytych.



## Uczenie przez wzmacnianie

Reinforcement Learning, opiera się na strategii prób i błędów, w której inteligentne algorytmy uczą się poprzez interakcje z otoczeniem. Model ten polega na systematycznym i ciągłym doskonaleniu, gdzie algorytmy otrzymują pozytywne nagrody dla poprawnych predykcji, a dla błędnych - karę. Dokonując ciąglej analizy tych negatywnych i pozytywnych wyników, maszyna stopniowo doskonali swój proces uczenia się, dopasowując swoje strategie do osiągnięcia najlepszego rezultatu. To podejście do uczenia maszynowego jest kluczowe w dziedzinach, gdzie maszyna musi samodzielnie nauczyć się odpowiednio reagować na różnorodne, nieprzewidywalne sytuacje.

Pętla procesu uczenia

- Zbieranie danych – na początku iteracji kolekcjonuje się dane o środowisku do bufora w podobny sposób jak we *wstępnym zbieraniu danych*, ale z użyciem kolekcjonującej polityki agenta zamiast polityki losowej.
- Uczenie – po zebraniu danych na temat środowiska używa się ich do obliczenia straty (ang. loss) i wytrenowania agenta.
- Ewaluacja – zazwyczaj podczas pętli uczenia przeprowadza się na bieżąco ewaluację agenta, często nie jest to jednak wymóg.

# 17. Porównanie podejścia strukturalnego i obiektowego do tworzenia oprogramowania

## PODEJŚCIE STRUKTURALNE

**Podejście strukturalne** - przedmiotem zainteresowania są elementy systemu, wzajemne powiązania tych elementów, relacje które w nim zachodzą; definiowane są etykiety-obiekty z których system się składa, strumienie przepływu danych. To podejście strukturalne jest w chwili obecnej najczęściej, najczęściej i najskuteczniej stosowane do praktycznej budowy systemu informatycznego.

W podejściu strukturalnym dąży się do formalnej analizy systemu. W wyniku tej analizy tworzone są hierarchiczne struktury, których elementami są:

- procesy,
- dane,
- związki zachodzące między nimi.

Cechą charakterystyczną tego podejścia jest oddzielne modelowanie danych i procesów, wykorzystujące diagramowe i macierzowe metody i techniki.

Jest stosowana dla systemów informacyjnych organizacji lub dla dobrze widocznej dziedziny problemowej. W odróżnieniu od obiektowej jest prostsza do zrozumienia.

**W metodyce strukturalnej kierować się należy trzema ogólnymi zasadami tj.:**

- **Modelowanie**, tj. wykorzystywanie modeli graficznych do klarownego i jednoznacznego przedstawienia systemu (aktualny system i struktura danych oraz żądany system i struktura danych),
- **Podział na części**, tj. podział systemu na części składowe, co sprzyja lepszemu zrozumieniu jego funkcjonowania oraz umożliwia podział pracy, Iteracja, tj. możliwość wnoszenia poprawek i uzupełnień do pierwszych, często nie w pełni poprawnych wersji
- **reprezentacji aktualnego i żadanego systemu.**

Fazy cyklu życia	Metody i techniki
<b>Analiza potrzeb informacyjnych</b>	<b>Analiza dokumentacji Wywiad Ankieta Analiza dokumentów Obserwacja</b>
<b>Modelowanie funkcji systemu (struktura funkcjonalna systemu)</b>	<b>Diagramy hierarchii funkcji Diagramy zależności funkcji Formularze opisu wymagań</b>
<b>Modelowanie procesów systemu (struktura funkcjonalna systemu)</b>	<b>Diagramy przepływu danych Słowniki procesów, przepływów, obiektów</b>
<b>Modelowanie danych (struktura informacyjna systemu)</b>	<b>Diagramy powiązania danych, Diagramy związków encji Słowniki danych Normalizacja</b>

