

Contents

FIZYKA	2
ALGORYTMY I STRUKTURY DANYCH	18
ARCHITEKTURA KOMPUTERÓW	23
BAZY DANYCH	28
INŻYNIERIA OPROGRAMOWANIA.....	33
SZTUCZNA INTELIGENCJA	52
BEZPIECZEŃSTWO INFORMACJI.....	57
TECHNOLOGIE INTERNETOWE I MOBILNE.....	67
PROGRAMOWANIE	69
INTERNET RZECZY, SIECI KOMPUTEROWE	85
GRAFIKA KOMPUTEROWA W GRACH	92

FIZYKA

1. Wyjaśnij pojęcie ładunku elektrycznego, opisz jakimi charakteryzuje się właściwościami.

Link: https://pl.wikipedia.org/wiki/%C5%81adunek_elektryczny

Ladunek elektryczny ciała (lub układu ciał) – fundamentalna właściwość materii przejawiająca się w oddziaływaniu elektromagnetycznym ciał obdarzonych tym ładunkiem. Ciała obdarzone ładunkiem mają zdolność wytwarzania pola elektromagnetycznego oraz oddziaływania z tym polem.

Ladunek elektryczny ciała może być dodatni lub ujemny. Dwa ładunki jednego znaku odpychają się, a pomiędzy ładunkiem dodatnim i ujemnym działa siła przyciągająca.

Ladunki elektryczne są skwantowane, elektronowi przypisano elementarny ładunek ujemny, protonowi dodatni. Oddziaływania naładowanych cząstek elementarnych bada elektrodynamika kwantowa, opisuje się je za pomocą wymiany fotonu.

Często używa się skrótowego pojęcia ładunek elektryczny dla ciała obdarzonego ładunkiem elektrycznym.

Uporządkowany ruch ładunków elektrycznych nazywany jest prądem elektrycznym.

Właściwości:

Oprócz istnienia dwóch rodzajów ładunków odkryto szereg innych właściwości ładunków elektrycznych.

Ladunek jest skwantowany. Oznacza to, że ładunek elektryczny jest wielkością dyskretną i istnieje najmniejsza porcja ładunku, jaką może posiadać ciało. W układzie jednostek SI ta najmniejsza porcja wynosi około $e=1,602 \cdot 10^{-19} C$. Żadna swobodna cząstka nie może mieć ładunku mniejszego niż ten i dlatego ładunek dowolnego ciała musi być całkowitą wielokrotnością tej porcji.

Wielkość ładunku nie zależy od jego rodzaju. Inaczej mówiąc, najmniejszy możliwy ładunek dodatni (z dokładnością do czterech miejsc znaczących) wynosi $+1,602 \cdot 10^{-19} C$, a najmniejszy możliwy ładunek ujemny $-1,602 \cdot 10^{-19} C$; ich wartości są identyczne. Takie po prostu są prawa fizyki w świecie.

Ladunek jest zachowany. Ładunek elektryczny nie może być stworzony ani zniszczony; może tylko zostać przeniesiony z jednego miejsca w drugie, z jednego ciała na drugie. Często mówimy, że ładunki „znoszą się”, to skrót myślowy. Oznacza, że gdy dwa ciała obdarzone równymi, ale przeciwnymi znaków ładunkami znajdują się blisko siebie, to wówczas (przeciwne skierowane) siły wywierane na inne naładowane ciała znoszą się, dając wypadkową siłę równą zero. Ważne, żeby zrozumieć, że ładunki ciał w żaden sposób nie znikają, natomiast całkowity ładunek we wszechświecie jest stały.

Ladunek jest zachowany w układach izolowanych. W zasadzie, gdy ujemny ładunek zniknie z twojego laboratorium i pojawi się na Księżyku, to zasada zachowania ładunku jest spełniona. Jednak tak się nigdy nie dzieje. Jeżeli całkowity ładunek, jaki znajduje się w lokalnym układzie na twoim stole laboratoryjnym zmienia się, to istnieje mierzalny przepływ ładunku do lub z układu. Ponownie, ładunki przemieszczają się, a ich wpływy znoszą się, ale całkowity ładunek w twoim otoczeniu (jeżeli jest izolowane) zostaje zachowany. Ostatnie dwa punkty odnoszą się do zasady zachowania ładunku (ang. law of conservation of charge).

ZAPAMIĘTAJ:

Najmniejszym ładunkiem swobodnie spotykany w przyrodzie jest ładunek elementarny.

Wartość ładunku elementarnego wynosi około $1,602 \cdot 10^{-19}$ kulombów.

Elektron obdarzony jest właśnie ujemnym ładunkiem elementarnym.

Pomiar ładunku elementarnego jako pierwszy dokonał Robert Andrews Millikan. Elektron ma masę równą około $9,11 \times 10^{-31}$ kilogramów.

2. Podstawowe pojęcia obwodów elektrycznych: prąd, potencjał, napięcie, energia, moc, rezystancja, impedancja.

Prąd elektryczny – uporządkowany ruch ładunków elektrycznych.

Link: http://www.fizykon.org/elektrycznosc/el_co_to_jest_prad_elektryczny.htm

Ładunki elektryczne, to zwykle cząstki, które potrafią wytwarzać pole elektryczne. Prąd tworzyć mogą zarówno ładunki dodatnie (np. jony dodatnie: jon wodoru, jon siarczany itp), jak i ujemne (np. elektrony, czy jony ujemne w rodzaju jonu OH⁻) (więcej na temat ładunków elektrycznych znajduje się w tematach poświęconych elektrostatyce).

Cząstki mogą poruszać się w różny sposób. Jak wiadomo z teorii kinetyczno cząsteczkowej (kinetyczno molekularnej). Wszystkie atomy i cząsteczki w naszym otoczeniu są w nieustannym ruchu. Ten ruch, bez względu na to, czy atomy są naładowane (nazywają się wtedy jonami), czy nie jeszcze nie tworzy prądu elektrycznego. Wynika to z faktu, że średnio tyle samo cząstek naładowanych porusza się np. w lewą, co i w prawą stronę i całkowity bilans "wychodzi na zero".

Prąd pojawia się dopiero wtedy, gdy w tym ruchu chaotycznym zostanie wyróżniony jakiś kierunek, preferujący poruszanie się w jakąś stronę. Najczęściej wyróżnienie kierunku w ruchu ładunków odbywa się poprzez przyłożenie pola elektrycznego.

Po przyłożeniu pola elektrycznego na ładunki zaczyna działać siła elektryczna – na ładunki dodatnie sił o zwrocie zgodnym z kierunkiem pola, a na ładunki ujemne siła o zwrocie przeciwnym.

I dopiero takie uporządkowane - w jednym kierunku - przesuwanie się ładunków tworzy prąd elektryczny.

Potencjał pola elektrostatycznego – stosunek energii potencjalnej, jaką miałby ładunek umieszczony w danym punkcie pola do wartości tego ładunku. Stosunek [iloraz] energii potencjalnej [E p] ładunku q znajdującego się w danym punkcie pola elektrostatycznego do wartości tego ładunku nazywamy potencjałem elektrostatycznym tego punktu [V].

Wynika z niego, że potencjał zależy od ładunku źródłowego [Q] i odległości punktu pola od tego ładunku. Znak ładunku decyduje o tym, czy potencjał jest dodatni, czy ujemny. Potencjał jest wielkością skalarną [wyrażoną tylko wartością] i z tego powodu potencjał w punkcie, w którym nakładają się dwa lub więcej pól, jest sumą algebraiczną potencjałów pól składowych.

Napięcie jest to ciśnienie źródła zasilania obwodu elektrycznego, które przepycha naładowane elektrony (prąd elektryczny) przez pętlę przewodzącą, umożliwiając wykonanie pracy, np. oświetlenie pomieszczenia.

W skrócie napięcie ciśnienie i jest mierzone w woltach (V). Termin ten upamiętnia włoskiego fizyka Alessandro Voltę (1745–1827), wynalazcę stosu galwanicznego — pierwowzoru dzisiejszej baterii.

Energia (gr. ενέργεια energeia od ergon „praca”) – skalarna wielkość fizyczna charakteryzująca stan układu fizycznego (materii)[1][2] jako jego zdolność do wykonania pracy[3].

Energia występuje w różnych postaciach np.: energia kinetyczna, energia potencjalna, energia sprężystości, energia cieplna, energia jądrowa.

Energia może zmieniać swoją postać, jednak nie może być tworzona ani niszczona (zasada zachowania energii). Np. produkcja energii w elektrowni węglowej oznacza tylko przekształcenie energii chemicznej w elektryczną.

Z punktu widzenia termodynamiki niektóre formy energii są funkcjami stanu i potencjałami termodynamicznymi[4]. Energia i jej zmiany opisują stan i wzajemne oddziaływanie obiektów fizycznych (ciąż, pól, cząstek, układów fizycznych)[1][2], przemiany fizyczne i chemiczne oraz wszelkiego rodzaju procesy występujące w przyrodzie[4]. W termodynamice, energię która może zostać zamieniona na pracę w określonych warunkach nazywa się energią swobodną.

Energia jest wielkością addytywną[4].

Moc określa tempo wykonania pracy w czasie (lub podobnie, tempo przekazywania energii w czasie). Możliwość pomiaru mocy była jednym z kluczowych czynników w procesie projektowania i konstrukcji maszyny parowej, której pojawienie się doprowadziło do rewolucji przemysłowej. Porównanie mocy maszyny parowej z mocą np. konia pozwoliło właścicielom fabryk podejmować decyzję o zakupie maszyn. W czasach współczesnych pojęcie mocy pomaga nam zrozumieć w jaki sposób najskuteczniej wykorzystywać dostępne zasoby energii.

Rezystancja to wielkość charakteryzująca opór, jaki dany przedmiot stawia przepływającemu prądowi. Mimo iż brzmi to groźnie, to w rzeczywistości można ją wykorzystać do własnych celów. Dzięki rezystancji możliwe jest wytwarzanie ciepła i światła, zmniejszanie przepływu prądu, gdy jest to konieczne, oraz dostarczanie do urządzeń prądu o odpowiednim napięciu.

Kiedy na przykład elektrony płyną przez żarnik żarówki, natykają tak duży opór, że znacznie zwalniają. Gdy przedzierają się między atomami żarnika, atomy te gwałtownie się ze sobą zderzają, wydzielając ciepło, które wytwarza światło żarówki.

Wszystko stawia przepływającym elektronom jakiś opór, nawet najlepsze przewodniki (w istocie istnieje pewna grupa materiałów, które nie stawiają żadnego oporu, nazywają się one nadprzewodnikami, ale swoje właściwości zyskują dopiero w bardzo niskich temperaturach i w tradycyjnej elektronice ich się nie używa). Im wyższa rezystancja, tym bardziej ograniczony przepływ prądu.

Impedancia – wielkość charakteryzująca zależność między natężeniem prądu i napięciem w obwodach prądu zmennego. Odpowiada ona oporowi elektrycznemu(rezystancji) w obwodach prądu stałego.

Zacznijmy do najtrudniejszego słowa z naszego słowniczka – impedancja. Słuchawki mają na celu dostarczyć odbiorcy dźwięk z wybranego sprzętu zewnętrznego. Potrzebne są do tego prąd i napięcie. Wielkość określająca zależność pomiędzy natężeniem prądu a napięciem w obwodach prądu przemiennego to właśnie opór pozorny (impedanca) wyrażony w omach (Ω).

3. *Pole magnetyczne, pole elektryczne, pole elektromagnetyczne – opisz właściwości i zastosowanie w architekturze komputera.*

Pole elektromagnetyczne to połączony efekt pól magnetycznego i elektrycznego. Pole elektromagnetyczne występuje zawsze i wszędzie – energia towarzysząca zjawiskom elektromagnetycznym, to jedna z najstarszych form energii we Wszechświecie, która była jednym z czynników kształtujących ewolucję Ziemi. Źródłem pola elektromagnetycznego są także wszelkie organizmy żywe, w tym człowiek.

Sztuczne pole elektromagnetyczne powstaje wszędzie tam, gdzie płynie prąd: każde gniazdko elektryczne jest jego źródłem. A fale radiowe, które są falami elektromagnetycznymi, wykorzystuje się od ponad 120 lat do łączności: od bezprzewodowego telegrafo przez radio, telewizję aż po wifi czy bluetooth.

Pole elektromagnetyczne w różnych miejscach ma różne częstotliwości i długości fal. W telekomunikacji wykorzystywane są fale, które przenoszą stosunkowo niewiele energii – fale o częstotliwościach niejonizujących.

Pole elektryczne – pole wektorowe określające w każdym punkcie siłę działającą na jednostkowy, spoczywający ładunek elektryczny. Pole elektryczne wytwarzane jest przez ładunki elektryczne oraz zmieniające się pole magnetyczne.

Koncepcję pola elektrycznego wprowadził Michael Faraday (w połowie XIX wieku) jako opis oddziaływania ładunków elektrycznych. Z biegiem czasu okazało się, że pole elektryczne ma dużo szersze znaczenie.

Ladunek poruszający się wytwarza nie tylko pole elektryczne, ale również pole magnetyczne. W ogólności oba te pola nie mogą być traktowane oddzielnie – mówi się wtedy o polu

elektromagnetycznym. Podstawowymi prawami opisującymi pole elektromagnetyczne są równania Maxwella. Nośnikami oddziaływań elektromagnetycznych są fotony.

Zachowawczość pola elektrycznego

Siły elektryczne wytworzone przez spoczywające lub poruszające się ruchem jednostajnym ładunki są zachowawcze, czyli praca wykonana przy przesunięciu ładunku w polu elektrycznym na drodze zamkniętej jest równa zeru. Często krótko nazywa się zachowawczym samo pole elektryczne ładunków spoczywających zwane polem elektrostatycznym.

Wynikiem zachowawczości pola elektrycznego jest jego potencjalność (istnienie energii potencjalnej i potencjału) oraz bezwirowość. Obie te cechy są matematycznie równoważne z zachowawczością.

Pole elektryczne wytworzone przez zmieniające się pole magnetyczne nie jest zachowawcze i powinno być rozpatrywane wspólnie z polem magnetycznym jako pole elektromagnetyczne.

Źródłowość pola elektrycznego

Pole elektryczne wytworzone przez ładunki elektryczne jest polem źródłowym, linie sił tego pola rozpoczynają się i kończą na ładunkach. Matematycznym wyrazem źródłowości pola elektrycznego jest prawo Gaussa.

Pole magnetyczne – stan przestrzeni, w której siły działają na poruszające się ładunki elektryczne, a także na ciała mające moment magnetyczny niezależnie od ich ruchu. Pole magnetyczne, obok pola elektrycznego, jest przejawem pola elektromagnetycznego. W zależności od układu odniesienia, w jakim znajduje się obserwator, to samo zjawisko może być opisywane jako objaw pola elektrycznego, magnetycznego albo obu.

Stałe pole magnetyczne jest wywoływanie przez ładunki elektryczne znajdujące się w ruchu jednostajnym. Dlatego też przepływ prądu (który też jest ruchem ładunków elektrycznych) wytwarza pole magnetyczne. Ładunki poruszające się ruchem zmiennym (np. hamowane) powodują powstawanie zmiennego pola magnetycznego, które rozchodzi się jako fala elektromagnetyczna. Powstawanie pola magnetycznego na skutek przepływu prądu elektrycznego i innych ruchów ładunków elektrycznych opisuje prawo Biota-Savarta oraz prawo Ampèrea, które w postaci uogólnionej wchodzą w skład równań Maxwella.

Niektóre materiały magnetyczne, jak np. ferromagnetyki, wytwarzają stałe pole magnetyczne. Jest to spowodowane superpozycją orbitalnych momentów magnetycznych elektronów (w półklasycznym modelu Bohra przez orbitalny ruch obdarzonych ładunkiem elektrycznym elektronów wokół jądra). Zjawisko to jest dokładniej wyjaśnione w opisie magnetyzmu.

Pole magnetyczne jest też wytwarzane przez zmienne pole elektryczne. Z kolei zmienne pole magnetyczne wytwarza pole elektryczne. Takie wzajemnie indukowanie się pól zachodzi w fali elektromagnetycznej. Stałe w czasie pole magnetyczne nie wytwarza pola elektrycznego – wynika to wprost z równań Maxwella.

4. Cechy i zakresy fal elektromagnetycznych – zastosowanie w przesyłaniu sygnału.

fala elektromagnetyczna – rozchodzenie się zmiennych pól elektrycznych i magnetycznych w przestrzeni. Do fal elektromagnetycznych zalicza się: fale radiowe, mikrofale, podczerwień, światło widzialne, ultrafiolet, promieniowanie rentgenowskie i promieniowanie gamma. Podane fale różnią się między sobą długością i częstotliwością.

Wielkościami charakteryzującymi falę elektromagnetyczną są:

- częstotliwość to liczba pełnych zmian pola magnetycznego i elektrycznego zachodzących w ciągu sekundy (jedno zdarzenie cyklu w ciągu jednej sekundy). Wyrażamy ją w hercach,
- długość fali jest z kolei liczona jako odległość jaka dzieli dwa sąsiadujące ze sobą punkty, w których pole elektryczne i magnetyczne jest dokładnie takie samo.

Pomiędzy obiema tymi wielkościami istnieje stała zależność – im większa częstotliwość fali, tym mniejsza jest jej długość.

Fale radiowe i telewizyjne mają najmniejsze częstotliwości. Są wykorzystywane przede wszystkim w komunikacji. Dzięki ich istnieniu możliwe jest przekazywanie obrazu i dźwięku, co jest podstawą działania stacji radiowych i telewizyjnych. Ze względu na długość fale radiowe dzieli się na długie i krótkie. Na tzw. falach krótkich nadają rozgłośnie radiowe, które wykorzystują różne częstotliwości nadawania dla różnych miejsc w kraju. Są też stacje, które na obszarze całej Polski nadają na jednej częstotliwości – wtedy wykorzystywane są tzw. fale długie. Dzięki nim można odbierać programy stacji radiowych z innych krajów europejskich.

PODSTAWY ELEKTROTECHNIKI I ELEKTRONIKI

1. Porównaj budowę i zasadę działania diody półprzewodnikowej i diody Zenera.

Dioda półprzewodnikowa – element elektroniczny należący do rodziny diod, zawierający w swojej strukturze złącze „p-n” wykonane z materiałów półprzewodnikowych. Jest to nieliniowy, dwukoncowkowy element, którego wyprowadzenie przymocowane do warstwy „p” (+ positive) nazywane jest anodą, a do warstwy „n” (- negative) katodą. Jedną z głównych zalet diod jest prąd płynący tylko w jednym kierunku (od anody do katody) po spolaryzowaniu złącza „p-n” napięciem w kierunku przewodzenia (gdy napięcie na anodzie jest większe niż na katodzie). Natomiast po spolaryzowaniu złącza „p-n” napięciem w kierunku zaporowym (napięcie na anodzie jest mniejsze niż na katodzie), „ujemny” prąd nie zostanie przepuszczony przez diodę i przez to możemy nazwać ją zaworem elektrycznym, umożliwiającym przepływ prądu jedynie w jednym kierunku.

Dioda półprzewodnikowa zbudowana jest z dwóch półprzewodników: typu „p” i „n” domieszkowanych w różnym stopniu. Razem tworzą złącze „p-n”. Warstwa „n” z domieszkami donorów ma nadmiar elektronów, zaś warstwa „p” z domieszkami akceptorów ma ich niedobór (dziury). Po zetknięciu się tych warstw w diodach półprzewodnikowych rozkład elektronów ulega równomierному rozkładowi. Elektrony, których dotychczas „brakowało” w warstwie „p”, są przenoszone (dyfundują) do niej z warstwy „n”, zaś dziury przemieszczają się z warstwy „p” do „n”. Proces łączenia się elektronów z dziurami nazywany jest procesem rekombinacji.