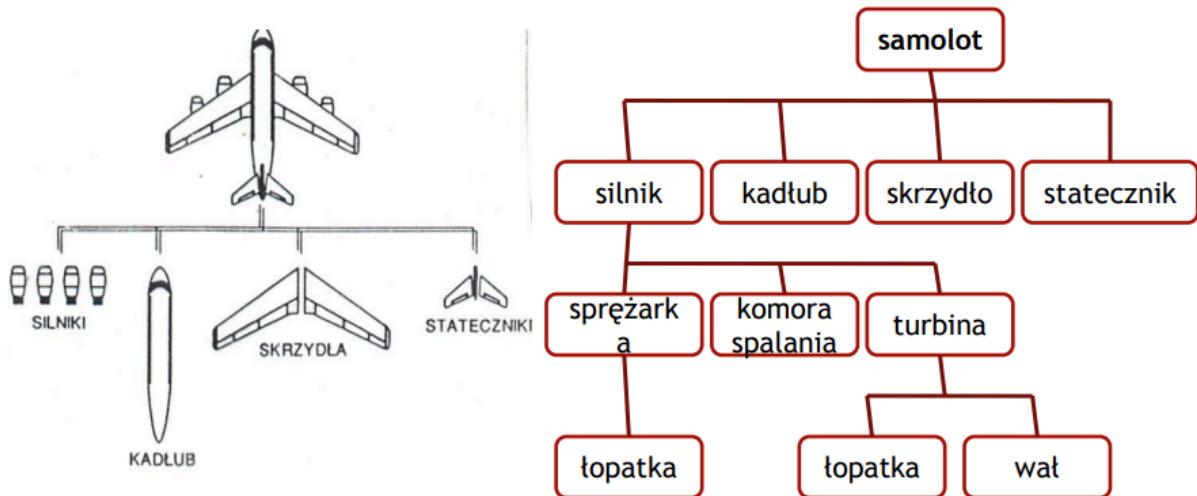
**Zalety:**

- skupione wokół wymagań systemu (firmy),
- elastyczne w reagowaniu na zmiany,
- ułatwiają komunikowanie się z użytkownikiem dzięki czemu może powstać system, którego użytkownik rzeczywiście potrzebuje,
- łatwość wykrycia błędów i luk już na etapie analizy (poprzez weryfikację różnymi metodami),
- opracowane systemy są łatwe do modernizacji (sprzęt na dalszym planie),
- system posiada pełną, spójną dokumentację - ułatwia to eksploatację, modernizację, nie wiąże z jednym analitykiem (projektantem).

**Wady:**

- przesunięcie punktu ciężkości prac na fazę analizy i związane z tym długie oczekивание użytkownika na „namacalne” efekty prac nad systemem





## PODEJŚCIE OBIEKTOWE

Podejście obiektowe - techniczne podejście do analizowania i projektowania aplikacji, systemu lub firmy poprzez zastosowanie programowania obiektowego.

Wyróżniamy:

- **OOA (Object-oriented analysis)**

Główna różnica między analizą obiektową a innymi formami analizy polega na tym, że w podejściu obiektowym organizujemy wymagania wokół obiektów, które integrują zarówno zachowania (procesy), jak i stany (dane) wzorowane na obiektach świata rzeczywistego, z którymi system wchodzi w interakcję.

Analiza obiektowa	Programowanie obiektowe
Określenie funkcjonalnych wymagań systemu	Modele struktury i zachowania systemu (UML)
Koncentruje się na tym co system ma robić	Skupia się na tym jak system działa
Obiektowe modelowanie dziedziny problemu	Odwzorowanie wyników analizy na obiekty i klasy
Identyfikacja i wyodrębnienie obiektów pojęciowych	Luźny związek z językiem implementacji

- **OOD (Object-oriented Design)**

Proces planowania systemu oddziałujących na siebie obiektów. Podczas OOD programista stosuje ograniczenia implementacyjne do modelu koncepcyjnego utworzonego w analizie obiektowej. Takie ograniczenia mogą obejmować platformy

sprzętowe i programowe, wymagania dotyczące wydajności, trwałe przechowywanie i transakcje, użyteczność systemu oraz ograniczenia narzucone przez budżet i czas. Koncepcje w modelu analitycznym niezależnym od technologii są mapowane na klasy implementujące i interfejsy, w wyniku czego powstaje model dziedziny rozwiązania, czyli szczegółowy opis tego, w jaki sposób system ma być zbudowany

Dane wejściowe	Wyniki
• <a href="#">Model pojęciowy</a>	<a href="#">Diagram sekwencji</a>
• <a href="#">Przypadek użycia</a>	<a href="#">Diagram klas</a>
<a href="#">Diagram sekwencji systemu</a>	
<a href="#">Dokumentacja interfejsu użytkownika</a>	

W skrócie:

- Projektant korzysta z wyników fazy analizy
  - modele koncepcyjne w dziedzinie zastosowania
  - słownik pojęć
  - przypadki użycia (scenariusze realizujące wymagane funkcje)
- **Fazy analizy i projektowania są realizowane równolegle**

## Cele (wyniki fazy projektowej)

- Zaprojektowanie klas i związków między tymi klasami
  - odwzorowanie: dziedzina zastosowań → dziedzina rozwiązań
  - specyfikacja interfejsów
  - uwzględnia się pewne wybrane ograniczenia implementacyjne
- Opracowanie modeli obiektowych systemu z różnych perspektyw
  - opisy architektury systemu (elementy systemu i ich powiązania)
  - opisy zachowania systemu (przepływ zleceń między obiektemi)

## KONSEKWENCJE PODEJŚCIA

- Organizacja programu: zbiór oddziałujących na siebie obiektów
  - obiekty integrują stan (dane) i zachowanie (metody)
  - obiekty wywołują wzajem swoje metody
- Program bliski naszemu sposobowi postrzegania rzeczywistości
  - zmniejszenie luk reprezentacji (representational gap)
  - łatwiejszy do opanowania kod
- Podejście obiektowe do procesu tworzenia systemu ułatwia
  - projektowanie dużych i złożonych systemów

- współprace w zespole
- ponowne wykorzystanie istniejącego kodu
- testowanie

❖ KRÓTKO OD AGH

❖ Krótko ode mnie XD (LINK)

## 18. Omów zjawisko overfittingu oraz sposoby jego zapobiegania

**Overfitting** jest zjawiskiem, które występuje w uczeniu maszyn, gdy model uczy się **zbyt dobrze na danych treningowych**, przez co traci zdolność do generalizacji i **działa słabo na nowych, niewidzianych wcześniej danych**. Dzieje się tak, ponieważ model "zapamiętuje" szczegóły i szумy w danych treningowych, zamiast uczyć się ogólnych wzorców.

Jednak zanim głębiej opisane zostanie samo zjawisko...

### Czym jest uczenie maszyn?

Uczenie maszyn oznacza troszeczkę coś innego niż uczenie ludzkie. Człowiek uczy się głównie przez zapamiętywania i walidację wiedzy na podstawie przykładów.

Program komputerowy uczy się na podstawie doświadczeń E dla zadań T i oceny P, gdy ocena P zadania T poprawia się na podstawie E.

Czyli w istocie program uczący to program, w wyniku działania którego wyznaczamy **algorytmy działania systemu**.

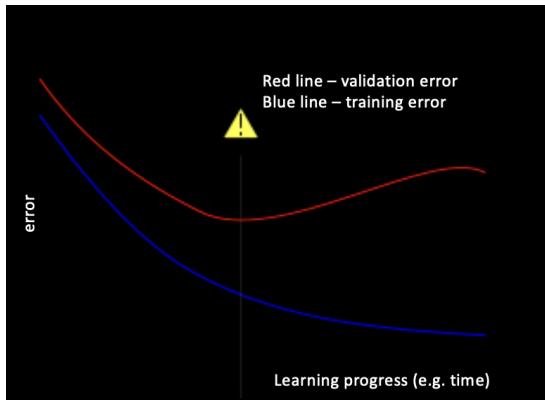
#### Parametry to wiedza lub umiejętności!

Uczenie to **konstruowanie i zmiana reprezentacji dostarczonych faktów**. W przypadku świata przepełnionego algorytmiką z pewnością jest to zadanie bardzo trudne, ponieważ w większości wyuczane pojęcia zakładamy, że należą do pewnej klasy znanej **a priori**, a celem jest znalezienie elementu do niej należącego. Największą trudność stanowi znalezienie hipotezy dokładnie odpowiadającej pojęciu, ponieważ wpływa na to:

- stosowanie hipotez niedokładnych
- stosowanie błędnych etykiet w zbiorze danych
- obecność błędów w wartościach atrybutów obiektów w zbiorze uczącym

Uczenie jest w ogóle możliwe, ponieważ w świecie istnieją regularności i uczenie maszyn pomaga przede wszystkim w wyciąganiu wniosków, eliminowaniu nadmiaru informacji i ich porządkowaniu.

## Definicja overfittingu



### Overfitting - definicja

Przestrzeń hipotez  $H$ ,

$\text{error}_D(h)$  – błąd rzeczywisty  $h$  na rozkładzie  $D$ ,  
 $\text{error}_{\text{train}}(h)$  – błąd (próbki)  $h$  na danych treningowych.

Hipoteza  $h \in H$  jest przeuczona na zbiorze treningowym, jeżeli istnieje hipoteza  $h' \in H$ , dla której

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

i

$$\text{error}_D(h) > \text{error}_D(h')$$

**Overfitting** określa się także jako **nadmierne dopasowanie**, **przetrenowanie modelu** podczas uczenia. Jak pokazano na rysunku, objawia się to przede wszystkim nagłym wzrostem błędu rzeczywistego (**validation error** – błąd określany przez weryfikację modelu z zupełnie oddzielnym zbiorem walidacyjnym) wraz z czasem uczenia, podczas gdy błąd próbki (**training error** – błąd określany przez weryfikację modelu z obiektami ze zbioru uczącego) cały czas spada. W tym właśnie momencie model traci zdolność do generalizacji i zbyt bardzo dopasowuje się do danych uczących.

### Przyczyny overfittingu

1. **Zbyt skomplikowany model:** Modele o dużej liczbie parametrów (np. głębokie sieci neuronowe) mają dużą zdolność do adaptacji, co może prowadzić do overfittingu.
2. **Niewystarczająca ilość danych treningowych:** Gdy dane treningowe są ograniczone, model może nie być w stanie nauczyć się ogólnych wzorców i skupić się na specyficznych przykładach.
3. **Szumy w danych:** Szumy lub błędy w danych mogą być traktowane przez model jako istotne wzorce, co prowadzi do overfittingu.
4. **Zbyt długie trenowanie modelu:** Zbyt długi czas trenowania modelu może prowadzić do overfittingu, ponieważ model ma więcej okazji do nauki specyficznych szczegółów danych treningowych.

### Sposoby zapobiegania

Konieczna jest świadomość, że można sobie radzić z tym problemem i można wykorzystać:

- **Mniej skomplikowane modele uczenia maszynowego** – z mniejszą liczbą parametrów. Tutaj wyróżnić można przykładowo **pruning** – w przypadku sieci neuronowych to usuwanie niepotrzebnych neuronów lub gałęzi, aby zmniejszyć złożoność modelu. Pruning również stosuje się do drzew decyzyjnych (**rule-post pruning, reduce-error pruning**) i innych modeli.

W tym sposobie należy **uważyć z upraszczaniem modeli**, ponieważ zwyczajnie mogą przestać odpowiadać zagadnieniu (wtedy już można mieć do czynienia z **underfitting**).

Jeszcze jedną techniką w **przypadku zespołów klasyfikatorów** może być sama

**selekcja klasyfikatorów** i wszystkie kombinacje składowych modeli wchodzących w skład zespołu. Czasami zwyczajnie model może być przekombinowany i warto go upraszczać.

Kolejną użyteczną techniką przeciwdziałania (tym razem zorientowaną na danych) może być cały profesjonalny proces **przygotowania danych do uczenia**, czyli przeciwdziałanie kłopotliwie wielowymiarowości:

- **czyszczenie, transformacja i unifikacja** danych, a także **usuwanie szumu**
- **dyskretyzacja danych**
- **selekcja i redukcja cech**
- **selekcja przykładów** w zbiorze uczącym

- **Więcej danych treningowych** – zwiększenie ilości danych treningowych może pomóc modelowi nauczyć się bardziej ogólnych wzorców. Tutaj można również wykorzystać różne techniki jak: **augmentacja danych** – polegająca na sztucznym zwiększaniu liczby danych treningowych poprzez modyfikację istniejących, np. poprzez **obracanie, skalowanie** czy **zmiany kolorów** obrazów.
- **Techniki regularyzacji** – jak chociażby **dropout**, który losowo dezaktywuje neurony lub inne składowe modeli podczas uczenia wprowadzając pewną losowość w cały proces.

Inna technika to **L1 i L2 regularyzacja** – Dodanie **kar za duże wartości wag modelu** może pomóc w zmniejszeniu złożoności modelu.

- **Monitorowanie wydajności** modelu na danych walidacyjnych podczas trenowania i **zatrzymanie trenowania**, gdy wydajność przestaje się poprawiać.

Metoda wydzielania, która bazuje na podziale danych na trzy podzbiory: **treningowy, walidacyjny i testowy**. Zbiór treningowy służy do uczenia modelu. Zbiór walidacyjny służy do weryfikacji skuteczności, natomiast dopiero w momencie osiągnięcia satysfakcjonujących wyników wykorzystuje się zestaw testowy do ostatecznej weryfikacji modelu.