Dioda Zenera zachowuje się dokładnie w taki sam sposób jak dioda ogólnego przeznaczenia. Po spolaryzowaniu diody Zenera w kierunku przewodzenia, w momencie, gdy anoda ma potencjał wyższy od katody, rozpoczyna się przewodzenie prądu. W odróżnieniu od normalnej diody, która doprowadza do blokady przepływu prądu w polaryzacji zaporowej, dioda Zenera przewodzi w kierunku zaporowym w momencie przekroczenia określonego progu napięcia. Zjawisko to ma miejsce, kiedy napięcie, które przyłożone jest do diody, przekracza wartość znamionową, co doprowadza do przebicia prądu Zenera w tak zwanej warstwie zubożonej. W tym momencie przez diodę płynie prąd, aby niemożliwy był dalszy wzrost napięcia. Zjawisko to jest złożone i różni się bez względu na parametry i sposób wykonania diody. W przypadku diody Zenera prąd przepływający przez diodę gwałtownie wzrasta aż do maksymalnej wartości danego obwodu. Określone jest to rezystorem zaporowym. Wsteczny prąd nasilenia osiągnięty raz pozostaje niezmienny w przypadku wachlarza napięć wstecznych. Moment, w którym dochodzi do ustabilizowania napięcia na diodzie, nazywa się napięciem Zenera. Jego wartość może wynosić od jednego volta do nawet setek woltów. Wśród różnic pomiędzy diodą Zenera a normalnymi diodami wymienić można:

- Małą odporność dynamiczną,
- przebicie, które gwarantuje nieprzebicie diody,
- wartość napięcia określoną dokładnie, z niewielką tolerancją,
- gwarancję możliwie gwałtownego przejścia do tak zwanego stanu przebicia złącza.

2. Porównaj budowę i zasadę działania tranzystora bipolarnego i tranzystora polowego.

Tranzystor bipolarny (dawniej: tranzystor warstwowy, tranzystor złączowy) – odmiana tranzystora, półprzewodnikowy element elektroniczny, mający zdolność wzmacniania sygnału. Zbudowany jest z trzech warstw półprzewodnika o różnym typie przewodnictwa. Charakteryzuje się tym, że niewielki

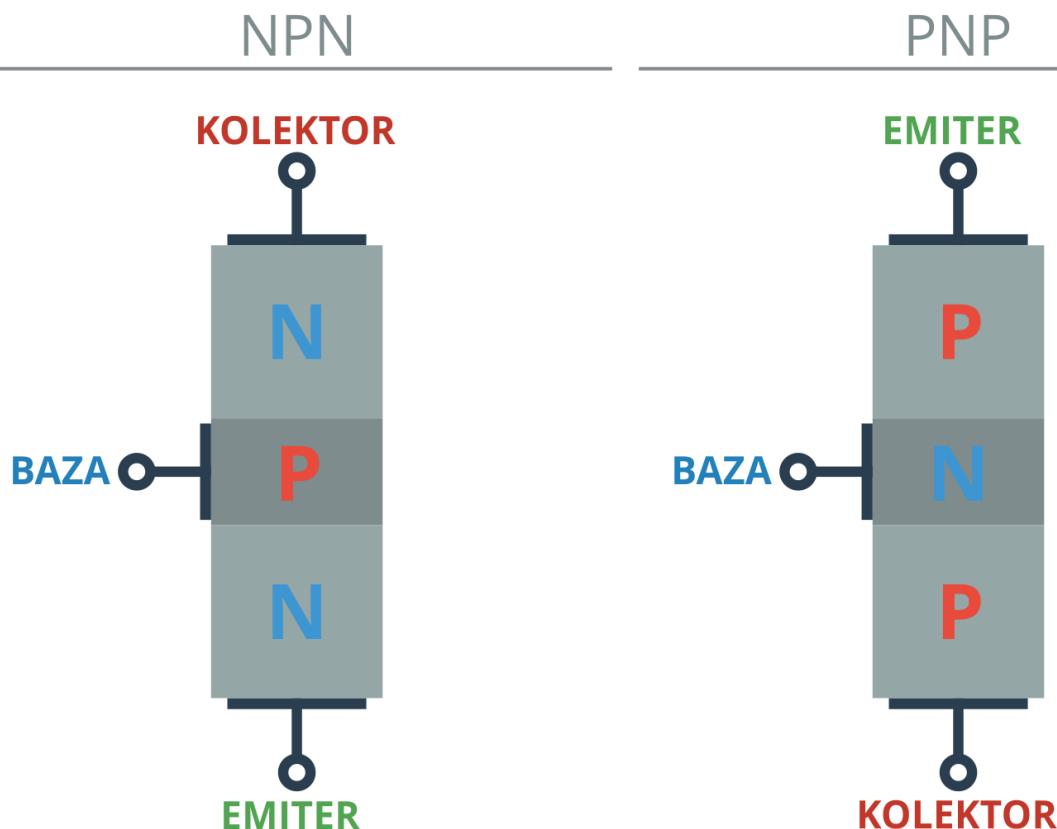
prąd płynący pomiędzy dwiema jego elektrodami (nazywanymi bazą i emiterem) steruje większym prądem płynącym między emiterem a trzecią elektrodą (nazywaną kolektorem).

„Tranzystor bipolarny to trzykońcowkowy element półprzewodnikowy, zdolny do wzmacniania prądu.”

Ale co to znaczy, że element (lub układ) wzmacnia sygnał? W żadnym wypadku nie należy tego zdania rozumieć w ten sposób, że w magiczny sposób nasz tranzystor „dokłada” energii, generując np. jeden amper prądu wyjściowego po zasilaniu go znacznie mniejszym prądem. Prawda jest inna: tranzystor „przepuszcza” większy prąd (z zasilacza) po wysterowaniu go prądem o znacznie niższej wartości. Dzięki temu możliwe jest sterowanie obciążeniem (np. elementem wykonawczym – przekaźnikiem, silnikiem, elektromagnesem, żarówką czy diodą LED) za pomocą niewielkiego prądu, dostarczanego np. przez port mikrokontrolera. Stosując pewne tricki układowe, można za pomocą tranzystora bipolarnego zbudować bardzo przyzwoity przedwzmacniacz audio, a nawet... solidną końcówkę mocy. Zanim jednak przejdziemy do opisu konkretnych zastosowań, poznajmy nieco bliżej naszego głównego bohatera – zarówno jego anatomicę, jak i umiejętności oraz... pewne wady, których – mimo rozwoju technologii półprzewodnikowych – nie udało się zlikwidować.

Budowa tranzystora bipolarnego

Tranzystory bipolarne zawierają w swojej obudowie płytke krzemową (dawniej germanową), której struktura została zmodyfikowana w procesie produkcji poprzez tzw. domieszkowanie – czyli taką zmianę mikrostruktury materiału bazowego, by wytworzyć trzy sąsiadujące ze sobą obszary, różniące się rodzajem nośników prądu. Jeżeli dwa obszary skrajne mają nadmiar elektronów (czyli ładunków ujemnych – ang. „negative”), a środkowy obszar ma nadmiar tzw. „dziur” (czyli ładunków dodatnich – ang. „positive”), mówimy o tranzystorze NPN. W przeciwnym przypadku mamy do czynienia z tranzystorem PNP, różniącym się od poprzednika tym, że wszystkie płynące przezeń prądy i występujące na końcówkach tranzystora napięcia, są skierowane dokładnie przeciwnie. Budowę tranzystora można zatem wyobrazić sobie jak – nieco skromną – kanapkę, w której pomiędzy dwiema kromkami chleba mamy cienki plaster szynki bądź sera (jak pokazano na schemacie).



Analogia taka, choć wydaje się być banalna, ma pewien sens – obszar środkowy w strukturze tranzystora bipolarnego faktycznie jest bardzo cienki, w porównaniu do dwóch obszarów z nim sąsiadujących. Środkowy obszar jest połączony z wyprowadzeniem tranzystora, zwany bazą – zaś dwa obszary okalające nazywamy emiterem oraz kolektorem. Pomiędzy poszczególnymi obszarami (kolektorem a bazą oraz emiterem i bazą) występują obszary przejściowe, zwane złączami P-N – nie różniącymi się zbytnio od złącz, znanych ze zwykłych, krzemowych diod prostowniczych. Dlatego też tranzystor można (w uproszczeniu) zamodelować za pomocą dwóch diod, połączonych ze sobą anodami (w przypadku tranzystora NPN) lub katodami (PNP). Możesz to łatwo sprawdzić, dokonując pomiaru napięcia przewodzenia pomiędzy odpowiednimi końcówkami tranzystora, za pomocą odpowiedniej funkcji (oznaczonej symbolem diody), dostępnej w prawie każdym multymetrem cyfrowym – dawniej z powodzeniem wykorzystywano w tym celu zwykły omomierz (patrz: schemat).

Zasada działania tranzystora bipolarnego

Model diodowy – jakkolwiek słuszny z punktu widzenia polaryzacji poszczególnych końcówek – nie wyjaśnia jednak, jak tranzystor działa. Choć w rzeczywistości działanie tranzystora (podobnie zresztą, jak w przypadku wszystkich innych elementów półprzewodnikowych) opiera się na dość skomplikowanych i trudnych zjawiskach tzw. fizyki ciała stałego, opisywanych przez zestawy wzorów i równań matematycznych, możemy pozwolić sobie na pewne uproszczenie. Przyjmijmy, że tranzystor może być zamknięty (wtedy praktycznie żaden prąd nie przepływa pomiędzy jego emiterem, a kolektorem), albo otwarty (prąd przepływa bez większych przeszkód). Do otwarcia tranzystora konieczne jest spolaryzowanie jego złącza baza-emiter, tzn. przyłożenie do niego napięcia, niezbędnego do – choćby częściowego – otwarcia tranzystora. Jak wiadomo, dla diody krzemowej napięcie przewodzenia (czyli napięcie występujące pomiędzy anodą, a katodą, gdy przez diodę przepływa prąd) wynosi od 0,5V do 0,8V. W tranzystorach bipolarnych nie jest inaczej – tutaj także pomiędzy bazą a emiterem, w stanie otwarcia tranzystora panuje napięcie około 0,6 V. Po właściwym spolaryzowaniu złącza baza-emiter, przez obwód kolektor-emiter może płynąć prąd o wartości wielokrotnie większej, niż prąd płynący przez złącze baza-emiter. Tranzystor bipolarny – symbol: Co ważne, symbol tranzystora bipolarnego czerpie niejako z modelu diodowego – strzałka, wskazująca emiter, ma kierunek zgodny z diodą złącza baza-emiter, dzięki czemu łatwo zapamiętać, który symbol oznacza którą wersję tranzystora.

Tranzystory polowe tak jak i tranzystory bipolarne są elementami półprzewodnikowymi lecz różnią się od bipolarnych tym, że są sterowane polem elektrycznym co oznacza, że nie pobierają mocy na wejściu. Pomimo takiej różnicy oba rodzaje tranzystorów mają wspólną cechę: są to elementy działające na zasadzie sterowania przepływem ładunku. W obu przypadkach są to elementy trzykońcowe, w których przewodność między dwoma końcówkami zależy od liczby nośników ładunków znajdujących się między nimi, a z kolei liczba nośników ładunków zależy od wartości napięcia doprowadzonego do elektrody sterującej zwanej bazą w tranzystorach bipolarnych lub bramką w tranzystorach polowych.

Działanie tranzystora polowego polega na sterowaniu przepływem prądu przez kanał za pomocą pola elektrycznego wytwarzanego przez napięcie doprowadzone do bramki. Ponieważ w tranzystorze polowym nie ma żadnych przewodzących złącz więc do bramki nie wpływa ani z niej nie wypływa żaden prąd i jest to chyba najważniejsza cecha tranzystorów polowych. Z właściwości tej wynika duża wartość rezystancji wejściowej tranzystora polowego co szczególnie w zastosowaniach takich jak przełączniki analogowe trudno jest przecenić.

Zasadniczą częścią tranzystora polowego jest kryształ odpowiednio domieszkowanego półprzewodnika z dwiema elektrodami: źródłem (symbol S, od ang. source, odpowiednik emitera w tranzystorze bipolarnym) i drenem (D, drain, odpowiednik kolektora). Pomiędzy nimi tworzy się tzw. kanał, którym płynie prąd. Wzdłuż kanału umieszczona jest trzecia elektroda, zwana bramką (G, gate, odpowiednik bazy). W tranzystorach epiplanarnych, jak również w przypadku układów scalonych, w

których wytwarza się wiele tranzystorów na wspólnym kryształe, wykorzystuje się jeszcze czwartą elektrodę, tzw. podłoże (B, bulk albo body), służącą do odpowiedniej polaryzacji podłoża.

Wewnętrzna budowa tranzystora to trudny temat, ale w skrócie można napisać, że w środku MOSFET-a znajdziemy:

- bramkę, oznaczoną jako G (ang. gate) – jest to metalizowana powłoka,
- izolator z tlenku krzemu – oddzielający bramkę od innych podzespołów,
- podłoże, oznaczone jako B (ang. bulk) – półprzewodnik domieszkowany przeciwnie do rodzaju kanału (za chwilę powiemy sobie, czym są kanały),
- dren, oznaczony jako D (ang. drain), oraz źródło, oznaczone jako S (ang. source) – to obszary o domieszkowaniu przeciwnym do podłoża, na którym się znajdują.

Zasada działania tranzystora MOSFET

Z uwagi na ich popularność poniżej omówione zostały tzw. tranzystory z kanałem wzbogaconym, w których kanał powstaje po przyłożeniu napięcia między bramką a źródłem. Drugi typ tranzystorów MOSFET, z tzw. kanałem zubożanym, nie będzie tutaj omawiany.

Przez wyłączony tranzystor w obwodzie dren-źródło nie płynie prąd. Dzieje się tak, ponieważ dren znajduje się na potencjale wyższym niż stykające się z nim podłoże (zwarte ze źródłem), co polaryzuje utworzone tam złącze p-n zaporowo. Innymi słowy, znajdująca się tam „niechcący” dioda zostaje zatkana. Ta dioda jest często umieszczana nawet na schematach ideowych, wewnątrz MOSFET-a.

Sytuacja zaczyna się zmieniać, jeżeli do bramki zaczniemy przykładać napięcie dodatnie względem źródła. Bramka jest odizolowana od pozostałych elementów tranzystora, dlatego nie płynie przez nią prąd. Naprzeciw bramki, po drugiej stronie izolatora, znajduje się podłoże zawierające dużo nośników dodatnich. Dodatni potencjał bramki odpycha je w głąb podłoża, przez co w obszarze pod bramką pozostają atomy domieszek i elektrony, przyciągnięte z całego podłoża.

Skąd się wzięły te elektrony? W półprzewodniku typu P znajdują się one w sposób naturalny (pochodzą od krzemu), ale ich liczebność jest wiele tysięcy razy mniejsza niż dziur. Kiedy warstwa elektronów robi się odpowiednio „gruba”, tworzy się kanał, przez który może płynąć prąd między drenem a źródłem. Kanał ten jest typu N, ponieważ tworzą go elektrony – stąd drugi człon nazwy.

3. Przedstaw zasady działania podstawowych układów elektronicznych: wzmacniacz, generatorów, filtry.

Wzmacniacz sygnałów elektrycznych – rodzaj wzmacniacza, którego zadaniem jest wytworzenie na wyjściu wzmacnionego wejściowego sygnału elektrycznego, kosztem energii pobieranej ze źródła zasilania. Wzmacniacze są budowane przy użyciu elementów aktywnych (niegdyś lamp elektronowych, obecnie tranzystorów).

Generator elektryczny – urządzenie przetwarzające na energię elektryczną inne rodzaje energii, w tym energię mechaniczną. Większość generatorów wytwarza energię elektryczną w wyniku indukcji elektromagnetycznej. Generatory te mają elementy poruszające się w polu magnetycznym lub wytwarzane jest zmienne pole magnetyczne. Zjawisko odwrotne, czyli przekształcenie energii elektrycznej w energię mechaniczną występuje w silnikach. Silniki i generatory wykorzystujące zjawisko indukcji elektromagnetycznej mają wiele podobieństw. Zalicza się je do tak zwanych maszyn elektrycznych. Część generatorów elektrycznych wytwarza energię elektryczną w wyniku innych zjawisk fizycznych.

Filtrem nazywamy fragment obwodu elektronicznego odpowiedzialny za przepuszczanie lub blokowanie sygnałów o określonym zakresie częstotliwości lub zawierającym określone harmoniczne.

W praktyce definicja ta oznacza, że filtrem elektrycznym można nazwać każdy układ, realizujący operacje na sygnale elektrycznym w dziedzinie częstotliwości. Filtry stosuje się najczęściej w celu usunięcia z sygnału niepożądanych składowych, zwykle zakłóceń oraz szumu.

4. Przedstaw typy i zasady: modulacji i demodulacji.

Link: <https://www.amt.pl/pl/modulacja-am-fm-i-pm>

<https://iviterserwis.pl/artykuly/jak-dziala-radio>

Modulacja analogowa AM, FM i PM

Po co moduluwać sygnał analogowy?

Jeśli chcesz przesłać sygnał audio na dużą odległość, okaże się, że próba przesłania sygnału w postaci oryginalnej będzie bardzo trudna. Sygnały audio nie mają odpowiednich właściwości, aby dotrzeć do oddalonego odbiornika. Możesz ustawić olbrzymią głośność, ale uda ci się tylko zirytować swoich sąsiadów. Z drugiej strony sygnały częstotliwości radiowej (RF) mają cenne właściwości propagacyjne. Możesz przesyłać te sygnały na duże odległości, przez przestrzeń kosmiczną, powietrze, lasy i budynki, a nawet wokół gór, i nadal można je prawidłowo odbierać. A co, jeśli mógłbyś spakować swój sygnał audio na sygnał RF? To jest cel modulacji - zabranie cennych informacji, które próbujesz przesłać, czy to głosu, czy muzyki, i przekazanie ich komuś znajdującemu się w dużej odległości przez nałożenie tych informacji na nośnik RF. Modulator łączy te sygnały do transmisji. Demodulator rozdziela dwa elementy z powrotem, dzięki czemu możesz odzyskać cenne informacje i pozbyć się sygnału RF.

Modulacja amplitudy

Modulacja AM utrzymuje stałą częstotliwość i fazę oraz skaluje amplitudę proporcjonalnie do sygnału audio. W najprostszym przypadku, gdy sygnał audio jest falą sinusoidalną, widok w domenie częstotliwości wygląda jak trzy tony - nośna plus dwa pasma boczne oddzielone szybkością modulacji. Modulacja niskiej częstotliwości pokazuje pasma boczne blisko nośnej, a modulacja wysokiej częstotliwości pokazuje pasma boczne dalej od nośnej.

Modulacja FM utrzymuje stałą amplitudę i zmienia częstotliwość fali w czasie, proporcjonalnie do sygnału audio. Jeśli sygnał audio jest falą sinusoidalną, wówczas w dziedzinie częstotliwości FM wygląda jak sygnał nośny plus pasma boczne, które pokrywają się z modulacją. Możesz to ustalić za pomocą funkcji Bessela. Pamiętaj, że może to wyglądać tak samo jak AM w dziedzinie częstotliwości.

Modulacja fazy

Modulacja PM utrzymuje również stałą amplitudę, ale przesuwa kształt fali o fazę. Wygląda tak samo jak FM w dziedzinie czasu. Czasami FM i PM nazywane są modulacją kąta, ponieważ są tak podobne w dziedzinie czasu. W dziedzinie częstotliwości PM wygląda jak sygnał nośny plus pasma boczne, które pokrywają się z modulacją. Podobnie jak w przypadku FM, można określić te pasy boczne za pomocą funkcji Bessela.

Demodulacja – proces odczytu informacji przekazywanych drogą radiową. Jest to proces odwrotny do modulacji. Podczas modulacji sygnał przekazywany jest za pomocą fali elektro-magnetycznej, najczęściej w postaci fali prostokątnych. W procesie demodulacji odpowiednie urządzenie mierzą amplitudę, szerokość i pozycję układu fali. Dzięki temu proces dekoduje informację zakodowaną podczas modulacji i w zależności od użytego układu przekształca ją na dane zrozumiałe dla człowieka, np. głos, albo obraz wideo.

Istnieje kilka sposobów demodulacji w zależności od tego, jak parametry sygnału pasma podstawowego, takie jak amplituda, częstotliwość lub faza, są przesyłane w sygnale nośnym. Na przykład dla sygnału modulowanego modulacją liniową, taką jak AM (modulacja amplitudy), możemy użyć detektora synchronicznego. Z drugiej strony, w przypadku sygnału modulowanego modulacją kątową, musimy zastosować demodulator FM (modulacja częstotliwości) lub demodulator PM (modulacja fazy). Funkcje te pełnią różne rodzaje obwodów. Wiele technik, takich jak odzyskiwanie nośnej, odzyskiwanie zegara, bit poślizgu, synchronizację ramki, odbiornik RAKE, kompresji impulsów, sygnał odbierany Strength Indication, wykrywania i korekcji błędów, itp, są wykonywane tylko przez demodulatory, chociaż jakiekolwiek konkretne demodulator może

wykonywać jedynie niektóre żadna z tych technik. Wiele rzeczy może działać jak demodulator, jeśli przepuszczają fale radiowe nieliniowo.

5. Podaj przykłady zastosowania przetworników A/C i C/A.

Przetwornik analogowo-cyfrowy A/C to układ, który służy do zamiany sygnały analogowego, inaczej ciągłego, na reprezentację cyfrową, tj. sygnał cyfrowy. Proces ten polega na takim uproszczeniu sygnału analogowego do postaci dyskretnej – skwantowanej, czyli zastąpieniu wartości zmieniających się płynnie, aby uzyskana wartość zmieniająca się skokowo była w odpowiedniej skali odwzorowana. Przetwarzanie dokonywane w przetworniku podzielone jest na etap próbkowania, kwantyzacji i kodowania. Analogowy sygnał ma charakter ciągły w czasie, dlatego dla przetworzenia go na sygnał cyfrowy konieczna jest jego zamiana na ciąg liczb.

Zastosowanie przetworników w praktyce

Można podać wiele przykładów zastosowań przetworników A/C lub C/A. Przetwornik będzie potrzebny choćby do wczytywania obrazu przez skaner do postaci bitmapy. Jednocześnie przetworniki C/A i A/C stosowane są w zestawach audio. Muzyka cyfrowa nie miałaby racji bytu bez tego elementu. Sprzęty grające, stacjonarne i przenośne, wykorzystują przetwornik cyfrowo-analogowy, służący do przetworzenia muzyki cyfrowej do postaci analogowej, słyszalnej z głośników. Przetworniki znajdziemy też w urządzeniach komputerowych.

6. Scharakteryzuj typy i przedstaw zasady działania bramek logicznych.

Bramki logiczne to narzędzia stanowiące elementy konstrukcyjne maszyn, automatów czy robotów. Komputery, z których obecnie korzystamy, wykorzystują miliony mechanizmów logicznych nazywanych bramkami cyfrowymi (lub inaczej bramkami logicznymi). Są to elementy elektroniczne, które przyjmują sygnały binarne na wejściach i zwracają wartości 1 lub 0 – prawda lub fałsz.

Bramki logiczne w układach cyfrowych

We wnętrzu komputera nieustannie odbywają się operacje, podczas których komputer korzysta właśnie z bramek logicznych – poziomy napięć są odpowiednikiem bitów. Bramka logiczna na swoim wyjściu zwraca odpowiednie napięcie elektryczne, które ponownie mieści się w określonym powyżej przedziale i oznacza jeden z dwóch stanów (1 – prawda lub 0 – fałsz). Wynika ono z funkcji logicznej realizowanej przez bramkę (czyli na przykład koniunkcja, negacja czy alternatywa). Bramki logiczne, które stosowane są w elektronice, to urządzenia wykorzystujące odpowiednio połączone elementy elektroniczne – są to przede wszystkim tranzystory, kondensatory, diody półprzewodnikowe i rezystory (oporniki). Dzisiejsze procesory składają się z miliardów bardzo niewielkich, odpowiednio połączonych tranzystorów. Są one kluczem do działania każdego układu cyfrowego. Każdy tranzystor posiada trzy elektrody – źródło, dren i bramkę. Po przyłożeniu napięcia do ostatniej z nich zmienia się przewodnictwo kanału. Taka znajomość działania tranzystorów jest wystarczająca do dobrego zrozumienia zasady działania bramek logicznych. Bramki logiczne można również postrzegać po prostu jako niewielkie czarne skrzynki, na których wejście podaje się napięcia, aby na wyjściu pojawiło się napięcie wynikowe zależne od funkcji (sygnału wejściowego). Podejście tego rodzaju jest w pełni wystarczające, aby można było zaprojektować nawet najbardziej złożone sieci cyfrowe.

Bramka NOT

Bramka NOT pod względem działania jest najprostsza ze wszystkich tu opisanych (i ogólnie wśród wszystkich wykorzystywanych bramek logicznych). Jej działanie polega na negacji (odwróceniu) sygnału, który otrzyma na wejściu. W praktyce wygląda to następująco: jeżeli na wejściu zostanie nadany sygnał o wartości 1 (prawda), to na wyjściu pojawi się sygnał o wartości 0 (fałsz). Natomiast jeżeli na wejściu pojawi się sygnał 0, na wyjściu pojawi się 1.

Przykład 1:

wejście = 1

wyjście = 0

Przykład 2

wejście = 0

wyjście = 1

Bramkę NOT przedstawiana jest również za pomocą symbolu graficznego – to trójkąt równoramienny, który można również postrzegać jako strzałkę zwróconą w prawo – w miejscu przecięcia się dłuższych ramion znajduje się okrąg styczny do jego wierzchołka (jego średnica to około $\frac{1}{4}$ długości krótszego boku trójkąta). Wejście na bramkę znajduje się po lewej stronie graficznego przedstawienia (prostopadle do krótkiego ramienia trójkąta), natomiast wyjście jest po prawej od symbolu (za opisanym kołem). Bramka posiada tylko jedno wejście i jedno wyjście. Przykładowy układ scalony (TTL), który zawiera 6 takich bramek (NOT) nosi nazwę “7404”.

Bramka AND

Przy działaniu bramki logicznej AND wynik 1 można otrzymać tylko w przypadku, kiedy oba wejścia będą równe 1. Tego typu bramki mogą występować w wersjach trzywejściowych, czterowejściowych oraz o znacznie większej liczbie wejść. Należy pamiętać, że niezależnie od tego, ile wejść będzie znajdowało się w stanie wysokim – stan wysoki na wyjściu będzie możliwy tylko w przypadku, jeżeli na każdym wejściu będzie znajdowała się logiczna jedynka.

Przykład 1:

wejście nr 1 = 1

wejście nr 2 = 0

wejście nr 3 = 1

wyjście = 0

Przykład 2:

wejście nr 1 = 1

wejście nr 2 = 1

wejście nr 3 = 1

wyjście = 1

Symbol logiczny bramki AND przypomina figurę zbudowaną z kwadratu oraz połówki koła (przeciętego wzduż średnicy, której długość jest równa długości boku kwadratu). Prawy bok kwadratu jest połączony z płaską częścią połowy koła. Całość przypomina strzałkę z wygiętym grotkiem, która jest zwrócona w prawą stronę. Po jej lewej stronie, na lewym boku kwadratu umieszczone są wejścia na bramkę, a wyjście analogicznie do symbolu bramki NOT jest po prawej stronie. Przykładowy układ scalony (TTL), który zawiera 4 takie bramki (bramki AND z dwoma wejściami) nosi nazwę “7408”.

Bramka NAND (-AND)

Działanie bramki NAND (-AND) jest dokładnie odwrotne do działania bramki AND. Alternatywną nazwą NAND jest określenie “-AND” – myślnik przed oznaczeniem AND oznacza negację. Tak też można interpretować działanie bramki NAND – jest ono dokładnie takie, jakby na wejściu bramki AND zostały umieszczone odpowiednie wartości, na jej wyjściu pojawił się wynik, a następnie został on przepuszczony przez jeszcze bramkę NOT. Można również interpretować działanie NAND w taki sposób, jakby przed każdym z wejść na bramkę AND była bramka NOT. W uproszczeniu działanie bramki NAND można opisać tak – stan niski (0) pojawia się jedynie wtedy, jeżeli na wszystkich wejściach pojawi się stan wysoki (1). Warto zaznaczyć, że bramka NAND posiada nieograniczoną liczbę wejść.

Przykład 1:

wejście nr 1 = 1

wejście nr 2 = 0

wejście nr 3 = 1

wyjście = 1

Przykład 2:

wejście nr 1 = 1

wejście nr 2 = 1

wejście nr 3 = 1

wyjście = 0

7. Porównaj: układ kombinacyjny i układ sekwencyjny.

Układ sekwencyjny - jednym z rodzajów układów cyfrowych charakteryzujący się tym, że stan wyjść "y" zależy od stanu wejść "x" oraz od poprzedniego stanu, zwanego stanem wewnętrznym, pamiętanego w zespole rejestrów.

Układ kombinacyjny - jeden z rodzajów układów cyfrowych charakteryzujący się tym, że stan wyjść zależy wyłącznie od stanu wejść.

Różnica:

Układ kombinacyjny charakteryzuje się tylko tym, że stan wyjść zależy WYŁĄCZNIE od stanu wejść, a układ sekwencyjny stan wyjść zależy od stanu wejść oraz od poprzedniego stanu, pamiętanego w zespole rejestrów.

8. Scharakteryzuj zasadę superpozycji w odniesieniu do obliczeń rozpływów prądów w obwodzie.

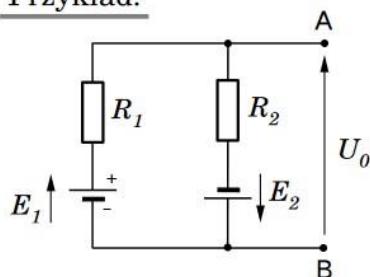
Zasada superpozycji – twierdzenie teorii obwodów, stosowane do rozwiązywania układów elektrycznych liniowych.

Metoda superpozycji

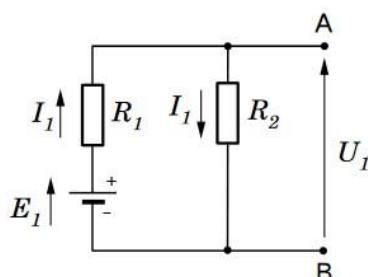
Metodę supopozycji możemy stosować dla układu liniowego zawierającego co najmniej dwa źródła.

Odpowiedź układu liniowego na kilka wymuszeń jest równa sumie odpowiedzi na każde wymuszenie oddzielnie.

Przykład:



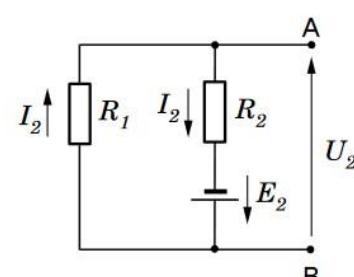
Układ z dwoma źródłami (wymuszeniami) E_1, E_2



Układ ze źródłem E_1

$$I_1 = \frac{E_1}{R_1 + R_2}$$

$$U_1 = R_2 I_1 = \frac{R_2}{R_1 + R_2} E_1$$



Układ ze źródłem E_2

$$I_2 = \frac{E_2}{R_1 + R_2}$$

$$U_2 = -R_1 I_2 = -\frac{R_1}{R_1 + R_2} E_2$$

$$U_0 = U_1 + U_2$$

Opierająca się na zasadzie superpozycji metoda, pomaga w wyznaczeniu prądu lub napięcia w wybranej gałęzi układu liniowego, zawierającego co najmniej dwa źródła niezależne. Może to być układ o strukturze zarówno szeregowo-równoległej jak i mostkowej. Prąd (napięcie) w danej gałęzi obwodu jest równy sumie prądów (napięć) wytworzonych w niej przez każde z niezależnych źródeł z osobna przy wyłączeniu pozostałych niezależnych źródeł, tj. zastąpieniu źródeł napięciowych zwarciami, a prądowych rozwarciami; źródła sterowane pozostają bez zmian. Umożliwia to analizę kilku podukładów, prostszych od układu wyjściowego.

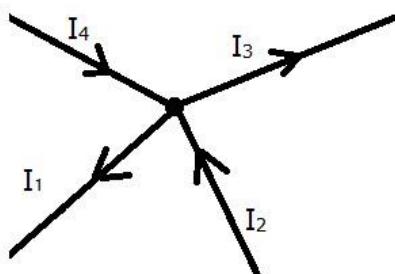
9. Przedstaw zastosowanie prawa Ohma i praw Kirchhoffa – w kontekście obliczania prądów i spadków napięć w obwodzie.

Pierwsze prawo Kirchhoffa traktuje o natężeniach w węźle: **suma natężen prądów wpływających jest równa sumie natężen prądów wypływających z węzła.**

Jeżeli przez I_{in} oznaczymy sumę natężen prądów wpływających, a przez I_{out} prądów wypływających, to:

$$I_{in} = I_{out}$$

Dla przykładu w poniższym obwodzie do węzła wpływają prądy o natężeniach I_2 oraz I_4 , a wypływają I_1 oraz I_3



Można zatem zapisać:

$$I_2 + I_4 = I_1 + I_3$$

lub w równoważnej postaci, jeżeli prądy wypływające uwzględniajemy ze znakiem minus: $-I_1 + I_2 - I_3 + I_4 = 0$

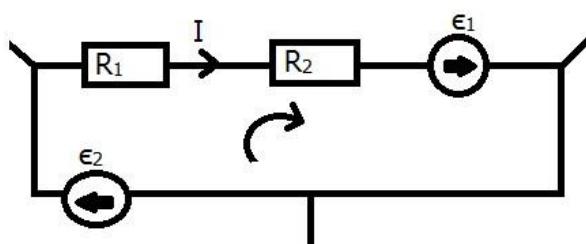
Drugie prawo Kirchhoffa traktuje o spadkach napięcia w oczku: **suma spadków napięć w oczku jest równa sumie sił elektromotorycznych.**

Lub w równoważnej postaci: **suma przyrostów i spadków napięć w oczku jest równa zero.**

Korzystając z tego prawa należy określić kierunek przehodzenia oczka (strzałka wewnątrz), jeżeli siły SEM działają w tym kierunku należy uwzględnić je ze znakiem plus, jeżeli w przeciwnym, ze znakiem minus. Napięcia na opornikach należy uwzględnić ze znakiem minus, gdy kierunek przepływu prądu jest zgodny z kierunkiem przehodzenia oczka, a ze znakiem plus, jeżeli jest przeciwny.

Dla przykładu w obwodzie poniżej ustalono kierunek przehodzenia oczka zgodny z ruchem wskazówek zegara. Siła elektromotoryczna ϵ_1 oraz ϵ_2 działają zgodnie z tym kierunkiem, będą zatem uwzględnione ze znakiem plus. Przez oporniki prąd przepływa zgodnie z kierunkiem przehodzenia, więc spadki napięć będą uwzględnione ze znakiem minus. Ostatecznie:

$$\epsilon_1 + \epsilon_2 - IR_1 - IR_2 = 0$$



Oporniki są tutaj połączone szeregowo, więc natężenie prądu przepływającego jest takie samo, wynosi I , a napięcie na opornikach $U_1 = I \cdot R_1$ oraz $U_2 = I \cdot R_2$ zgodnie z **prawem Ohma**.

Prawo: prawo Ohma

Natężenie prądu płynącego przez przewodnik jest wprost proporcjonalne do napięcia przyłożonego między jego końcami.

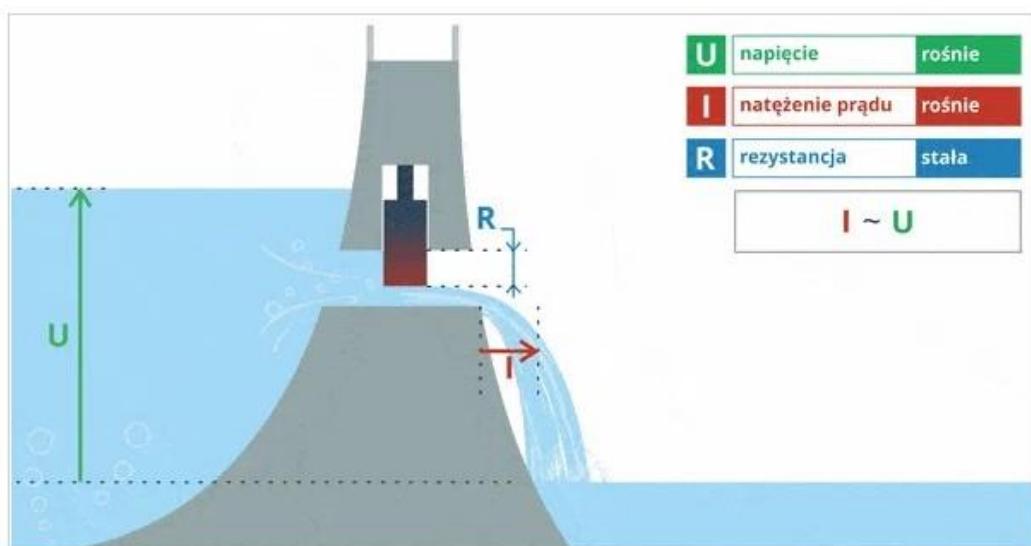
Tę zależność zapisujemy w postaci wzoru:

$$I = \frac{U}{R}$$

Interpretacja prawa Ohma

Zanim przejdziemy do przykładu, zastanówmy się, jak można zrozumieć te wzory. W odniesieniu do prądu prawo Ohma mówi, że jest on wprost proporcjonalny do przyłożonego napięcia. Przykładowo: jeśli zwiększymy napięcie 10 razy, to prąd również wzrośnie 10 razy. Wynika to jasno ze wzoru $I = U / R$ (im wyższe będzie napięcie podstawione do wzoru, tym większy będzie prąd).

Pamiętasz naszą wodną analogię z początku kursu? Tam jest to doskonale widoczne: gdy poziom wody podniesie się (napięcie wzrośnie), to z tamy wypłynie więcej wody (wzrośnie prąd).



Analogia wodna: przy stałym oporze zwiększenie napięcia prowadzi do wzrostu prądu

Jak widać, mamy tutaj **zwiększające się napięcie** (poziom wody) przy **stałym oporze** (położenie śluzy w tamie). Wniosek: wzrost napięcia prowadzi do przepływu większego prądu przy tym samym oporze.

10. Podaj przykład wykorzystania twierdzenia o mocy maksymalnej w obwodzie.

Теорема о максимуме отдаваемой мощности

Теорема о максимуме отдаваемой мощности является не столько средством анализа, сколько помощью в разработке схем. Согласно этой теореме максимум мощности будет передаваться от источника в нагрузку только в том случае, когда сопротивление нагрузки будет равно выходному сопротивлению источника. Если сопротивление нагрузки больше или меньше сопротивления источника, то отдаваемая мощность будет меньше максимальной.

Данную теорему, по сути дела, можно применить при проектировании радиопередатчиков, где "импеданс" антенны должен быть согласован с "импедансом" оконечного усилителя мощности (для получения максимальной выходной мощности радиочастоты). Импеданс - это комплексное сопротивление переменному и постоянному токам, он очень похож на обычное сопротивление и должен быть равным между источником и нагрузкой, в целях передачи в нагрузку максимальной мощности. Если импеданс нагрузки слишком высок, то это приведет к снижению выходной мощности. Если импеданс нагрузки слишком низок, то это приведет не только к снижению выходной мощности, но и к перегреву усилителя из-за рассеивания мощности на его внутреннем сопротивлении.

Ограничения теоремы о максимуме отдаваемой мощности.

Максимальная отдаваемая мощность не "совпадает" с максимальной "производительностью". Применение данной теоремы к распределению мощности переменного тока не приведет к получению максимального или даже высокого КПД. При распределении мощности переменного тока более важной является высокая производительность, которая достигается относительно низким импедансом генератора по сравнению с импедансом нагрузки.

К примеру, высокое качество звука можно получить при относительно низком выходном импедансе схемы и относительно высоком импедансе динамика (нагрузки). Отношение "выходного импеданса" к "импедансу нагрузки" известно как коэффициент затухания, обычно он находится в диапазоне от 100 до 1000.

Максимальная отдаваемая мощность не совпадает и с целью получения низкого уровня шума. Например, для минимизации шумов между антенной и усилителем радиочастоты приемника. Данная цель так же требует несоответствия импедансов усилителя и антенны.