**Walidacja krzyżowa** – techniki takie jak **k-krotna walidacja krzyżowa** mogą pomóc w ocenie wydajności modelu na różnych podzbiorach danych, zapewniając jego dobre uogólnienie. W tej metodzie dane są dzielone na K części i model jest trenowany K razy, za każdym razem używając innej części jako danych walidacyjnych.

Istnieją różne strategie dla kryterium stopu trenowania, które zależą przede wszystkim od charakteru wykorzystywanego modelu.

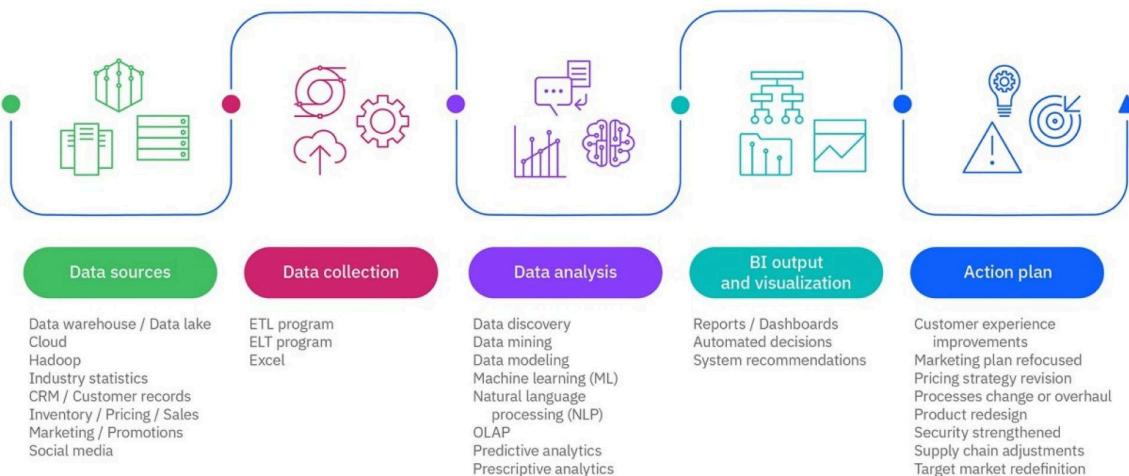
# 19. Etapy tworzenia systemów analityki biznesowej

**Business Intelligence** (analityka biznesowa) – proces przekształcania danych w informacje, a informacji w wiedzę, która może być wykorzystana do zwiększenia konkurencyjności przedsiębiorstwa.

**Systemy analityki biznesowej** — zgodnie z nazwą zajmują się określeniem potrzeb biznesowych, są to połączenia dedykowanych narzędzi, oprogramowania oraz wiedzy eksperckiej. Dostarczane są często w formie pakiety rozwiązań

BI można podzielić na główny 5/6 etapów:

- Zdefiniowanie problemu – należy precyzyjnie wskazać i opisać problem/ będący przedmiotem analizy biznesowej. W tym przypadku identyfikacja danych, które mają być przeglądane i analizowane, na przykład z hurtowni danych lub jeziora danych, chmury, Hadoop, statystyk branżowych, łańcucha dostaw
- Gromadzenie danych – dane potrzebne do przeprowadzenia Business Intelligence mogą być przechowywane w wielu różnych systemach i formatach, co utrudnia ich zebranie w jednym miejscu oraz dokonanie ich poprawnej analizy. Etap ten obowiązuje identyfikację wszystkich źródeł danych i pozyskanie z nich informacji, które są potrzebne do analizy. Przygotowanie danych może polegać na ręcznym gromadzeniu informacji w arkuszu kalkulacyjnym lub w programie do automatycznego wyodrębniania, przekształcania i ładowania (ETL).
- Sortowanie danych – czyszczenie i agregacja zbioru danych z informacji, które znalazły się w nim przypadkowo albo mają niską wartość i jakość. Po segregacji zbiór danych zaleca się uporządkować wg ustalonych kryteriów.
- Eksploracja i pogłębiona, właściwa analiza danych – etap obejmujący identyfikację cech i wzorców. Dotyczy on także analizy zebranego materiału analitycznego.
- Przygotowanie raportu – to finałowy element analizy biznesowej, polegający na raportowaniu i przedstawieniu wniosków w formie czytelnej prezentacji.
- (NIE ZAWSZE PODAJĄ) Podejmowanie decyzji – decydenci i zainteresowane strony na podstawie raportu, danych i zdobytych informacji określają najlepszy sposób działań dla firmy.