ALGORYTMY I STRUKTURY DANYCH

1. Czym powinien cechować się dobry algorytm.

Podstawowymi cechami algorytmów są:

- poprawność — algorytm powinien zwracać poprawne wyniki
- jednoznaczność — algorytm powinien przy takim samym zbiorze danych wejściowych zwracać takie same wyniki
- skończoność dla każdego zbioru poprawnych danych wejściowych algorytm powinien zwracać wyniki w skończonej liczbie kroków
- efektywność algorytm powinien prowadzić do rozwiązania problemu w jak najmniejszej liczbie kroków.

2. Scharakteryzuj podstawowe struktury algorytmów (algorytmy liniowe, algorytmy z rozgałęzieniami, algorytmy z powtórzeniami).

Algorytm liniowy to algorytm mający prostą postać. Składa się z ciągu instrukcji, które są wykonywane jedna po drugiej w kolejności, jaka wynika z ich następstwa w zapisie algorytmu. Taki algorytm nosi również nazwę algorytmu sekwencyjnego .

Oto jak wygląda algorytm (wersja 1):

- Podnieś słuchawkę.
- Wybierz cyfrę 6.
- Wybierz cyfrę 1.
- Wybierz cyfrę 6.
- Wybierz cyfrę 2.
- Wybierz cyfrę 2.
- Wybierz cyfrę 2.
- Wybierz cyfrę 2.
- Przekaż informacje.
- Odłóż słuchawkę.

Kolejny rodzaj algorytmów to algorytmy warunkowe, zwane inaczej algorytmami rozgałęzionymi.

W rzeczywistości, większość algorytmów nie jest liniowa, ma bardziej rozbudowaną strukturę. Często występują w nich instrukcje, których wykonanie uzależnione jest od spełnienia pewnego warunku lub też spełnienie pewnego warunku powoduje wykonanie jednej instrukcji, a niespełnienie go innej. Taką instrukcję nazywamy instrukcją warunkową.

Aby zrealizować taką sytuację zastosujemy instrukcję warunkową. Zrobimy to po to, aby opisać czynności, powinny być wykonane wtedy kiedy zostało nawiązane poprawne połączenie, jak również nie zostało nawiązane. Zauważ, że wtedy wykonawca znajdzie się w punkcie wyjścia, czyli jakby w ogóle nie podjął próby telefonowania.

Algorytm może mieć postać taką (wersja 2):

1. Podnieś słuchawkę.
2. Wybierz cyfrę 6.
3. Wybierz cyfrę 1.
4. Wybierz cyfrę 6.
5. Wybierz cyfrę 2.
6. Wybierz cyfrę 2.
7. Wybierz cyfrę 2.
8. Wybierz cyfrę 2.
9. Czy połączyleś się z koleżanką ?
Jeśli TAK, to przejdź do kroku 10. Jeśli NIE, to przejdź do kroku
10. Zaproś koleżankę.

11. Odłóż słuchawkę.

Instrukcja iteracyjna – ze znaną ilością powtórzeń

Przyjrzyj się uważnie algorytmowi przedstawionemu wcześniej, powyżej. Zauważysz, że istnieją tu powtarzające się instrukcje, aż czterokrotnie występuje „Wybierz cyfrę 2”.

Takie wielokrotne powtarzanie niektórych instrukcji jest cechą charakterystyczną wielu algorytmów, jednak nie zawsze (tak jak w tym algorytmie) możemy określić dokładnie liczbę powtórzeń. Może ona zależeć od spełnienia pewnych warunków. Wielokrotne powtarzanie instrukcji umożliwiają instrukcje iteracyjne (pętle). Działa ona według schematu: Wykonuj instrukcję A dokładnie n razy.

Oto algorytm (wersja 3):

1. Podnieś słuchawkę.
2. Wybierz cyfrę 6.
3. Wybierz cyfrę 1.
4. Wybierz cyfrę 6.
5. Wykonaj czynność cztery razy: Wybierz cyfrę 2.
6. Czy połączysz się z koleżanką? Jeśli tak, to przejdź do kroku 7. Jeśli nie, to przejdź do kroku 8.
7. Zaproś koleżankę.
8. Odłóż słuchawkę.

3. Podaj przykład oszacowania złożoności obliczeniowej algorytmu.

Ilość zasobów potrzebnych dla działania algorytmu może być rozumiana jako jego złożoność. W zależności od rozważanego zasobu mówimy o złożoności czasowej lub pamięciowej. Oczywiście w większości wypadków ilość potrzebnych zasobów będzie się różnić w zależności od instancji problemu.

Jako przykład może nam posłużyć problem rozkładu liczb na czynniki pierwsze. Domyślamy się, że (niezależnie od algorytmu) im większa liczba, tym więcej zasobów będzie potrzebnych do jej rozłożenia. Taką własność ma znakomita większość problemów obliczeniowych: im większy rozmiar danych wejściowych tym więcej zasobów (czasu, procesorów, pamięci) jest potrzebnych do jego rozwiązania. Złożoność algorytmu jest więc funkcją rozmiaru danych wejściowych.

Kolejnym problemem jest fakt, iż złożoność nie zależy tylko i wyłącznie od rozmiaru danych, ale może się znacznie różnić dla instancji o identycznym rozmiarze. Dwa często spotykane sposoby radzenia sobie z tym problemem to: branie pod uwagę przypadków najgorszych (złożoność pesymistyczna) i pewien sposób uśrednienia wszystkich możliwych przypadków (złożoność oczekiwana).

Złożoność czasowa

Przyjętą miarą złożoności czasowej jest liczba operacji podstawowych w zależności od rozmiaru wejścia. Pomiar rzeczywistego czasu zegarowego jest mało użyteczny ze względu na silną zależność od implementacji algorytmu, użytego kompilatora, maszyny na której algorytm wykonano, a także umiejętności programisty. Dlatego w charakterze czasu wykonania rozpatrujemy zwykle liczbę operacji podstawowych (dominujących). Operacjami podstawowymi mogą być na przykład: podstawienie, porównanie lub prosta operacja arytmetyczna.

Złożoność pamięciowa

Podobnie jak złożoność czasowa jest miarą czasu działania algorytmu, tak złożoność pamięciowa jest miarą ilości wykorzystanej pamięci. Jako tę ilość najczęściej przyjmuje się użytkę pamięci maszyny abstrakcyjnej (na przykład liczbę komórek pamięci maszyny RAM) w funkcji rozmiaru wejścia. Możliwe jest również obliczanie rozmiaru potrzebnej pamięci fizycznej wyrażonej w bitach lub bajtach.

4. Metody projektowania algorytmów (rekurencja, „dziel i zwycięzaj”, metoda zachłanna, programowanie dynamiczne, algorytmy z powrotami) – działanie, zastosowania.

Rekurencja jest to sytuacja, w której funkcja (czy też w naszym przypadku metoda) wywołuje samą siebie w celu rozwiązania pewnego problemu - stąd m.in. słynne już powiedzenie żeby zrozumieć rekurencję, trzeba zrozumieć rekurencję. W wielu sytuacjach pozwala w znaczny sposób uprościć zapis iteracyjnego rozwiązania naszego problemu (na przykład skomplikowanej pętli) - jednak jak się później dowiesz, nie zawsze jest to dobry pomysł. W tym momencie może Ci się także wydawać, że metoda wywołując samą siebie doprowadzi do sytuacji typu nieskończona pętla - i tak może się zdarzyć, jednak dzięki odpowiedniemu zapisowi można się przed tym zabezpieczyć.

Rekurencja wykorzystywana jest w wielu sytuacjach, jednak najpopularniejsze to:

- algorytmy wyszukiwania (na przykład quick sort)
- rekurencyjne struktury danych (na przykład lista wiązana)
- specyficzne algorytmy, w których wykorzystanie rekurencji jest czymś naturalnym

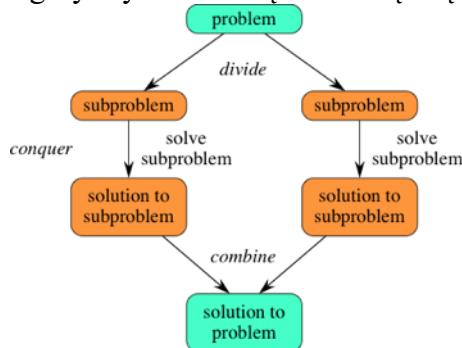
Dziel-i-rządź

Zarówno sortowanie przez scalanie, jak i szybkie sortowanie wykorzystują wspólny paradygmat algorytmiczny, oparty na rekurencji. Ten paradygmat, dziel-i-rządź, dzieli problem na podproblemy, które są podobne do pierwotnego problemu, rekurencyjnie rozwiązuje podproblemy, a następnie łączy ze sobą rozwiązania tych podproblemów, aby rozwiązać pierwotny problem. Ze względu na to, że dziel-i-rządź rozwiązuje podproblemy rekurencyjnie, każdy podproblem musi być mniejszy niż

problem pierwotny i musi istnieć przypadek bazowy dla podproblemów. Pomyślmy o paradygmacie dziel-i-rządź jako o algorytmie złożonym z trzech części:

1. Podziel problem na pewną liczbę podproblemów, które są mniejszymi przypadkami tego samego problemu.
2. Rządź podproblemami, rozwiązuając je rekurencyjnie. Jeśli są wystarczająco małe, rozwiąż podproblemy jako przypadki bazowe.
3. Połącz rozwiązania poszczególnych podproblemów w jedno rozwiązanie pierwotnego problemu.

Z łatwością możemy zapamiętać kroki algorytmu dziel-i-rządź jako podziel, rządź, połącz. Poniższy obrazek pokazuje nam jeden krok algorytmu, przy założeniu, że operacja dzielenia problemu tworzy dwa podproblemy (choć niektóre algorytmy dziel-i-rządź tworzą więcej niż dwa podproblemy):



Algorytmy zachłanne (ang. greedy algorithms) – algorytmy podejmujące w każdym kroku taką decyzję, która w danej chwili wydaje się najkorzystniejsza. Inaczej mówiąc, algorytmy zachłanne dokonują zawsze wyborów lokalnie optymalnych licząc, że doprowadzi to do znalezienia rozwiązania globalnie optymalnego. W ogólnym przypadku algorytmy zachłanne nie zawsze znajdują rozwiązanie optymalne. Są one zatem podzioborem algorytmów heurystycznych. Jednocześnie są to algorytmy deterministyczne – nie ma w nich losowości.

Bardzo prostym przykładem algorytmu zachłanego może być szukanie najwyższego punktu na określonym obszarze poprzez przesuwanie się zawsze w kierunku największego nachylenia (nigdy się nie cofając ani nie rozpatrując kilku wariantów drogi). Jak widać, w ten sposób prawdopodobnie dojdziemy do wierzchołka położonego najbliżej od punktu początkowego, który niekoniecznie będzie najwyższym.

Programowanie dynamiczne

Rozwiązaniem danego problemu często jest kombinacja rozwiązań pod problemów, na które można problem rozłożyć. Natomiast nie od razu wiemy, jaka dekompozycja jest optymalna; początkowo mamy niedeterministyczny wybór wielu różnych dekompozycji. W sytuacji, gdy nie wiemy, jaka dekompozycja jest optymalna, nie możemy uruchomić rekursji, ponieważ na każdym etapie mielibyśmy wiele wyborów i w sumie złożoność mogłaby być wykładnicza.

W takich sytuacjach stosujemy metodę zwaną programowaniem dynamicznym. Metoda ta, z grubsza biorąc, wygląda następująco: Jeśli problem możemy rozbić na pod problemy i liczba wszystkich potencjalnych pod problemów jest wielomianowa, to zamiast korzystać z rekursji możemy obliczyć wartości wszystkich pod problemów stosując odpowiednią kolejność: od "mniejszych" pod problemów do "większych". Rozmiary problemów muszą być odpowiednio zdefiniowane, nie powinno być zależności cyklicznej.

Wartości obliczone dla pod problemów zapamiętujemy w tablicy. Mając obliczone wartości pod problemów, na które można rozbić dany problem, wartość problemu obliczamy korzystając z wartości zapamiętanych w tablicy.

Najistotniejsze jest tutaj określenie zbioru potencjalnych pod problemów. Z reguły zbiór ten jest znacznie większy niż zbiór pod problemów będących częściami jednego optymalnego rozwiązania.

Można wykonywać w dowolnym przynależnym porządku ze względu na (i,j) (na przykład po przekątnych, zaczynając od głównej przekątnej). Przynależność polega na tym, że jest już ostatecznie policzone to, z czego w danym momencie korzystamy.

Algorytm ma złożoność czasową $O(n^3)$ i jest to "typowa" złożoność algorytmów tego typu. Duża złożoność wynika stąd, że liczymy wartości dla mnóstwa pod problemów, które mogą być zupełnie nieistotne z punktu widzenia optymalnego rozwiązania.

Algorytm z powrotami

Link: http://nqueens.fatmagnus.ppa.pl/algor_zasada.php

5. Porównaj podstawowe struktury danych (tablice, listy, kolejki, stosy, mapy, grafy, drzewa) – cechy, zastosowania.

Link: <https://stormit.pl/struktury-danych/>

https://eduinf.waw.pl/inf/alg/001_search/0123.php

https://eduinf.waw.pl/inf/alg/001_search/0108.php

ARCHITEKTURA KOMPUTERÓW

1. Porównaj konstrukcję modelu programowego procesora w podejściu CISC i RISC.

Termin RISC oznacza "komputer z ograniczonym zestawem instrukcji". Jest to strategia projektowania procesora oparta na prostych instrukcjach i szybkiej wydajności.

RISC to mały lub ograniczony zestaw instrukcji. Tutaj każda instrukcja ma na celu osiągnięcie bardzo małych zadań. W maszynie RISC zestawy instrukcji są proste i proste, co pomaga w komponowaniu bardziej złożonych instrukcji. Każda instrukcja ma tę samą długość; instrukcje są nawiązywane razem, aby uzyskać złożone zadania wykonane w ramach jednej operacji. Większość instrukcji jest wykonywana w jednym cyklu maszynowym. To pipelining jest kluczową techniką używaną do przyspieszenia maszyn RISC.

RISC to mikroprocesor zaprojektowany do wykonywania kilku instrukcji w tym samym czasie. W oparciu o małe instrukcje, te układy wymagają mniej tranzystorów, co sprawia, że tranzystory są tańsze w projektowaniu i wytwarzaniu. Niektóre inne funkcje RISC obejmują:

- Mniejsze zapotrzebowanie na dekodowanie
- Jednolity zestaw instrukcji
- Identyczny rejestr ogólnego przeznaczenia
- Proste węzły adresujące
- Niewiele typów danych w sprzęcie

Ponadto, podczas pisania kodów, RISC ułatwia to, pozwalając programistom na usuwanie niepotrzebnych kodów i zapobiega marnowaniu cykli.

Termin CISC oznacza "Complex Instruction Set Computer". Jest to strategia projektowania procesora oparta na pojedynczych instrukcjach, które mogą wykonywać operacje wieloetapowe.

Komputery CISC mają zwarte programy. Ma wiele skomplikowanych instrukcji, których wykonanie zajmuje dużo czasu. W tym przypadku pojedynczy zestaw instrukcji obejmuje wiele etapów; każdy zestaw instrukcji ma ponad trzysta oddzielnych instrukcji. Większość instrukcji jest wykonywana w dwóch do dziesięciu cyklach maszynowych. W CISC nie można łatwo wdrożyć potokowania instrukcji.

Maszyny CISC mają dobre wyniki, oparte na uproszczeniu kompilatorów programów; ponieważ zakres zaawansowanych instrukcji jest łatwo dostępny w jednym zestawie instrukcji. Projektują złożone instrukcje za pomocą jednego prostego zestawu instrukcji. Wykonują operacje niskiego poziomu, takie jak operacja arytmetyczna lub ładowanie z pamięci i magazynu pamięci. CISC ułatwia posiadanie dużych węzłów adresujących i więcej typów danych w sprzęcie urządzenia. CISC uważa się jednak za mniej efektywny niż RISC, ponieważ nieefektywność usuwa kody, które prowadzą do marnowania cykli. Ponadto mikroprocesorowe układy scalone są trudne do zrozumienia i programowania, ze względu na złożoność sprzętu.

	RISC	CISC
Akronim	Jest to skrót od "Komputer ze zredukowanym zestawem instrukcji".	Jest to skrót od "Complex Instruction Set Computer".
Definicja	Procesory RISC mają mniejszy zestaw instrukcji z kilkoma węzłami adresującymi.	Procesory CISC mają większy zestaw instrukcji z wieloma adresującymi węzłami.
jednostka pamięci	Nie ma jednostki pamięci i używa oddzielnego sprzętu do implementacji instrukcji.	Posiada jednostkę pamięci do realizacji złożonych instrukcji.
Program	Ma twardą jednostkę programowania.	Ma jednostkę mikro-programowania.
Projekt	Jest to złożony projekt kompilatora.	Jest to prosty kompilator.
Obliczenia	Obliczenia są szybsze i bardziej precyzyjne.	Obliczenia są powolne i precyzyjne.
Rozszyfrowanie	Dekodowanie instrukcji jest proste.	Dekodowanie instrukcji jest złożone.
Czas	Czas realizacji jest znacznie mniejszy.	Czas realizacji jest bardzo wysoki.
Pamięć zewnętrzna	Nie wymaga pamięci zewnętrznej do obliczeń.	Do obliczeń wymaga pamięci zewnętrznej.
Rurociągi	Pipelining działa poprawnie.	Pipelining nie działa poprawnie.
Utrapienie	Utylizacja jest najczęściej zmniejszona w procesorach.	Procesory często zwlekają.
Rozszerzenie kodu	Rozszerzanie kodu może być problemem.	Rozszerzenie kodu nie stanowi problemu.
Miejsce na dysku	Przestrzeń jest zapisana.	Przestrzeń jest zmarnowana.
Aplikacje	Używany w zaawansowanych aplikacjach, takich jak przetwarzanie wideo, telekomunikacja i przetwarzanie obrazu.	Używane w aplikacjach o niskim poziomie, takich jak systemy bezpieczeństwa, automatyzacja domu itp.

2. Omów podstawy realizacji systemu pamięci podrzcznej uwzględniając jej poziomowość.

Pamięć podrzczna procesora (ang. CPU cache) – jest pamięcią typu SRAM (pamięć statyczna) o krótkim czasie dostępu. Zlokalizowana jest często bezpośrednio w jądrze procesora. Zastosowanie wielopoziomowej hierarchii pamięci podrzcznej zmniejsza średni czas dostępu do pamięci głównej. Współcześnie stosuje się 2 i 3-poziomowe pamięci podrzczne. Najważniejszymi parametrami funkcjonalnymi pamięci podrzcznych są pojemność i czas dostępu.

L-1 cache

Zlokalizowana we wnętrzu procesora pamięć podrzczna pierwszego poziomu przyspiesza dostęp do bloków pamięci wyższego poziomu, który stanowi zależnie od konstrukcji pamięć operacyjną lub pamięć podrzczną drugiego poziomu (L-2). Z uwagi na ograniczenia rozmiarów i mocy procesora zawsze jest najmniejsza. Umieszczona jest najbliżej głównego jądra procesora i umożliwia najszybszą komunikację procesora. Typowe pamięci L-1 współczesnych procesorów są 2-drożne, posiadającą rozzieloną pamięć danych i kodu, a długość linii wynosi 64 bajty.

L-2 cache

Pamięć drugiego poziomu, o rozmiarze od 64 KB do 12 MB, 2, 4 lub 8-drożna, o długości linii od 64 do 128 bajtów, jest wykorzystywana jako bufor pomiędzy stosunkowo wolną pamięcią RAM a jądrem procesora i pamięcią cache L1. Ostatnimi procesorami nie posiadającymi pamięci podręcznej drugiego poziomu były pierwsze procesory Celeron (jądro Convington, taktowane 266–300 MHz). Obecność pamięci drugiego poziomu powoduje duży wzrost wydajności w wielu aplikacjach – dzieje się tak, ponieważ dane poddawane obróbce muszą być pobierane z pamięci RAM, która ma opóźnienia rzędu kilkudziesięciu-kilkuset nanosekund. Dzisiejsze procesory są wyposażone w złożone układy przewidywania, jakie dane będą potrzebne – dane te są pobierane z wyprzedzeniem do pamięci cache, która ma opóźnienia rzędu kilku-kilkunastu nanosekund, co znacznie skraca czas oczekiwania procesora na dane do obliczeń.

L-3 cache

Pamięć podręczna procesora trzeciego poziomu jest wykorzystywana, gdy pamięć L2 jest niewystarczająca, aby pomieścić potrzebne dane. Obecność cache trzeciego poziomu pozwala na znaczącą poprawę wydajności w stosunku do procesorów o konstrukcji pamięci dwupoziomowej w wielu aplikacjach i programach. W systemach z wieloma procesorami lub rdzeniami, pamięć cache trzeciego poziomu najczęściej jest współdzielona przez wszystkie rdzenie i ma od kilku do kilkudziesięciu megabajtów.

3. Zdefiniuj budowę modelu programowego jednostki centralnej – omów niezbędne rejestrty, tryby adresowania, listę instrukcji oraz model operacji warunkowych.

Link:

http://smurf.mimuw.edu.pl/external_slides/W4_Struktura_modelu_programowego/Architektura_Komputerow_Wyklad_4_Struktura_modelu.html

4. Omów architekturę przykładowego mikrokontrolera. Czym różni się mikrokontroler od mikroprocesora?

<https://pl.gadget-info.com/difference-between-microprocessor>

Pojęcie mikrokontroler oznacza układ scalony z wyspecjalizowanym mikroprocesorem, który spełnia dwa podstawowe kryteria:

- Jest zdolny do autonomicznej pracy, tzn. w najprostszach zastosowaniach nie wymaga przyłączenia zewnętrznych układów pomocniczych, np. pamięci;
- Został zaprojektowany z myślą o pracy w systemach kontrolno-pomiarowych, ma zatem rozbudowany system komunikacji z otoczeniem, zarówno przy użyciu sygnałów cyfrowych, jak i analogowych.

Standardowy mikroprocesor jest złożony z jednostki centralnej i szczegółowej pamięci danych która przyjmuje postać zbioru rejestrów. Większość mikroprocesorów posiada także generator zegara. Mikroprocesory natomiast nie mają wbudowanej pamięci programu i układów wejścia/wyjścia ani układów peryferyjnych.

Mikrokontroler może być zatem uważany na pierwszy rzut oka na mikroprocesor który posiada dodatkowo wbudowane układy pamięci i układy do komunikacji z otoczeniem. Ważna różnica dotyczy ponadto zbioru operacji realizowanych przez CPU, nazywanego listą instrukcji. W przypadku mikroprocesorów lista instrukcji jest zaprojektowana pod kątem ułatwienia obliczeń numerycznych i transmisji danych, natomiast mikrokontrolery mają rozbudowane grupy instrukcji do obsługi urządzeń peryferyjnych.

5. Przedstaw klasyfikację układów programowalnych. Krótko scharakteryzuj każdą z klas.

Linki:

- <http://www.eurobajt.pl/programowalne-uklady-cyfrowe-co-warto-o-nich-wiedziec/>
- [https://education.fandom.com/wiki/PWr - Programowalne uk%C5%82ady logiczne \(klasyfikacja, zasoby, technologia programowania\)](https://education.fandom.com/wiki/PWr - Programowalne uk%C5%82ady logiczne (klasyfikacja, zasoby, technologia programowania))
- <http://rspn.univ.rzeszow.pl/wp-content/uploads/2015/01/Programowalne-uk%C5%82ady-logiczne.pdf>

6. Na czym polega minimalizacja funkcji logicznych.

Linki: https://eduinf.waw.pl/inf/alg/002_struct/0005.php

Minimalizacja funkcji logicznych polega na upraszczaniu wyrażeń logicznych, tak aby zawierały jak najmniejszą liczbę tylko niezbędnych argumentów oraz operacji logicznych. Zadanie to ma podstawowe znaczenie dla techniki cyfrowej, gdzie funkcje urządzenia realizowane są za pomocą tzw. bramek logicznych (układów wykonujących podstawowe operacje logiczne). Jest chyba jasne, iż prostsze wyrażenie oznacza mniejszą złożoność układu elektronicznego, a co za tym idzie mniejszy koszt wykonania (szczególnie w dużych seriach i nie na potrzeby wojska), większą niezawodność (mniej elementów, które mogą przestać działać) oraz zwykle szybsze działanie (mniej uzależnień powoduje szybszą stabilizację sieci logicznej).

7. *Omów podstawowe rodzaje przerzutników. Wyjaśnij zasadę działania przerzutnika typu: D, T, JK.*

Linki:

https://eduinf.waw.pl/inf/prg/009_kurs_avr/2011.php

http://zto.ita.pwr.wroc.pl/~luban/uklady_sek/przerzutniki/przerzutniki.html

BAZY DANYCH

1. Jakie są różnice pomiędzy SQL, MySQL oraz SQL Server?

Link: <https://pl.joecomp.com/difference-between-sql-and-mysql>

SQL vs MySQL

SQL oznacza Structured Query Language. Jest to standardowy język dostępu i manipulowania bazami danych. MySQL to system zarządzania bazami danych, taki jak SQL Server, Oracle, Informix, Postgres, itp. MySQL to RDMS (Relational Database Management System).

Rozważając narzędzie do zarządzania danymi, dwoma najpopularniejszymi opcjami są MySQL i SQL Server. Oba są skuteczne w utrzymywaniu uporządkowanych i łatwo dostępnych danych za pośrednictwem interfejsu użytkownika. Obie technologie mają koncepcję schematu (czyli przechowywania w tabeli) do przechowywania danych.

SQL jest językiem. W szczególności "Structured Query Language". Teraz byłoby lepiej, gdybyśmy zaczęli różnicować temat jako różnicę między serwerem SQL a MySQL i przenosić je punkt po punkcie.

SQL Server i MySQL Dostawcy:

Projekt rozwojowy MySQL udostępnił swój kod źródłowy na warunkach Powszechniej Licencji Publicznej GNU, a także na podstawie różnych zastrzeżonych umów. MySQL był własnością i był sponsorowany przez jedną firmę nastawioną na zysk, szwedzką firmę MySQL AB, która obecnie należy do Oracle Corporation.

SQL Server jest własnością firmy Microsoft i jest zwykle określane jako Microsoft SQL Server. Ma długą historię wydań i jest aktualizowana, często dodając do niej wszystkie najnowsze trendy i technologie, dzięki czemu jest dziś jedną z zaufanych aplikacji bazodanowych.

Mocne strony: SQL Server i MySQL

Aby dać lepsze wyobrażenie o różnicach w MySQL i SQL Server -MySQL jest nastawiony bardziej na wybór danych, aby można było je wyświetlać, aktualizować i zapisywać ponownie. MySQL jest słabszy w obszarach wstawiania i usuwania danych. Jest to jednak doskonały wybór do przechowywania danych i tworzenia odniesień do danych.

Oto kilka konkretnych różnic technicznych w MySQL i SQL Server, gdy sprowadza się do standardu ANSI SQL: funkcje takie jak procedury przechowywane, wyzwalacze, widoki i kursory stały się część serwera baz danych MySQL w wersji MySQL 5.0 i nadal nie znajdziesz bogatego zestawu funkcji pod względem funkcji i możliwości rozwoju. Jednak przechowywane obiekty kodu MySQL są zbliżone do standardów ANSI, ale znowu nie mają szerokości i głębokości T-SQL , zastrzeżonego rozszerzenia Microsoft i Sybase na SQL.

Bezpieczeństwo: SQL Server i MySQL

Bezpieczeństwo jest głównym problemem zarządzania danymi. Obie technologie, które są MySQL i Microsoft SQL Server, to skarga EC2 i upewnij się, że mają odpowiednie wsparcie bezpieczeństwa dla budowania aplikacji rządowych. Idąc dalej, Microsoft SQL Server jest liderem w oferowaniu wszechstronnych zabezpieczeń, ponieważ analizator bezpieczeństwa Baseline Security Analyzer firmy Microsoft pomaga administratorom zapewnić aktualność instalacji programu SQL Server. MySQL nie ma takiego narzędzia.

Wsparcie: SQL Server i MySQL

Zarówno serwer SQL, jak i MySQL mają wsparcie od swoich dostawców, zarówno w formie bezpłatnej, jak i płatnej. MySQL, jak wiemy, jest obecnie filią firmy Oracle, która jest modelem

Capability Maturity Model (CMM) na poziomie 5 i oferuje wsparcie za pośrednictwem przedstawicieli technicznych i "Virtual MySQL DBA Assistant".

Z drugiej strony, Microsoft był pionierem serwera SQL przez lata i zapewniał pomoc w swojej bazie danych SQL i pamięci masowej w chmurze. Co więcej, bezpłatny program Microsoft SQL Server Migration Assistant (SSMA) ułatwia migrację danych z Oracle, Microsoft Access, MySQL i Sybase do SQL Server.

Wniosek: Mój SQL vs SQL Server

widziałem różnicę między SQL Server i MySQL, obraz jest teraz prawie jasne. Wszystko sprowadza się do twoich potrzeb, jak wiele bezpiecznej, skalowalnej i wydajnej bazy danych chcesz. Z większością punktów wynika, że Microsoft SQL Server zapewnia dodatkowe funkcje w MySQL i jest bardziej zaufany na rynku deweloperskim.

2. Charakterystyka procesu projektowania bazy danych.

Linki:

<https://edux.pjwstk.edu.pl/mat/251/lec/wyklad5/norm.htm>

http://www.informatyka.orawskie.pl/?pl_projektowanie-bazy-danych,113

3. Przetwarzanie informacji w bazach danych. Porównaj metody oraz zastosowanie.

Link: http://firma.orawskie.pl/?pl_bazy-danych,96

Przetwarzanie danych to uporządkowane wykonywanie operacji na danych:

- modyfikowanie
- wyszukiwanie
- sortowanie
- prezentowanie (wyświetlanie, drukowanie)

4. Porównaj typy baza danych i system zarządzania bazą danych.

Link: <https://hostovita.pl/blog/porownanie-relacyjnych-systemow-zarzadzania-bazami-danych-sqlite-mysql-postgresql/>

5. Jakie są typy bazy danych, jaki wybrałbyś do np. małego sklepu internetowego, dużego portalu społeczeństwowego.

Linki:

<https://chcenawczoraj.pl/software/co-to-jest-baza-danych-jakie-rodzaje-baz-warto-znac>

6. Do czego wykorzystujemy relacyjne bazy danych, a do czego noSQL?

<https://itwiz.pl/czym-jest-nosql-jak-wykorzystac-nierelacyjne-bazy-danych/>

https://teamquest.pl/blog/461_relatyjne-bazy-danych-dlaczego-warto-je-znac

7. Optymalizacja zapytań do baz danych – podaj podstawowe metody na przykładzie.

<https://mansfeld.pl/bazy-danych/optymalizacja-bazy-danych-mysql/>

Odpowiedni dobór typów danych do kolumn

Czy wiesz, że jeżeli znasz długość stringu jaki będziesz przechowywać w polu to lepiej z punktu widzenia optymalizacji używać CHAR niż VARCHAR? Zysk pod względem wydajności jest dwukrotny! Tak na marginesie, statyczna alokacja pamięci zawsze była, jest i zawsze będzie szybsza.

Czy wiesz, że TIMESTAMP zawiera dwa razy mniej miejsca w pamięci niż DATETIME? Czytaj więcej w artykule: typy danych w MySQL.

1.2 Kolejność kolumn w tabelach

Na wydajność bazy danych ogromny wpływ ma kolejność kolumn w poszczególnych tabelach. Kolumny powinniśmy dodawać według następującej kolejności:

1. Kolumna z kluczem głównym (Primary Key)
2. Kolumna z kluczem obcym (Foreign Key)
3. Kolumny często wyszukiwane (najczęściej padające po słowie WHERE)
4. Kolumny często uaktualniane (najczęściej padające po słowie SET)
5. Częściej używane kolumny typu NULL
6. Rzadziej używane kolumny typu NULL

2. Mierzymy czasy zapytań MySQL

Mimo, że wprawiony programista z doświadczenia „wie”, które zapytania kosztują bazę sporego wysiłku, a które raczej nie stanowią dla niej problemu, to warto od czasu do czasu przyjrzeć się z pokorą i prześledzić czas wykonania się zapytań. Osobiście, przy tworzeniu oprogramowania testuję zapytania w phpMyAdmin – wiem, mało wygodne rozwiążanie. Zazwyczaj używa się do tego profilera. Odsyłam do dokumentacji, gdzie można prześledzić najpopularniejsze przypadki użycia:

3. Optymalizacja kodu aplikacji

MySQL wspiera wielowątkowość procesora i sam silnik bazy danych jest wyspecjalizowany do przetwarzania danych zawartych w bazach danych. Dlatego jeżeli np. chcesz wyciągnąć średnią lub zsumować rekordy zazwyczaj lepiej jest użyć wbudowanych funkcji w MySQL niż tworzyć sobie „własne pętelki” w PHP. Baza danych to nie tylko „głupi worek na dane”. Właśnie po to są tworzone funkcje jak AVG() i SUM() aby przyspieszyć aplikacje i przy okazji ułatwić pracę programistom.

Jeżeli musisz przetwarzać dane w logice aplikacji, najpierw je wyciągnij do zmiennej przechowującej tablicę wyników, następnie zamknij połączenie a potem to już rób z danymi co ci się podoba.

4. Optymalizacja zapytań MySQL

Najczęstszym problemem w zapytaniach jest nieszczęsne SELECT *. Im więcej danych wyciągamy, tym gorzej. Baza zwraca większe pakiety danych co jak wiadomo ujemnie wpływa na wydajność. Definiuj na sztywno, z których kolumn dane są ci potrzebne. Każda kolumna niepotrzebnie zwrócona w zapytaniu to marnotrawstwo.

Fajną sprawą jest też funkcja EXPLAIN. Dzięki niej można sprawdzić jak silnik bazy podchodzi do konkretnego zapytania, czy używa indeksów, jeżeli tak to jakich.

Nie stosuj podzapytań, nie rób JOINów kilkunastu tabel na raz a wszystko będzie w porządku.

5. Redundancja danych!

Redundancja danych czyli celowe zdenormalizowanie danych w celu przyspieszenia ich przetwarzania. Optymalizacja ta polega na specjalnym złamaniem reguły normalizacji baz danych i przechowywania tych samych informacji w dwóch miejscach po to by te dane szybciej wyciągnąć. Niby nie powinno się tego robić bo łamie sens relacyjnych baz danych ale wykonany w przemyślany sposób, (czyli z zabezpieczeniami przed utratą integralności) trik z danymi nadmiarowymi może przynieść niesamowite rezultaty – tym bardziej jeżeli łączone tabele zaczynają się rozrastać.

Rozwiązania tego typu uważam za kwintesencję informatyki, inżynierii baz danych oraz architektury. Tak samo jak w matematycznych zadaniach optymalizacji czasem używa się metod Monte Carlo. Oczywiście o tej metodzie na studiach magisterskich się tylko wspomina – a szkoda. Bo jak można uzyskać nieco mniej dokładny wynik dużo mniejszym nakładem pracy to właśnie tak (w większości przypadków) się powinno zrobić.

Prosty przykład z życia: wartość końcowa faktury może być przechowywana w polu faktury.wartosc_faktury mimo, że można ją otrzymać poprzez zsumowanie pozycje_faktury.wartosc_pozycji. Wykonując kwerendę odpytującą o np. roczny przychód firmy na podstawie wystawionych faktur wystarczy wtedy zsumować wartość z faktur a nie sumę sum z poszczególnych pozycji.

Dzięki temu zabiegowi, który polega na (mogłoby się wydawać) stworzeniu niepotrzebnego nadmiarowego pola, kwerendy z tego typu raportami mogą się wykonywać szybciej o tyle razy ile wynosi średnia liczba pozycji na fakturze! Jeżeli na fakturach jest zazwyczaj jedna pozycja cały trik jeszcze pogorszy sprawę ale jeżeli na fakturach jest 10 pozycji, raport roczny otrzymamy ok. 10 razy szybciej. To dlatego nie da się stworzyć uniwersalnego poradnika jak optymalizować bazy danych, bo wszystko zależy od konkretnego przypadku.

6. Optymalizacja serwera MySQL

Zmiana miejsca przechowywania plików tymczasowych jest jednym z najprostszych trików. Zamiast zapisywać pliki tymczasowe do RAM zapisz je na dysku. Pamiętaj, że dysk jest wolniejszym medium do przechowywania danych dlatego jeżeli serwer będzie intensywnie z nich korzystał zmiana wpłynie niekorzystnie na czas wykonywania się zapytań MySQL.

7. Indeksowanie

Indeksy to szczególny przypadek redundantnych danych, są to pary klucz – lokalizacja, które mają za zadanie zwrócić wyniki bez przeszukiwania całych tabel (coś jak spis treści w książce). Najlepsza zasada indeksowania: indeksujemy możliwe jak najmniej pól, a jeżeli już musimy indeksować to te pola, które najczęściej padają po słowie WHERE.

Świętym rozwiązaniem jest tworzenie osobnego indeksu dla dwóch tabel. Korzysta się z tego wtedy, kiedy często łączymy dwie tabele. Wszystkie zapytania, w których dochodzi do ich łączenia będą się wykonywać radykalnie szybciej.

Jeżeli w aplikacji wykorzystujesz wyszukiarkę słów kluczowych (np. wyszukiwarka w CMSie lub sklepie internetowych) stwórz indeks typu fulltext i odpytuj nie za pomocą LIKE '%słowo kluczowe%' ale MATCH AGAINST. Warto znać takie triki jeżeli zabieramy się za tworzenie własnych CMSów i systemów sklepowych.

8. Wybór silnika składowania danych

Silnik InnoDB obsługuje transakcje, zapewnia lepszą współbieżność ale jest wolniejszy zarówno przy odczytcie jak i zapisywaniu. Z kolei MyISAM źle znosi jednoczesne odczytywanie i zapisywanie. Mechanizm Memory jest nieporównywanie szybszy jeżeli będziemy z niego korzystać jak z pamięci podręcznej.

Czy wiesz, że dla każdej tabeli można niezależnie wybrać z jakiego systemu składowania danych ma ona korzystać? Czytaj więcej w krótkim porównaniu trybów składowania danych w MySQL.

7. W jakim celu stosowane są wyzwalacze w bazach danych – podaj przykłady.

<https://pl.wikipedia.org/wiki/Wyzwalacz>

https://www.plukasiewicz.net/Artykuly/SQL_Triggers_Part_I

8. Współczesne bazy danych w odniesieniu do modelu relacyjnego.

<https://bazy4danych.prv.pl/rodzaje-baz-danych>

<http://mst.mimuw.edu.pl/lecture.php?lecture=bad&part=Ch1>

9. Agregacja danych – gdzie ja lepiej zrobić po stronie serwera, czy użytkownika końcowego, podaj przykład

Operacje agregacji i grupowania

Agregacja:

- łączenie danych z wielu rekordów,
- zwykle znacząco redukuje liczbę rekordów,
- możliwe zliczanie rekordów,
- dotyczy danych numerycznych,
- umożliwia wyliczanie wartości zdefiniowanych funkcji,
- działa również w przypadku wartości NULL,
- tracone są dane *individualne*.

Grupowanie:

- możliwe jest tworzenie grup na poziomie:
 - całej tabeli – jedna grupa;
 - grupy w tabeli – jeden stopień grupowania,
 - grupy z podgrupami – grupowanie dwustopniowe,
 - grupowanie hierarchiczne.
- możliwa jest eliminacja grup,
- możliwe jest zliczanie rekordów w grupach,
- możliwe jest porządkowanie wg parametrów grup,
- parametry definiujące grupy mogą być wyświetlane.