# 20. Ciągła integracja i automatyzacja procesu tworzenia oprogramowania.

Aby rozpocząć temat automatyzacji procesu tworzenia oprogramowania to na początku trzeba się zapoznać jak ten proces wygląda.

## Software Development and Design

### Software Development Life Cycle (SDLC)

- SDLC is the process of developing software, starting from an idea and ending with delivery. This process consists of six phases. Each phase takes input from the results of the previous phase.
- SDLC is the process of developing software, starting from an idea and ending with delivery. This process consists of six phases. Each phase takes input from the results of the previous phase.
- Although the waterfall methods is still widely used today, it's gradually being superseded by more adaptive, flexible methods that produce better software, faster, with less pain. These methods are collectively known as "Agile development."



1. Wymagania i analiza - w tym etapie definiujemy wymagania, potrzeby co do danego produktu
2. Design - deweloperzy wraz z architektami oprogramowania definiują dokumenty, które opisują architekturę produktu.
3. Implementacja - deweloperzy implementują zaprojektowany produkt, który spełnia początkowe założenia
4. Testowanie - testowane są funkcjonalności produktu
5. Deployment - aplikacja zostaje zainstalowana w środowisku produkcyjnym. Finalnie po takim deploymencie release manager dokonuje release'u produktu dla użytkowników końcowych.
6. Utrzymanie - faza, w której naprawiane są bugi, dostarczane jest wsparcie dla użytkowników w przypadku jakiejś awarii, aplikacja jest udoskonalana i wdrażane są nowe funkcjonalności np na prośbę klientów

Obecnie większość teamów deweloperskich pracuje wykorzystując zwinne metodyki pracy przez co ten proces wytwarzania oprogramowania zaczyna koło. Aby zminimalizować czas potrzebny na wydanie nowego oprogramowania bardzo popularnym podejściem jest automatyzacja wielu procesów wchodzących w skład SDLC.

#### **DEVELOPMENT:**

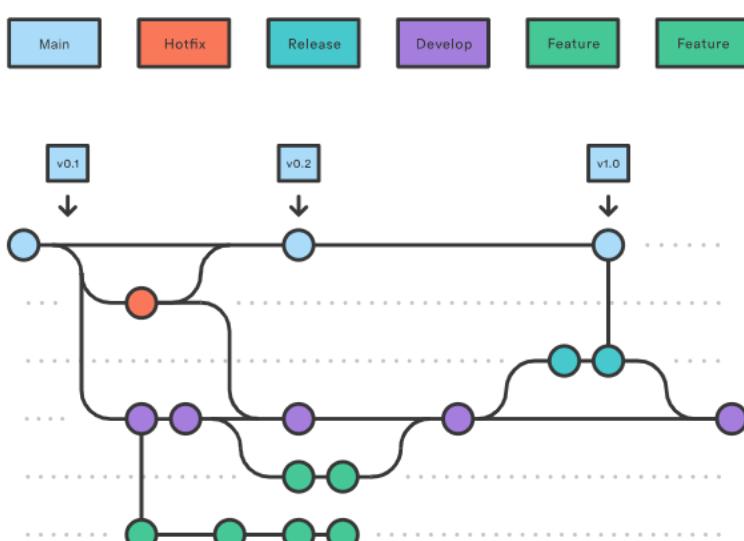
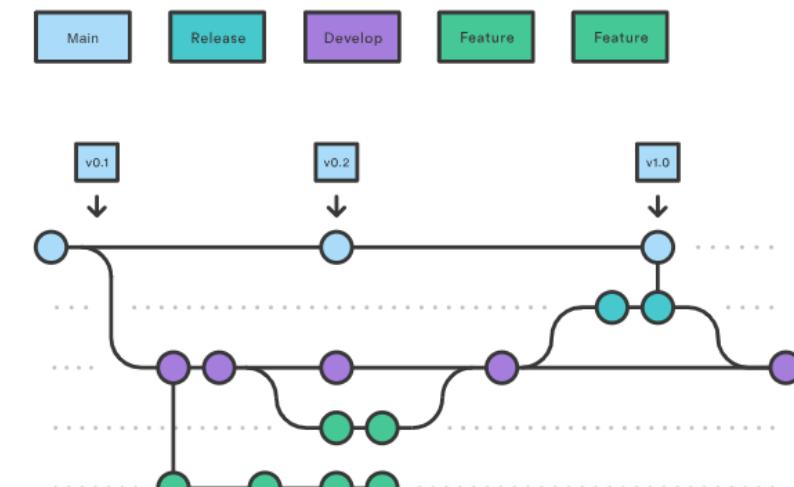
Deweloperzy piszący kod muszą gdzieś go przechowywać a serwery FTP i przechowywanie lokalnie na komputerze to nie najlepsze rozwiązanie.