Funkcje agregacji

-
- AVG (<exp>) — Średnia z wartości wyrażenia <exp>,
 - COUNT (<exp>) — Liczba wierszy dla których <exp> nie jest NULL,
 - COUNT (*) — Liczba wszystkich wierszy,
 - COUNT (DISTINCT (<exp>)) — Liczba wszystkich wierszy dla których wartość <exp> jest różna,
 - MAX (<exp>) — Maksymalna wartość <exp>,
 - MIN (<exp>) — Minimalna wartość <exp>,
 - STDDEV (<exp>) — Odchylenie standardowe dla <exp>,
 - SUM (<exp>) — Suma wartości <exp>,
 - VARIANCE (<exp>) — Wariancja dla <exp>.

Lepiej robić na stronie serwera.

INŻYNIERIA OPROGRAMOWANIA

- Wskaż w jakich fazach procesu wytwarzanego oprogramowania przeprowadza się walidację i weryfikację statyczną, a w jakich walidację i weryfikację dynamiczną. Z jakimi dokładnie produktami jest to związane?**

<https://qabrio.pl/walidacja-i-weryfikacja/>

Weryfikacja:

Egzaminowanie poprawności i dostarczenie obiektywnego dowodu, że produkt procesu wytwarzania oprogramowania spełnienia zdefiniowane wymagania.

Walidacja:

Sprawdzanie poprawności i dostarczenie obiektywnego dowodu, że produkt procesu wytwarzania oprogramowania spełnienia potrzeby i wymagania użytkownika.

Z powyższych definicji możemy wysnuć proste założenie, że walidacja to spełnienie wymagań użytkownika. Weryfikacja to sprawdzenie, czy produkt jest zgodny z architekturą. Możemy wyróżnić dwa sposoby weryfikacji, statyczną i dynamiczną. Weryfikacja statyczna jest wykonywana przed skompilowaniem kodu i może nią być np. inspekcja kodu. Weryfikacja dynamiczna jest wykonywana na działającym oprogramowaniu, z używając danych testowych.

Na zdrowy rozsądek, czy na poziomie specyfikacji, nie powinniśmy przewidzieć potrzeb użytkowników? Tak samo można by spytać, po co komu testerzy, skoro programiści mogliby pisać bezbłędnie?

Oprogramowanie, jak i specyfikacje tworzą tylko ludzie i każdy ma prawo do pomyłki. Podczas tworzenia specyfikacji nie jesteśmy w stanie przewidzieć wszystkich scenariuszy, tak samo, jak tego, czy z każda funkcjonalność jest intuicyjna i jasna dla użytkownika. Dlatego właśnie, walidacja jest konieczna podczas procesu testowania.

Szczególna rola specyfikacji w systemach Embedded

Na jakim poziomie testów będziemy więc walidować a na jakim weryfikować? Poziomy testów opisane w syllabusie ISTQB to:

- testowanie modułowe
- testowanie integracyjne
- testowanie systemowe
- testowanie akceptacyjne

Z definicji weryfikacja następuje podczas testów modułowych, integracyjnych i systemowych. Walidacja podczas testów akceptacyjnych. Jednak z doświadczenia wiem, że testerzy podczas testów manualnych, często zgłaszają subiektywne opinie na temat działania oprogramowania, niepowiązane z danymi ze specyfikacji.

- Przedstaw uwarunkowania trójkąta inżynierii oprogramowania oraz wskaż w jaki sposób możemy w projekcie minimalizować ryzyka (w postaci zagrożeń) związane z jego atrybutami.**

<https://present5.com/menedzhment-proektov-programmnogo-obespecheniya-09-2011-neopravdannye/>

- Posługując się praktycznym przykładem wyjaśnij na czym polega planowanie adaptacyjne. Jaki model cyklu życia oprogramowania wspiera planowanie tego typu?**

https://mfiles.pl/pl/index.php/Adaptacyjne_zarządzanie_projektami

<https://www.gomodelcanvas.pl/blog/13-jak-zaprojektowac-model-celow-dla-strategii-adaptacyjnej>

Adaptacyjne zarządzanie projektami (ang. Agile Project Management) to podejście wykorzystujące zbiór różnych metodyk, określanych jako zwinne, lekkie lub elastyczne (ang. Agile Methodologies) oraz narzędzi stosowanych w zarządzaniu złożonymi i innowacyjnymi projektami.

Adaptacyjne zarządzanie projektami charakteryzuje się stałą współpracą z klientem, dlatego też ramy projektu nie są ściśle określone, a sam projekt podzielony jest na mniejsze części tzw. funkcjonalności (iteracje). Zespoły Agile dodatkowo często dokonują zmian i korekt w nawiązaniu do wymogów i oceny klienta, uwagę skupiając głównie na pracy nad powierzonym zadaniem.

Szybkie dostosowanie się i otwartość na zmiany w projekcie jest podstawą metodologii Agile. Nie wyróżnia się oddziennie wyodrębnionej fazy projektowania jak w tradycyjnym zarządzaniu projektami.

Dynamiczne otoczenie jak i zmieniające się wymagania klienta wymagają krótkoterminowego planowania i zaangażowania całego zespołu projektowego od którego oczekiwane jest duże zdyscyplinowanie i umiejętność współdziałania, z powodu braku jednoznacznego określenia struktury organizacyjnej. Kierownik projektu zaś pełni rolę mentora. Zespoły są ograniczone jedynie do kilku-kilkunastu osób i ich charakterystycznymi cechami są elastyczność, duży poziom kooperacji oraz znaczna efektywność.

Indywidualizm to kolejny aspekt wyróżniający się w metodach zwinnych. Odchodzi się w tym przypadku od standaryzacji. Cechą charakterystyczną adaptacyjnego zarządzania projektami jest stanowcze zmniejszenie ilości dokumentów. Metody Agile zapewniają gamę technik, dzięki którym możliwe jest zaczęcie realizacji projektu bez pewności wykonania założonych celów. Proponują również metody uporządkowania zadań w projekcie, aby zagwarantować że zespół będzie realizował odpowiednie czynności, bez znajomości całego zakresu projektu.

Metody oferowane w ramach adaptacyjnego zarządzania projektami nie są adekwatne do wszystkich rodzajów projektów, szczególnie takich które są bardzo duże i rozbudowane oraz niezbędne są spore wydatki technologiczne. Metody Agile nie nadają się również w zagmatwanych projektach, gdzie ciężko klientowi jest zrozumieć jak będzie wyglądał efekt końcowy.

4. Czy w przypadku zastosowania planowania predykcyjnego możemy zastosować kaskadowy model zarządzania cyklem wytwarzania oprogramowania? Uzasadnij swoją odpowiedź posługując się praktycznym przykładem.

<https://docplayer.pl/2766622-Tworzenie-oprogramowania-nie-jest-sprawa.html>

leca się planowanie predykcyjne. Planowanie predykcyjne jest bardzo często narzucone w projektach rządowych i dla wojska. Trudno wyobrazić sobie tutaj inny sposób planowania, ponieważ mamy najczęściej sztynową datę zakończenia i kwotę budżetu. Ustalenia pomiędzy twórcą systemu i zamawiającym muszą jednoznacznie precyzować, co właściwie jest do zrobienia, ile produkt finalny będzie kosztował i kiedy zostanie ukończony. To dlatego w tego typu projektach możliwe jest wykorzystanie procesu kaskadowego. Dla administracji nic przecież nie jest bardziej kłopotliwe niż brak jasnej wizji kosztu wytworzenia jakiegoś produktu i czasu jego realizacji.

Powstaje jednak pytanie, czy projekty informatyczne mogą być w ogóle przewidywalne. Bez wątpienia w większości projektów można się spotkać z „chaosem wymagań”. Chaos polega na wprowadzaniu zmian do wymagań w późniejszych etapach cyklu życia oprogramowania. Zmiany takie praktycznie zawsze powodują konieczność przebudowy pierwotnie stworzonego planu projektu. Oczywiście można próbować walczyć z taką sytuacją. Powszechnie stosowanym sposobem radzenia sobie ze zmianami „życzeń klienta” jest zamrożenie na pewnym etapie zbioru wymagań. Powstaje jednak pewien problem. Zamrożenie wymagań powoduje ryzyko stworzenia produktu, który właściwie nie jest potrzebny klientowi już w chwili wdrażania.

Można jednak inaczej próbować podejść do problemu zmieniających się wymagań. Zakładamy mianowicie, że „chaos wymagań” jest zjawiskiem nieuniknionym, a pełna przewidywalność to iluzja. Doskonale sprawdza się wówczas planowanie adaptacyjne, które traktuje zmiany jako coś zupełnie naturalnego. Zmiany muszą być oczywiście kontrolowane. Ciekawą rzeczą jest fakt, że w przypadku planowania adaptacyjnego ciężko powiedzieć, że system jako całość rozwija się zgodnie z planem, a to dlatego, że taki plan ulega ciągłej modyfikacji. Oznacza to tyle, że plan służy raczej do możliwości oszacowania konsekwencji wprowadzenia zmian niż do przewidywania, kiedy system zostanie oddany w ręce klienta. W przypadku planowania adaptacyjnego można oczywiście na stałe określić budżet przewidziany na projekt i czas jego zakończenia, jednak nie można przewidzieć zakresu wymagań, które zostaną zrealizowane. Planowanie adaptacyjne wymaga ścisłej permanent-

nej współpracy zespołu projektowego z klientem, a zbiór wymagań może być dość elastycznie modyfikowany, rozszerzany, a niekiedy zawężany, oczywiście wszystko za poznaniem obu stron. W tego typu projektach zastosowanie modelu kaskadowego z góry skazane jest na niepowodzenie. Plan adaptacyjny możliwy jest do zastosowania jedynie w przypadku zastosowania procesu iteracyjnego i jego licznych modyfikacji, z których niektóre przedstawione zostały w artykule.

Podsumowanie

Badania prowadzone w Stanach Zjednoczonych wykazały, że ponad 2/3 wszystkich projektów informatycznych kończą się niepowodzeniem. Niepowodzenie było najczęściej rezultatem braku środków finansowych na kontynuowanie projektu (nedoszczeganie kosztów), stworzenie produktu, który nie spełnia wymagań klienta lub zakończenie procesu wytwarzania z bliżej nieokreślonych względów (często politycznych).

Powodów porażki może być wiele, ale najczęściej wskazywanymi były:

- brak większego zaangażowania ze strony klienta;
- zbyt ogólnie sformuowane wymagania lub ich modyfikacja;
- brak „właściciela” projektu;
- brak planu działania.

Wydaje się więc, że rozważania na temat procesu wytwarzania oprogramowania są potrzebne, ponieważ twórcom systemów informatycznych zależy na tym, aby sukces jednego projektu przekładał się na następny, aby był powtarzalny. Takie podejście daje możliwość doskonalenia procesu wytwarzania oprogramowania poprzez dokonywanie pomiarów i osiągania, a to z kolei wpływa na polepszenie jakości produktu i zadowolenie klienta.

Nie wolno zapominać, że proces wytwarzania oprogramowania powinien być wspomagany przez narzędzia CASE, które oczywiście wymagają odpowiedniej konfiguracji na potrzeby danego projektu. Umiejętność wykorzystania tego typu narzędzi jest praktycznie warunkiem koniecznym powodzenia każdego większego przedsięwzięcia informatycznego. ■

- 5. Jakie uwarunkowania projektu powinna obejmować analiza na poziomie wymagań biznesowych, a jakie na poziomie wymagań użytkowników i poziomie wymagań funkcjonalnych?**
- http://www.cs.put.poznan.pl/jrojek/files/io1/requirements/Wymagania_opis.pdf

Postawienie problemu i ustalenie zakresu

Dla każdego projektu informatycznego (i nie tylko), aby miał sens i warto było go realizować, powinien zostać najpierw zdefiniowany **problem**, który powinien zostać rozwiążany lub po prostu motywacja do podjęcia działania. Jest to często bardzo trudna część cyklu tworzenia projektu - czasami problem jest powszechnie znany lub napotykany bardzo często, jednak należy umieć go dostrzec i odpowiednio zdefiniować. Często zaczyna się realizować konkretny projekt myśląc o tym, "co" chcemy stworzyć, natomiast zapomina się "dlaczego". Jest o tyle ważne, że mając na względzie problem, który chcemy rozwiązać, ma się większą motywację do ukończenia projektu oraz tego, co właściwie chcemy zrobić - jest mniejsza szansa na to, że projekt zacznie ewoluować w zupełnie nieznanym kierunku i nie będzie widać do czego dążymy. Warto sobie również uświadomić znaczenie tego, co chcemy osiągnąć - czy ma to wymiar zaledwie lokalny (jak w przypadku np. aplikacji usprawniającej sesje gier karcianych) czy znacznie większy (jak ułatwienie zarządzania kadrami pracowników na uczelni).

Zdefiniowanie problemu pozwala postawić sobie cel, zwany również **celem biznesowym**. Powodem rozpoczęcia realizacji projektu jest chęć osiągnięcia celu i z tego powodu powinien on być stały (lub względnie stały), mimo często zmieniających się wymagań lub wizji aplikacji. Należy mieć na uwadze, że często na problem napotykają osoby niezwiązane z informatyką, a ich dziedziny mogą być najróżniejsze. Na dodatek, osoby potencjalnie korzystające na powstałej aplikacji mogą mieć różne cele, które mogą być ze sobą sprzeczne.

Postawienie celu determinuje ustalenie **zakresu** funkcjonalności projektu (w bardzo ogólnej postaci). Z uwagi na często bardzo szeroką problematykę, nie zawsze należy zakładać, iż uda się dotknąć każdego aspektu poruszanej tematyki. Dlatego jasne zdefiniowane zakresu pozwala na skupieniu się na konkretnych rzeczach, ułatwia specyfikację wymagań i stanowi też czynnik psychologiczny ("tego nie zrobimy, ale tak zakładaliśmy, czyli nie czujemy, że coś pomijamy").

Opowieści użytkowników

Wspomniano wcześniej o tym, iż użytkownicy mają różne cele, których chcieliby zrealizować i pewne oczekiwania. Często posiadają również pewną wizję korzystania z systemu, co jest istotne o tyle, że pozwala nam łatwiej zacząć definiować konkretne procesy, wymagania, a nawet prototypować interfejs użytkownika. Takie zdania opisujące wizję są nazywane opowieściami użytkowników (ang. *user stories*).

Opowieści mogą przybierać różną formę. Mogą to być pojedyncze zdania "zapisane" przez różnych użytkowników, czasami na małych karteczkach, które ułatwiają zarządzanie tym, co rzeczywiście powinno zostać zrealizowane.

Jako student chcę mieć możliwość przeglądania ocen, które zostały mi wystawione.

Jako wykładowca chcę mieć dostępную listę studentów, którym mogę wystawić oceny.

Jako pracownik dziekanatu chcę móc w łatwy sposób wydrukować wszystkie protokoły.

Zamiast pojedynczych zdań, opowieści często mogą być bardziej rozbudowane i rzeczywiście przypominać pewną historię oraz opis tego, jak użytkownik chce używać systemu.

Wymagania funkcjonalne i przypadki użycia

Wymagania funkcjonalne to konkretne aspekty funkcjonalności aplikacji, które definiują działania, które mogą być udostępniane dla użytkownika czy samego oprogramowania. Ich specyfikacja jest jednym z kluczowych punktów inicjowania projektu, gdyż dobrze zdefiniowane i opisane wymagania mogą znacznie uprościć implementację oraz analizę wprowadzanych modyfikacji.

Ich zbieranie można zacząć od procesów biznesowych (zwykle kolejne punkty opisują co najmniej jedno wymaganie) oraz opowieści użytkownika, gdzie „zaszyta” jest konkretna funkcjonalność. Sam proces identyfikacji wymagań może następować od najbardziej ogólnych do szczegółowych (tzw. podejście *top-down*), od szczegółowych do najbardziej ogólnych (*bottom-up*) lub w sposób mieszany.

Najpopularniejszym sposobem opisu wymagań funkcjonalnych są **przypadki użycia** (ang. *use cases*). Można je określić jako listy kroków (składających się na scenariusz) kolejnych interakcji użytkownika z systemem w celu realizacji danego wymagania. Jest to forma przyjazna zarówno dla klienta (gdyż nie wymaga technicznej wiedzy), analityka (jest prosta w użyciu) oraz programisty (wiadomo dokładnie jak należy zaimplementować wymaganie), ale także testera (który na tej podstawie może łatwiej ułożyć scenariusze testowe).

- 6. Na co dokładnie mają wpływ w projekcie programistycznym wymagania pozafunkcjonalne opisujące usługi lub charakterystyki wydajnościowe produktu programistycznego?**

<https://docplayer.pl/6113541-Wymagania-pozafunkcjonalne.html>

- 7. Czym się różnią wymagania systemowe od wymagań funkcjonalnych i wymagań determinowanych przez interfejsy zewnętrzne dla projektowanego oprogramowania?**

https://www.mimuw.edu.pl/~mmozdzonek/2008.wo/zsi_ wo_w02.pdf

https://edux.pjwstk.edu.pl/mat/200/lec/wykladы/3_4.html

Opis procesów biznesowych

Z analizy cech systemu oraz modelu kontekstowego wiadomo, które procesy trzeba przeanalizować. Szczegółowo należy analizować tylko procesy znajdują-ce się w zakresie projektu. Modele procesów powinny odwzorowywać wszystkie działania realizowane przez przedsiębiorstwo na różnych szczeblach zarządzania, które są przedmiotem projektu. Stosując metodę top-downd modelowania procesów, należy wyjść od zagregowanego modelu przedsiębiorstwa. Następnie przez dekompo-zycję uzyskujemy bardziej szczegółowe opisy działalności realizowanej na po-ziomie funkcji/działów/zespołów. Poziom organizacyjny jest reprezentowany wmodelu jako tor. Notacją stosowaną do modelowania procesów jest BPMN. W podejściu do modelowania procesów można przyjąć założenie, że cała działalność przedsiębiorstwa to reagowanie na zdarzenia zewnętrzne i produko-wanie rezultatów. Proces można zdefiniować jako sekwen-cję funkcji, które trzeba wykonać, aby optymalnie zareagować na zdarzenie ze-wnętrzne poprzez wygenerowanie odpowiednich rezultatów.

Lista wymagań funkcjonalnych

Wymagania funkcjonalne definiują,co system musi robić dla użytkowników. Opisują czynności, operacje, usługi wykonywane przez system. Lista wymagań funkcjonalnych jest stworzona na podstawie dotychczas zgromadzonej wiedzy,a w szczególności na podstawie scenariuszy użycia.Po sporządzeniu listy wymagań należy oszacować priorytet każdego z nich, jak również którą cechę systemu realizuje. Wymagania nierealizujące żadnej cechy systemu są życzeniami użytkowników niezbląającymi projektu do realizacji postawionego mu przez klienta celu i powinny być w pierwszej kolejności redukowane.

Lista wymagań niefunkcjonalnych

Wymagania niefunkcjonalne definiują, w jakich warunkach system musi realizować wymagania funkcjonalne. Nie mają one związku z cechami systemu, gdyż określają bezwzględnie konieczne wymogi. Wynikają z potrzeb użytkownika, budżetu, strategii firmy, czynników zewnętrznych, współpracy z innymi systemami. Dotyczą systemu jako całości, a nie poszczególnych cech systemu. W trakcie tworzenia listy wymagań niefunkcjonalnych należy je sklasyfikować w niżej wymienione grupy: □ produktowe (użyteczność, wydajność, niezawodność, mobilność), □ organizacyjne (dostawy, standardy, implementacyjne, czynnik ludzki), □ zewnętrzne (współpracy, etyczne, prawne, zabezpieczenia).

Definicja przypadków użycia

Diagram przypadków użycia odgrywa najważniejszą rolę w procesie projektowania systemu, ponieważ opisuje wymagania funkcjonalne, jakim system musi sprostać, i otoczenie, w którym się znajduje. Przypadek użycia jest zbiorem scenariuszy związanych ze sobą wspólnym celem użytkownika. Pozwala na zdefiniowanie przyszłego spodziewanego zachowania systemu. Przypadek użycia musi wchodzić w interakcję chociaż z jednym aktorem. Przypadek użycia można opisać za pomocą takich cech, jak (Cock-burn, 2004): nazwa, cel biznesowy, aktorzy, warunki wstępne, warunki końcowe, przepływ główny, przepływy alternatywne. W celu oceny priorytetu przypadku użycia należy wziąć pod uwagę korzyść biznesową, koszt, ryzyko, priorytet.

Interfejsy zewnętrzne

Rzadko który system może samodzielnie istnieć bez współpracy z innymi systemami informatycznymi. Stąd też istotną kwestią jest przygotowanie dokumentacji interfejsów, która opisuje szczegóły działania interfejsu: opis, format i struktura interfejsu, tabela mapowań, formaty pól, sposób kodowania, wyzwalacze, harmonogram.

8. Przedstaw istotne różnice pomiędzy strategią pozyskiwania wymagań zorientowaną na użycie a strategią pozyskiwania wymagań zorientowaną na produkt.

<http://analizawymagan.pl/techniki-identyfikacji-wymagan-kto-re-sprawdza-sie-w-agile-i-nie-tylko/>
<http://www.zim.pcz.pl/znwz/files/Strategia-informatyzacji-i-analiza-przedwdro-eniowa-a-cykli--ycia-oprogramowania-standardowego.pdf>

https://wwwdbc.wroc.pl/Content/998/fraczkowski_zarzadzanie_projektem.pdf

Pozyskiwania wymagań funkcjonalnych i nie funkcjonalnych

9. Wymień co najmniej trzy główne aktywności zarządzania wymaganiami dla produktu programistycznego, które nie przyczynią się do powstania konieczności wprowadzania zmian w projekcie.

<https://wolski.pro/2017/03/plan-zarzadzania-wymaganiami/>
https://pl.wikipedia.org/wiki/Zarz%C4%85dzanie_wymaganiami

Ograniczeniami w projekcie są czynniki, które mają podstawowy wpływ na opcje działań kierownika projektu. Typowe trzy główne ograniczenia to:

- **Czas** – ograniczenia, takie jak stała data zakończenia lub termin ostateczny w przypadku głównych punktów kontrolnych.
- **Zasoby** – (materiał, wyposażenie, sprzęt i ludzie oraz skojarzone z nimi koszty) – ograniczenie, takie jak uprzednio zdefiniowany budżet.
- **Zakres** – ograniczenie, takie jak zakładana funkcjonalność, technologia, produkty itp. Zmiana jednego z wymienionych ograniczeń zwykle wpływa na dwa pozostałe, a także na jakość projektu.

Na przykład zmniejszenie czasu trwania projektu (harmonogram) może zwiększyć liczbę pracowników potrzebnych do realizacji planu (zasoby) oraz zmniejszyć liczbę właściwości cechujących produkt (zakres). Menedżer projektu musi określić, czy można zaakceptować taką degradację. Taki związek jest nazywany potrójnym ograniczeniem zarządzania projektem lub trójkątem ograniczeń projektu. Podczas procesu planowania należy sporządzić listę ograniczeń projektu, aby upewnić się, że wszyscy wykonawcy projektu zostali o niej powiadomieni i mogą się do

niej odnieść. Właściwym dla wykonawców jest także uzgodnienie sposobu ich reakcji na niespodziewane ograniczenia, które mogą ujawnić się w czasie trwania projektu. Na przykład, jeżeli koszty pracy okażą się wyższe od przewidywanych, to wykonawcy mogą zażądać zmniejszenia zakresu projektu.

10. Jaka istotna cecha modelu spiralnego wyróżnia go w odniesieniu do każdego innego modelu zarządzania cyklem wytwarzania oprogramowania?

<https://agile247.pl/podejscie-iteracyjne-oraz-przyrostowe/>

https://mfiles.pl/pl/index.php/Metodyka_spiralna

Metodyka spiralna (ang. *spiral model*) - zakłada cykliczne realizowanie określonej sekwencji działań prowadzących do osiągnięcia rozwiązania docelowego. Stosowany jest najczęściej w bardzo dużych przedsięwzięciach informatycznych, wymagających ciągłego rozwoju i doskonalenia [Z. Handzel 2016, s. 112].

- Najważniejszą cechą modelu spiralnego jest analiza ryzyka w poszczególnej fazie prac, co stanowi próbę minimalizacji ryzyka niepowodzenia projektu.
 - Nieustanne sprawdzenie systemu/produkту przez użytkownika ma na celu wytworzenie produktu w pełni zadowalającego zleceniodawcę.
 - Budowa systemu rozpoczyna się od sformułowania głównych wymagań i założenia celów, następnie dokonuje się analiza wykonywania przedsięwzięcia razem z oceną alternatywnych rozwiązań. Opierając się na tą podstawie tworzony jest prototyp systemu.

Model spiralny wywodzi się z połączenia modelu kaskadowego oraz prototypowego. Głównym założeniem modelu jest stwierdzenie, że każda następna wersja powstającego systemu/produkту będzie stworzona na podstawie wyników wytworzenia wcześniejszych wersji (prototypów), dzięki czemu z każdym kolejnym przebiegiem procesu realizującego końcowy rezultat musi być coraz lepszy. Proces ten ma formę spirali, w której każda pętla odzwierciedla kolejne fazy procesu [M. Trocki 2017, s. 91]. Za twórcę modelu spiralnego uważa się B.W. Boehm, który zdefiniował go po raz pierwszy w 1988 roku w artykule "A Spiral Model of Software Development and Enhancement". Metodyka ta jest wykorzystywana przede wszystkim w projektach informatycznych do zarządzania przygotowaniem oprogramowania oraz opisywania, jak różne procesy współpracują z procesem projektowania [B. Bereza 2014, s. 13].



Rys. 1 Model spiralny przebiegu projektu

Model spiralny jest jedynym takim modelem ogólnym, w którym analizę ryzyka uwzględnia się na równi z fazami [planowania](#) i realizacji przedsięwzięcia [E. Bukhla 2011, s. 209]. Zakłada on w sobie etapowe dochodzenie do rozwiązania końcowego przez cykliczne wykonywanie tych samych, powtarzających się [etapów przedsięwzięcia](#) [Z. Handzel 2016, s. 112]. W sekwencyjnie realizowanych cyklach wykonywane są kolejne wersje systemu/produkту docelowego przedsięwzięcia, np. kolejne prototypy [systemu informatycznego](#). Każda taka wersja przechodzi przez dokładną kontrolę ze strony użytkownika/zleceniodawcy końcowego i p. realizację następnego etapu w cyklu zarządzania powstającym rozwiązaniem końcowym [Z. Ha rozwoju systemu/produkту jest opisany na jednym z czterech pól spirali projektowej, przedstawio

Przechodząc przez poszczególne etapy modelu na podstawie analizy ryzyka jest dokonywana ocena dalszego uzasadnienia biznesowego, wykonanej wersji systemu/produkту docelowego i zaleca się przyjęcie odpowiedniej wersji projektu do zrealizowania [R. Michalski 2008, s. 74]. W końcu każdego cyklu przedsięwzięcia, otrzymane produkty częściowe i postęp prac poddają się ocenie przez decydentów i kluczowych interesariuszy przedsięwzięcia. Następny cykl projektu może zacząć się dopiero po zatwierdzeniu przez nich efektów otrzymanych w poprzednim etapie [M. Trocki 2017, s. 91].

W zakresie prac projektowych, na różnych poziomach szczegółowości, formułowane są wymagania biznesowe, opracowany cel i koncepcja dojścia do niego, również tworzona jest koncepcja docelowego produktu, jego projekt i realizowanie [M. Trocki 2017, s. 93]. Definiowany jest plan testów i realizowane są prace związane z testowaniem, poddaje się analizie ryzyko projektowe oraz, na zakończenie poszczególnego cyklu, wykonywana jest ocena w celu zgłoszenia problemów do rozwiązań w kolejnym cyklu spiralnego wykonywania [Z. Szyjewski 2004, s. 255]. Ilość powtórzeń spirali jest zależne od rozmiaru przedsięwzięcia, jego kreatywności oraz innowacyjności, warunków zewnętrznych, które mają wpływ na project, ilości wymagań do uwzględnienia, możliwości wykonawcy prac oraz dostępnych środków i rezerw [M. Trocki 2017, s. 92]. Z racji tego, że w wyniku przejścia z jednego etapu do drugiego poddajemy ocenie dotychczasowe osiągnięcia, model spiralny pozwala zauważać ryzyko niezgodności z bieżącymi zapotrzebowaniami znacznie wcześniej [Z. Szyjewski 2004, s. 43].

Zalety i wady metodyki spiralnej

Zalety:

- Budowanie prototypów daje możliwość lepszej oceny zgodności i sprawdzenia wymagań.
- Nadaje się do dużych systemów/przedsięwzięć – pozwala na szybką reakcję na pojawiające się czynniki i wymagania [M. Trocki 2017, s. 92].
- Wielki nacisk na wykrywanie i eliminowanie zagrożeń, który skutkuje wysoką niezawodnością i szansą wykonywania przedsięwzięcia.
- Software jest stwarzany już we wcześniejszych etapach.
- Możliwość dopasowania na każdym etapie projektu.
- Zawiera połączenie iteracji z klasycznym modelem kaskadowym [B. Bereza 2014, s. 7].
- Elastyczny rozwój produktów częściowych projektu.
- Stabilność docelowego rozwiązania i stopień wykonania końcowych wymagań użytkownika.

Wady:

- Trudno do niej przekonać klientów.
- Wymaga obowiązkowej umiejętności szacowania ryzyka.
- Potrzebuje bardzo dobrze doświadczonych i wyszkolonych ekspertów do analizy planowania, ryzyka oraz relacji z klientem itd.
- W dużej mierze sukces projektu zależy od analizy ryzyka (występują poważne problemy, gdy jest źle oszacowane ryzyko).
- Model jest czasochłonny i kosztowny.
- Nie nadaje się do małych projektów.
- Ciężki i złożony do dokładnego przestrzegania.

11. Jaka jest największa wada zastosowania modelu ewolucyjnego w procesie wytwarzania oprogramowania?

Zalety i wady metodyki spiralnej

Zalety:

- Budowanie prototypów daje możliwość lepszej oceny zgodności i sprawdzenia wymagań.
- Nadaje się do dużych systemów/przedsięwzięć – pozwala na szybką reakcję na pojawiające się czynniki i wymagania [M. Trocki 2017, s. 92].
- Wielki nacisk na wykrywanie i eliminowanie zagrożeń, który skutkuje wysoką niezawodnością i szansą wykonywania przedsięwzięcia.
- Software jest stwarzany już we wcześniejszych etapach.
- Możliwość dopasowania na każdym etapie projektu.
- Zawiera połączenie iteracji z klasycznym modelem kaskadowym [B. Bereza 2014, s. 7].
- Elastyczny rozwój produktów częściowych projektu.
- Stabilność docelowego rozwiązania i stopień wykonania końcowych wymagań użytkownika.

Wady:

- Trudno do niej przekonać klientów.
- Wymaga obowiązkowej umiejętności szacowania ryzyka.
- Potrzebuje bardzo dobrze doświadczonych i wyszkolonych ekspertów do analizy planowania, ryzyka oraz relacji z klientem itd.
- W dużej mierze sukces projektu zależy od analizy ryzyka (występują poważne problemy, gdy jest źle oszacowane ryzyko).
- Model jest czasochłonny i kosztowny.
- Nie nadaje się do małych projektów.
- Ciężki i złożony do dokładnego przestrzegania.

12. Które perspektywy modelu „4+1” opisują wewnętrzną strukturę tworzonego oprogramowania na różnych poziomach abstrakcji i szczegółowości?

<http://project-media.pl/4plus1.php>

<http://www.wozna.org/students/2018-2019/uml/UML01.pdf>



Autorzy UML rozróżniają pięć perspektyw spojrzenia na system informatyczny i przyporządkowują im odpowiednie rodzaje diagramów UML:

- **perspektywa przypadków użycia** (zakres i funkcjonalność systemu) - opisuje funkcjonalność, jaką powinien dostarczać system, widzianą przez jego użytkowników, czyli opisuje zachowanie systemu obserwowane z zewnątrz; diagramy przypadków użycia, diagramy pakietów.
- **perspektywa projektowa** (logiczna, budowa systemu) - opisuje sposób realizacji funkcjonalności, strukturę systemu widzianą przez projektanta (tj. klasy, interfejsy, kooperacje); diagramy klas, obiektów, pakietów, struktur złożonych.



- **perspektywa procesowa** (dynamiczna, zachowanie)- zawiera podział systemu na procesy (czynności) i procesory (jednostki wykonawcze) oraz opisuje właściwości pozafunkcjonalne systemu i służy przede wszystkim programistom i integratorom; diagramy aktywności (czynności), maszyny stanowej, pakietów sekwencji, komunikacji, czasowe oraz przeglądowe diagramy interakcji.
- **perspektywa implementacyjna** (software) - opisuje poszczególne moduły i ich interfejsy wraz z zależnościami; perspektywa ta jest przeznaczona dla programisty (komponenty i pliki, zarządzanie konfiguracją); diagramy komponentów, diagramy pakietów.

Perspektywa 4+1 III

- **perspektywa wdrożeniowa (rozlokowanie, sprzęt)** - definiuje fizyczny podział elementów systemu i ich rozmieszczenie w infrastrukturze, czyli dotyczy fizycznej realizacji sprzętowej systemu; perspektywa taka służy integratorom i instalatorom systemu; diagramy wdrożenia, diagramy pakietów.

Modelowanie złożonych systemów jest zadaniem trudnym i angażuje wiele osób o różnym sposobie postrzegania systemu. Aby uwzględnić te punktu widzenia, UML jest często określany jako język modelowania z 4+1 perspektywą. Cztery pierwsze opisują wewnętrzną strukturę programu na różnych poziomach abstrakcji i szczegółowości. Ostatnia perspektywa opisuje funkcjonalność systemu widzaną przez jego użytkowników. Każda perspektywa korzysta z własnego zestawu diagramów pozwalających czytelnie przedstawić modelowane zagadnienie. Są to:

- Perspektywa przypadków użycia – opisuje funkcjonalność, jaką powinien dostarczać system, widzaną przez jego użytkowników.
- Perspektywa logiczna – zawiera sposób realizacji funkcjonalności, strukturę systemu widzaną przez projektanta
- Perspektywa implementacyjna – opisuje poszczególne moduły i ich interfejsy wraz z zależnościami; perspektywa ta jest przeznaczona dla programisty
- Perspektywa procesowa – zawiera podział systemu na procesy (czynności) i procesory (jednostki wykonawcze); opisuje właściwości pozafunkcjonalne systemu i służy przede wszystkim programistom i integratorom
- Perspektywa wdrożenia – definiuje fizyczny podział elementów systemu i ich rozmieszczenie w infrastrukturze; perspektywa taka służy integratorom i instalatorom systemu

13. Omów jakie atrybuty zewnętrzne oraz jakie atrybuty wewnętrzne powinny zostać zawsze zdefiniowane metrycznie (określone ilościowo) w przypadku specyfikacji, jakie w przypadku kodu źródłowego, a jakie dla danych testowych.

http://iso-cmm.eprace.edu.pl/19.Atrybuty_jakosci.html

Powyższa definicja rozważa jakość w dwóch aspektach. Po pierwsze należy sprecyzować "czego" jakość jest rozważana, czyli o jaki obiekt chodzi (produkt, usługa, czynność opracowania, itp.), a z drugiej strony "czyje" potrzeby są brane pod uwagę (użytkownika, kierownika projektu, itp.).

Cechy charakterystyczne definiujące jakość nie są w istocie takie same; zależą one od punktu widzenia i będą inne dla użytkownika, programisty, czy klienta-decydenta.

- ♦ Pierwszy będzie brał pod uwagę własności zewnętrzne dotyczące przede wszystkim widocznych aspektów dostarczonego oprogramowania, takich jak łatwość użycia czy parametry (mówią tutaj o *jakości zewnętrznej oprogramowania*).
- ♦ Drugi wysunie naprzód własności takie jak testowalność lub czytelność kodu projektowanego programu (mówią tutaj o *jakości wewnętrznej oprogramowania*).
- ♦ Trzeci ze swojej strony zainteresuje się własnościami takimi jak termin i koszty realizacji czy też gwarancje zapewnienia jakości deklarowanej przez dostawcę (na przykład świadectwem zgodności z normą ISO 9001). W tym ostatnim przypadku, można zauważać, że własności definiujące jakość odnoszą się do *procesu opracowywania*¹ oprogramowania a nie do samego produktu finalnego będącego wynikiem tego procesu.

W zależności od nastawienia, jakość może także być definiowana ze względu na:

- ♦ **produkty i usługi:** z **zewnętrznego** punktu widzenia (własności dostarczonego oprogramowania lub oferty oprogramowania²) lub z **wewnętrznego** (własności oprogramowania na pewnym szczególnym etapie procesu opracowywania, takim jak na przykład dokument specyfikacji) ;
- ♦ część lub całość **procesu opracowywania:** z punktu widzenia **zewnętrznego** (obserwowalność, niezawodność, sterowalność, itp.) lub **wewnętrznego** (metody, narzędzia, reguły, procedury, standardy, normy, itp.).

14. W jaki sposób możemy poprawnie oszacować opłacalność projektu informatycznego?

<https://vc.ru/finance/150050-skolko-dolzen-stoit-it-proekt>

<https://kjarocka.pl/zarzadzanie-projektami/estymacja-wprowadzenie-do-procesu-1/>

15. Jaka jest relacja pomiędzy kosztem a opłacalnością projektu programistycznego?

<https://mfiles.pl/pl/index.php/Projekt>

16. Co należy uwzględnić w procesie estymacji nakładu pracy dla danego przedsięwzięcia programistycznego? Na co rzutuje ten atrybut procesu w projekcie?

<https://kjarocka.pl/zarzadzanie-projektami/estymacja-wprowadzenie-do-procesu-1/>

Estymacja w procesie planowania

Według Encyklopedii Zarządzania proces estymacji polega na “określeniu przewidywanego czasu trwania poszczególnych zadań projektu, który umożliwia wyznaczenie ram czasowych całego projektu oraz stworzenie harmonogramu projektu.” Krótko mówiąc, estymujemy zadania po to, żeby:

- stworzyć harmonogram i określić datę dostarczenia projektu,
- móc monitorować postęp prac, chociażby z wykorzystaniem metody EVM, która pozwala na porównanie stanu faktycznego z planowanym w odniesieniu do czasu i kosztów.

Gdybyśmy chcieli spojrzeć na proces estymacji z perspektywy macierzy PMBOK (1), warto podkreślić, że leży on w obszarze wiedzy “Zarządzanie czasem” (ang. *Time Management*) i jest częścią grupy procesowej “Planowanie”.

Estymacja

Ok, zakres już mamy – przyjmijmy, że bez dziur ;) Mamy zdefiniowanie działania do wykonania oraz znamy ich kolejność. Pozostaje nam teraz estymacja – proces wcale niełatwyy. O technikach estymacji i pułapkach tego procesu opowiem w części 2. Dzisiaj przedstawię, co na estymację może mieć wpływ niezależnie od przyjętej techniki:

- doświadczenie osób dokonujących estymacji – im mniej doświadczona osoba, tym czasochłonność zadania będzie większa, niemniej koszt powinien pozostać na podobnym poziomie (dłuższy czas wykonania zadania, ale za niższą stawkę);
- rozumienie zakresu – jest to poniekąd związane z punktem poniżej, ale również odnosi się do tego “co” będziemy wykonywać, czyli na estymację wpłynie nie tylko sposób realizacji zadania, ale ogólne rozumienie tego, co mamy do zrobienia – im bardziej nieprecyzyjna specyfikacja, tym większe pole do interpretacji;
- przyjęte do estymacji założenia – jaki poziom “skomplikowania” zadania zakładamy, jakie mamy kryteria akceptacji, co rozumiemy przez wykonanie zadania oraz “jak” zamierzamy zadanie wykonać – pewnie z własnych doświadczeń wiecie, że każde zadania wykonać można na wiele sposobów;
- nasza natura – optymiści podają mniejszą czasochłonność zadań niż pesymiści;
- doświadczenia z poprzednich projektów – jeśli nic się “nie wysypało”, będziemy szacować bardziej optymistycznie, jednak mieliśmy sporo turbulencji, estymacje będą bardziej ostrożne – nauczeni doświadczeniem, będziemy się asekurować;
- świadomość ryzyk – im większa świadomość zagrożeń, tym mniej optymistyczne estymacje, stąd dobrą praktyką jest podanie optymistycznej (założenie, że znane nam zagrożenia się nie zmaterializują) i pesymistycznej estymacji (założenie, że znane nam zagrożenia się zmaterializują) oraz przyjęcie średniej z nich.