Zaczniemy zatem od miejsca gdzie kod jest przechowywany czyli od systemu kontroli wersji.

Obecnie najpopularniejszym z nich jest Git. Jest on scentralizowanym systemem kontroli wersji oznacza to że nasz kod jest przechowywany nie tylko lokalnie lecz też w zdalnym repozytorium

W celu uproszczenia procesu tworzenia oprogramowania zdefiniowano kilka workflowów, które upraszczają poruszanie się po Gicie:

1. Gitflow Workflow - bazuje na tym że dla ficzerów, releasów tworzymy oddzielne branche. na masterze znajdują się commity, które odpowiadają releasesom a na branchu development znajdują się ficzery.



2. Feature Branching - Podstawową ideą stojącą za Feature Branch Workflow jest to, że cały rozwój funkcji powinien odbywać się w dedykowanej gałęzi zamiast w gałęzi głównej. Taka hermetyzacja ułatwia wielu programistom pracę nad konkretną funkcją bez zakłócania głównej bazy kodu. Oznacza to również, że główna gałąź nigdy nie

będzie zawierać uszkodzonego kodu, co jest ogromną zaletą dla środowisk ciągłej integracji.

3. Forking Workflow - zasadniczo różni się od innych popularnych przepływów pracy Git. Zamiast używać pojedynczego repozytorium po stronie serwera jako "centralnej" bazy kodu, daje każdemu programiście własne repozytorium po stronie serwera. Oznacza to, że każdy współautor ma nie jedno, ale dwa repozytoria Git: prywatne lokalne i publiczne po stronie serwera. Forking Workflow jest najczęściej spotykany w publicznych projektach open source.
4. Centralized Workflow - jest zasadniczo blokiem konstrukcyjnym dla innych przepływów pracy Git. Większość popularnych przepływów pracy Git będzie miała jakiś rodzaj scentralizowanego repozytorium, z którego poszczególni programiści będą przesyłać i pobierać dane. Poniżej pokrótko omówimy kilka innych popularnych przepływów pracy Git. Te rozszerzone przepływy pracy oferują bardziej wyspecjalizowane wzorce w odniesieniu do zarządzania gałęziami rozwoju funkcji, poprawek i ostatecznego wydania.

Ważnym elementem procesu developmentu jest code review gdzie deweloperzy dokonują review innemu programiście w celu zachowania czystego kodu oraz znalezienia potencjalnych błędów na tym etapie co umożliwia szybką naprawę.

Pomocnymi narzędziami w procesie rozwoju oprogramowania mogą być tools do statycznej analizy kodu, które analizują kod i dostarczają programiście informacji na temat potencjalnych problemów bądź potencjalnych awarii. Pozwala to na wykrywanie błędów na etapie rozwoju oprogramowania oraz utrzymanie czystego kodu.

Przykładowe tools to:

- Sonarqube
- FindBugs,
- ESLint
- pylint

## TESTOWANIE:

Gdy nasz kod mamy już w repozytorium można zacząć kolejny etap jakim jest testowanie oprogramowania. Można wyróżnić wiele rodzajów testów

1. Testy jednostkowe - są wykonywane na bardzo niskim poziomie, blisko źródła aplikacji. Polegają one na testowaniu poszczególnych metod i funkcji klas, komponentów lub modułów używanych przez oprogramowanie. Testy jednostkowe można zazwyczaj dość tanio zautomatyzować i bardzo szybko uruchomić przy pomocy serwera ciągłej integracji.
2. Testy integracyjne - pozwalają sprawdzić, czy poszczególne moduły lub usługi wykorzystywane przez aplikację dobrze ze sobą współpracują. Przykładem może być testowanie interakcji z bazą danych lub upewnienie się, że

mikrousługi współpracują ze sobą w oczekiwany sposób. Tego typu testy są droższe do przeprowadzenia, ponieważ wymagają uruchomienia wielu części aplikacji.

3. Testy funkcjonalne - kładą nacisk na wymagania biznesowe aplikacji. Weryfikują one jedynie dane wyjściowe czynności i nie sprawdzają stanów pośrednich systemu podczas jej wykonywania.
4. Testy kompleksowe - zachowanie użytkownika w oprogramowaniu jest replikowane w kompletnym środowisku aplikacji. Pozwalają one zweryfikować, czy różne przepływy użytkowników działają zgodnie z oczekiwaniemi. Mogą być proste, np. załadowanie strony internetowej lub zalogowanie się, oraz znacznie bardziej złożone, np. weryfikacja powiadomień e-mail, płatności online itp.
5. Testy akceptacyjne - to formalne testy weryfikujące, czy system spełnia wymagania biznesowe. Wymagają one uruchomienia całej aplikacji na czas testowania i skupiają się na replikowaniu zachowań użytkowników. Mogą jednak także mierzyć wydajność systemu oraz odrzucać zmiany w przypadku niespełnienia określonych celów.
6. Testy wydajności - oceniają, jak system zachowuje się przy określonym obciążeniu. Testy te pomagają zmierzyć niezawodność, szybkość, skalowalność i responsywność aplikacji. W ramach testu wydajności można na przykład obserwować czasy odpowiedzi przy wykonywaniu dużej liczby żądań lub określić, jak system zachowuje się przy znacznej ilości danych. Tego typu testy pozwalają m.in. ustalić, czy aplikacja spełnia wymagania dotyczące wydajności, zlokalizować wąskie gardła czy zmierzyć stabilność podczas największego natężenia ruchu.
7. Testy bezpieczeństwa - aplikacja testowana jest pod względem potencjalnych podatności.
8. Testy smoke - to podstawowe testy sprawdzające podstawową funkcjonalność aplikacji. Mają być szybkie do wykonania, a ich celem jest zapewnienie, że główne funkcje systemu działają zgodnie z oczekiwaniemi.

## **DEPLOYMENT:**

Kolejny etap to deployment aplikacji.

W tym etapie aplikacja jest budowana i artefakty otrzymane w tym procesie (np jar czy jakikolwiek wykonywalny plik czy to kontener) zostają opublikowane do jakiegoś zdalnego repozytorium (Nexus, JFrog). Kolejno artefakty pobierane są na

serwer i są instalowane na wybranej platformie (może to być system linux czy to platforma umożliwiająca uruchamianie kontenerów np docker, lxc, podman czy jakiś orchestrator kontenerów np kubernetes, docker-swarm, Nomad).

Wymienione etapy są automatyzowane często z wykorzystaniem potoków CI/CD

CI (Continuous Integration) - jest to automatyzacja procesu integracji oprogramowania tzn zazwyczaj uruchamiany w momencie wrzucenia jakiś zmian do repozytorium wtedy uruchamiane automatycznie są testy i w momencie gdy wszystkie przejdą pomyślnie jest możliwość zmergowania tych zmian do repo. Gdy nie zmiana nie przejdzie testów wówczas jest to informacja dla dewelopera, że należy coś poprawić.

CD (Continuous Deployment) - składa się z takich elementów jak:

1. pobranie kodu z repozytorium
2. zbudowanie aplikacji
3. wrzucenie artefaktów z budowania do jakiegoś repozytorium np JFrog
4. Instalacja artefaktów na środowisku produkcyjnym

Najpopularniejszymi narzędziami CI/CD są:

1. Jenkins
2. Github Actions
3. Gitlab CI/CD
4. Bamboo
5. CircleCI
6. Travis-CI
7. Większość popularnych platform chmurowych ma swoje dedykowane narzędzia do procesu CI/CD np AWS ma CodeBuild, CodePipeline, CodeDeploy

Generalnie większość robi mniej więcej to samo ale warto zaznaczyć, że Jenkins to najbardziej skomplikowane narzędzie umożliwiające wiele różnych konfiguracji i fajnie się integruje z jakimiś funkcjami w Groovy ale no zazwyczaj jest self-hosted przez to coraz częściej wybierane są rozwiązania SaaS, które nie wymagają zarządzania tym narzędziem. Rozwiązaniami hybrydowymi są np Github Actions lub Gitlab CI/CD, które umożliwiają prace w trybie SaaS lub self-hosted.