Moim ulubionym, choć nie projektowym przykładem jest prosta czynność sprzątania mieszkania – spróbujcie ją oszacować i porównać wymagania oraz kryteria akceptacji z Waszymi drugimi połówkami ;) Na tym przykładzie doskonale zobaczycie jak bardzo można się różnić w rozumieniu – wydawać by się mogło prozaicznego – zadania.

- 17. W związku z tym, że w każdym projekcie programistycznym estymacji podlega wiele atrybutów procesu i produktów rozwoju oprogramowania, wyjaśnij w jaki sposób możemy ocenić sprawdzalność wybranych metod estymacji i procesu i produktu.**

<https://kjarocka.pl/zarzadzanie-projektami/techniki-estymacji/>

Techniki estymacji wg PMBOK

W momencie przystępowania do estymowania czasu project manager powinien posiadać dane wejściowe, które pozwolą mu zdecydować, w jaki sposób podejść do estymowania. Umiejętność wykorzystania właściwej techniki estymacji w konkretnym przypadku może mieć duży wpływ na ocenę czasu trwania projektu. Mowa tu o technikach takich jak:

- Metoda ekspercka (ang. *Expert Judgement*),
- Estymowanie przez analogię (ang. *Analogous Estimating*),
- Estymacja parametryczna (ang. *Parametric Estimating*),

- 18. Wyjaśnij w jaki sposób możemy dokonać oceny ryzyka dla realizowanego przedsięwzięcia programistycznego.**

<https://sekurak.pl/czym-jest-analiza-ryzyka-it-wprowadzenie/>

Analiza ryzyka jest jednym z głównych narzędzi zarządzania ryzykiem IT i ma na celu przede wszystkim:

- › zidentyfikować zagrożenia dla naszej infrastruktury IT,
- › wykonać ich priorytyzację (np. widzimy, które zagrożenia są dla nas najbardziej istotne),
- › oszacować potencjalne straty związane z naruszeniem naszego bezpieczeństwa,
- › zaproponować rozsądne decyzje o ryzyku (np. jego redukcję) wraz z racjonalnymi ramami budżetowymi.

Mówiąc jeszcze innymi słowy, **celem analizy ryzyka** jest podniesienie czy wręcz zbudowanie naszej świadomości dotyczącej stanu bezpieczeństwa mojego systemu IT, a w kolejnym kroku podjęcie pewnych działań zaradczych. Prowadzimy ją, przez cały czas mając na uwadze główny cel organizacji (czyli najczęściej maksymalizację profitów / minimalizację strat).

1. Pierwszy krok: inwentaryzacja zasobów IT

Bez wiedzy, co znajduje się w mojej infrastrukturze, ciężko mówić o bezpieczeństwie całości.

Zasobami są przykładowo: konkretne serwery, urządzenia sieciowe, aplikacje, pomieszczenia, w których znajdują się te urządzenia, stacje PC, komputery przenośne (w tym smartfony, tablety), konkretne bazy danych itp. Przy okazji warto tu zastanowić się nad komponentami, które nie znajdują się w mojej infrastrukturze, a które są konieczne do poprawnego działania całej organizacji (systemy działające w modelu outsourcingowym) – przykładem mogą być tutaj serwery poczty / DNS / czy witryna internetowa.

Na tym etapie analizy dla każdego zasobu możemy od razu oznaczyć jego krytyczność dla działania całej organizacji oraz określić aktualnie stosowane metody ochrony (przykładowe metody ochrony można zobaczyć [np. tutaj – Sans 20 Critical Security Controls](#)). Krytyczność może być przydatna np. do tego, aby dalszą analizę ryzyka zawieźć i przeprowadzić ją tylko dla systemów krytycznych.

2. Drugi krok: określenie potencjalnych zagrożeń dla każdego inwentaryzowanego zasobu

Na tym etapie bardzo pomocne są tzw. checklisty, czyli zestawy „standardowych zagrożeń” dla konkretnych komponentów. Dla ułatwienia – każde zagrożenie zazwyczaj można przyporządkować do jednego z czterech obszarów:

- › **zagrożenia ludzkie**, czyli wpływ działania czynnika ludzkiego na bezpieczeństwo naszego systemu IT. W tej kategorii znajdują się słynne „ataki hackerskie”, ale również zwykłe błędy ludzkie (np. administrator, który niepoprawnie przygotował backup);
- › **zagrożenia naturalne**, czyli żywioły: powódź, pożar, wichury, problemy związane z przepięciami itp.;
- › **zagrożenia techniczne**, czyli wszystko, co może zagrozić elementom normalnie niezbędnym do funkcjonowania systemów IT (np. elektryczność, odpowiednia wilgotność powietrza), odpowiednie działanie komponentów (np. dyski twarde, chłodzenie) itp.;
- › **zagrożenia administracyjne**, czyli wynikające z naruszeń obowiązujących nas przepisów prawnych (np. [ustawa o ochronie danych osobowych](#),  [Rekomendacja D KNF](#)), regulacji zewnętrznych (np.: [PCI DSS](#)) lub wewnętrznych zaleceń (np. wymaganych przez naszą centralę); czasem grupa ta wymienia jest jako podzbiór ww. trzech grup zagrożeń.

3. Trzeci krok: wskazanie podatności czyli realnych słabości naszych zasobów

Do każdego zasobu (krok pierwszy) mamy już przyporządkowany szereg potencjalnych zagrożeń (krok drugi), teraz chcemy zobaczyć, jakie realnie problemy związane z bezpieczeństwem mogą nas spotkać (tj. czy zagrożenia wskazane w punkcie wyżej mogą rzeczywiście zagrozić naszym zasobom).

PRZYKŁAD

Zasób „firmowy portal informacyjny” oznaczyliśmy m.in. zagrożeniem „atak hackerski”. Teraz chcemy sprawdzić, czy nasz system rzeczywiście posiada pewne słabości (podatności), które umożliwiłyby sukces tego ataku. W tym celu przeprowadza się często **testy penetracyjne** (czyli symulowane ataki na infrastrukturę IT – tak aby wykryć jej rzeczywiste słabości).

Zasób „główna serwerownia” i zagrożenie „zalanie”. Jeśli pomieszczenie znajduje się w piwnicy, a dodatkowo historycznie występowały na danym terenie podtopienia to teraz wskazujemy, że nasze pomieszczenie jest podatne. Zapisujemy to jako podatność.

Zasób „zewnętrzny portal dla kontrahentów” i zagrożenie „naruszenie ustawy o ochronie danych osobowych”. Jeśli w systemie wykryliśmy błędy umożliwiające nieautoryzowany dostęp do danych lub niezgodność z elementami wymaganymi **w stosownym rozporządzeniu** wykonawczym, jest to realna podatność.

4. Krok czwarty: określenie ryzyka

Teraz sprawdzimy wpływ wykorzystania konkretnych, zdefiniowanych wcześniej podatności na naszą organizację; inaczej – zweryfikujemy skutki wykorzystania słabości naszego systemu. Konkretna definicja „ryzyka” znowu zależy tutaj często od przyjętego metodologii.

Wyróżniają się dwa podejścia:

A) Podejście jakościowe

W tym modelu ryzyko zazwyczaj określane jest jako niskie / średnie / wysokie (możliwe są też inne, podobne skale), a wartość ryzyka można odczytać z tabeli, w której:

- › w kolumnach mamy **prawdopodobieństwo wykorzystania** danej podatności (niskie, średnie, wysokie),
- › w wierszach – **wpływ wykorzystania** podatności na naszą organizację (niski, średni wysoki).

Taka tabela definiuje finalne ryzyko, które niesie każda kombinacja powyższych warunków. Jako przykład przykład można zacytować dokument [OWASP Risk Rating Metodology](#):

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
Likelihood				

OWASP Risk Rating Metodology – obliczanie ryzyka.

Ryzyko jest tu więc prostą funkcją **prawdopodobieństwa wykorzystania** podatności oraz idących za tym **skutków**.

Podejście jakościowe jest zazwyczaj zalecane przy pierwszej analizie ryzyka (o wiele łatwiej oszacować ryzyko tą metodą).

B) Podejście ilościowe

W podejściu ilościowym staramy się określić **ryzyko w konkretnych liczbach** (np. stracie finansowej, którą potencjalnie generuje ryzyko). Zazwyczaj jest to metoda bardziej skomplikowana, niezalecana dla początkujących.

Najpierw **szacuje się prawdopodobieństwo** skutecznego wykorzystania podatności przez zagrożenie w zadanym okresie (najczęściej przyjmuje się tutaj jeden rok). Np. szacuję, że mój portal informacyjny zostanie zaatakowany skutecznie raz w przeciągu najbliższego roku (zrealizowałem wcześniej testy penetracyjne, z których wynika, że system zawiera istotne podatności, które umożliwiają nieautoryzowane, zdalne przejęcie kontroli nad portalem).

Oszacowanie strat. Następnie liczę potencjalne straty, czyli koszty, które poniosę w wyniku wykorzystania podatności przez zagrożenie. Powiniem tutaj oszacować możliwie wszystkie wartości takie jak: czas pracowników potrzebny na przywrócenie systemu do stanu początkowego, straty pieniężne, straty reputacji, kary wynikające z umów / ustaw / itd.

Finalnie mogę tutaj wskazać, ile wynoszą moje średnioroczne straty podczas eksploatacji danego zasobu. Pozwala mi to oszacować poziom ryzyka, ale też np. rozsądnie zaplanować budżet (widać np. że nie powinien być on większy niż straty – wtedy czasem lepiej nic nie robić – dojdziemy do tego w punkcie kolejnym dotyczącym podejmowania decyzji o ryzyku).

Często, choć nie zawsze, analiza ryzyka zawiera w sobie jeszcze kolejny krok: podjęcie decyzji o ryzyku (w innych przypadkach jest to ujmowane w ramach nadzorzonego procesu, czyli zarządzeniem ryzykiem).

5. Krok piąty: podjęcie decyzji o ryzyku

Większość metodologii mówi o czterech możliwych, wzajemnie wykluczających się podejściach dotyczących ryzyka, które zostały zobrazowane w dolnej części diagramu, a omówione poniżej.

1. Redukcja ryzyka

Może być realizowana na kilka sposobów, ale najczęściej jest to implementacja odpowiednich korekt w naszym systemie (np. wykonanie aktualizacji nieaktualnego oprogramowania, załatanie podatności w aplikacjach). Przykładowe zestawienie metod ochrony – **Sans 20 Critical Security Controls**.

2. Akceptacja ryzyka

W przypadku kiedy ryzyko jest małe, a potencjalna redukcja ryzyka kosztowna, możemy chcieć je zaakceptować. Jednak kryteria akceptacji ryzyka powinny być odpowiednio określone (np. w odpowiedniej polityce bezpieczeństwa). Nie chodzi tutaj o przypadek, kiedy nie realizujemy analizy ryzyka, a wszystkie (tj. również nieznane nam) ryzyka akceptujemy.

3. Przekazanie ryzyka

Może przybrać formę ubezpieczenia od ryzyka (postać znana np. w przypadku ubezpieczenia nieruchomości od zdarzeń losowych; dla zagrożeń IT rzadko w Polsce stosowana), ale również **outsourcingu**. Czyli ryzyko przekazuję do innej firmy, a odpowiedzialność za nie zawieram w odpowiednich zapisach umownych.

4. Uniknięcie ryzyka

Jeśli ryzyko jest duże, a system, w którym ono występuje, nie przynosi mi odpowiednich korzyści, być może najkorzystniejszym rozwiązaniem będzie w ogóle zrezygnowanie z eksploatacji danego systemu (a zarazem uniknięcie ryzyka).

19. Jakie możliwe działania można podjąć odnośnie ryzyka typu szansa, a jakie odnośnie ryzyka typu zagrożenia w projekcie programistycznym?

<https://omec.pl/blog/ryzyko-zagrozenia-i-szanse/>

Większości z nas ryzyko kojarzy się z czymś negatywnym. Według Słownika Języka Polskiego ryzyko to:

- „możliwość, że coś się nie uda; też: przedsięwzięcie, którego wynik jest niepewny,
- prawdopodobieństwo powstania szkody obciążające osobę poszkodowaną niezależnie od jej winy, jeśli umowa lub przepis prawný nie zobowiązały innej osoby do wyrównania szkody”[1].

W przypadku zarządzania projektami jest inaczej. Ryzyko może oznaczać zarówno zagrożenie, jak i szansę. Definicja ryzyka projektu to „niepewne zdarzenie lub zbiór zdarzeń, które w przypadku ich wystąpienia będą mieć wpływ na osiągnięcie celów. Miarą ryzyka jest iloczyn prawdopodobieństwa wystąpienia dostrzezonego zagrożenia lub szansy oraz wielkości jego/jej wpływu na cele”.[2] Przykładem obrazującym ryzyko, jako zagrożenie i szansę, może być ryzyko kursowe. Założmy, że jesteśmy importerem jakiegoś dobra i płacimy za nie walutą X. Jeżeli kurs tej waluty osłabi się w stosunku do waluty w jakiej musimy zapłacić za importowane dobro, to koszty importu zwiększą się lub będziemy zmuszeni importować mniej danego dobra. Z drugiej jednak strony, jeżeli kurs się umocni koszty zmniejszą się lub będziemy mogli importować więcej danego dobra. Istnieje kilka rodzajów reakcji na ryzyko z podziałem na zagrożenia i szanse.

zagrożenie	szansa	
unikanie	wykorzystanie	
redukowanie	wzmocnienie	
plan rezerwowy		
przeniesienie		
współdziałanie		
akceptowanie	odrzucanie	

20. Jaka jeszcze inna perspektywa oprócz perspektywy specyfikacyjnej i implementacyjnej dotyczy modelowania rozwiązania z wykorzystaniem języka UML? Na czym ona polega?

http://smurf.mimuw.edu.pl/external_slides/UML_cz%20I/UML_cz_I.html

Diagramy wdrożenia UML

Diagramy wdrożenia prezentują rozmieszczenie fizycznych składników systemu (artefaktów) i fizycznych elementów realizujących system (węzłów).

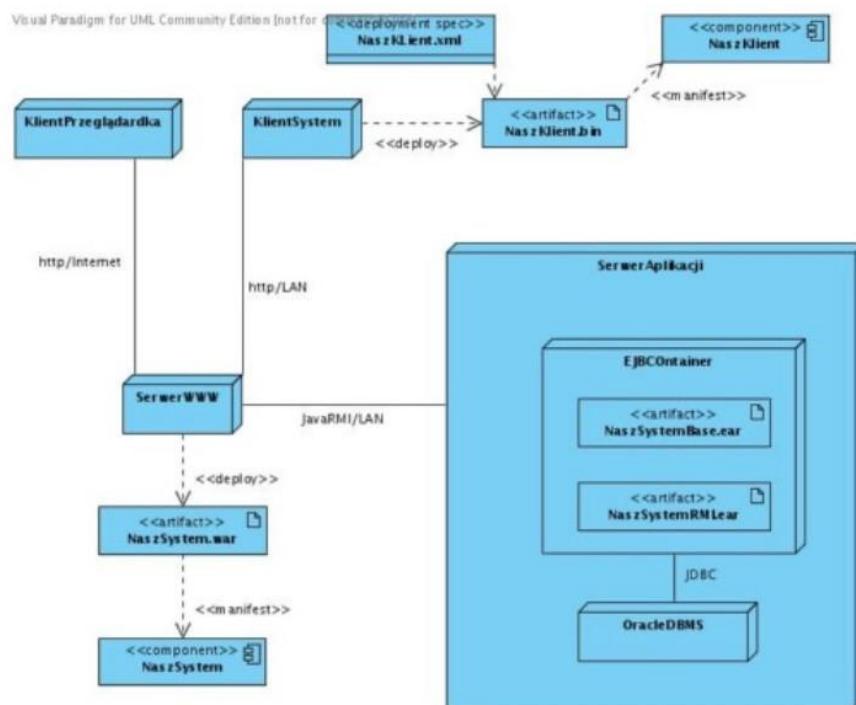
Węzły (nodes) dzielą się na:

- urządzenia (devices) – komputery, czujniki, sprzęt sieciowy itp.
- środowiska realizacji (execution environments) – systemy operacyjne, serwery aplikacji itp.

Artefakty (artifacts) dzielą się na:

- pliki wykonywalne
- pliki konfiguracyjne, z danymi itp.

W węzłach realizuje się oprogramowanie stanowiące system. Węzły połączone są liniami komunikacyjnymi.



SZTUCZNA INTELIGENCJA

1. Podaj rzeczywiste przykłady zastosowania metod sztucznej inteligencji

<https://automatykaonline.pl/Artykuly/Sterowanie/zastosowanie-metod-sztucznej-inteligencji-w-energetyce>

<https://productvision.pl/2017/zastosowanie-metod-sztucznej-inteligencji-w-projektowaniu-doswiadczen-klientow/>

2. Omów budowę sztucznych sieci neuronowych. Czym się różni sieć neuronowa w architekturze płytkiej od głębokiej.

<https://bulldogjob.pl/articles/1136-czym-jest-deep-learning-i-sieci-neuronowe>

Architektura głębokiej sieci neuronowej użytej do ekstrakcji modelu

Jak podaje [3], płytka sztuczna sieć neuronowa zbudowana z jednej warstwy, ale z odpowiednio dużą liczbą neuronów byłaby, w teorii, w stanie znaleźć każdą funkcję wyznaczoną z danych. To samo źródło sugeruje też, że pogłębienie sieci o większą liczbą warstw jest opłacalną alternatywą (choć nie bez swoich własnych problemów).

Jako architekturę wykorzystaną do ekstrakcji modelu użyto wielowarstwową sztuczną sieć neuronową zbudowaną z 4 ukrytych warstw po 512 neuronów każda, oraz ReLU w roli funkcji aktywacji. Wybór taki zmotywowany był faktem, że klasyfikator oparty o głębokie uczenie osiągnął najlepsze wyniki w ekstrakcji modelu z systemów opartych o SVM oraz Naïve Bayes w [132].

Architektura została dobrana tak, by być bardziej złożona niż pierwotny klasyfikator, biorąc pod uwagę typową liczbę cech zawartą w zbiorach danych w dziedzinie cyberbezpieczeństwa.

Tak, jak zilustrowano w [132], algorytm ekstrakcji modelu można podsumować w niniejszy sposób:

3. Czy jest predykcja. W jakim celu się ją stosuje?

<http://www.outsourcingportal.eu/pl/sztuczna-inteligencja-i-modele-predykcyjne-w-rozwiazaniach-red-hat>

<https://www.psi.pl/pl/blog/psi-polska-blog/post/prognozowanie-zuzycia-energii-z-wykorzystaniem-sztucznej-inteligencji/>

4. Gdzie stosujemy uczenie nadzorowane, a gdzie nienadzorowane – opisz oraz wymień typy znanego oprogramowania

<https://www.statystyczny.pl/co-to-jest-machine-learning/>

<https://course.elementsofai.com/pl/4/1>

Uczenie nadzorowane (*ang. supervised learning*) to takie, kiedy zbiór danych dostarczany maszynie do nauki zawiera również oczekwaną odpowiedź. Na przykład zdjęcia różnych kwiatków, a do tego nazwa każdego z nich. Albo zestaw maili z informacją, który z nich to spam a który nie. Dzięki takiemu nauczaniu oczekujemy, że po pokazaniu zdjęcia, którego wcześniej nie było w zbiorze danych, dowiemy się, jaki jest to kwiatek (wcześniej musiały być inne zdjęcia tego kwiatka, żeby komputer miał się skąd tego nauczyć). A nowy mail trafi albo do skrzynki odbiorczej albo do spamu.

Uczenie nienadzorowane (*ang. unsupervised learning*) to takie, kiedy nie dostarczamy żadnych odpowiedzi, tylko zestaw danych. Na przykład dostarczamy zdjęcia różnych kwiatków, ale nie mamy na ich temat żadnych więcej informacji. Oczekujemy, że zostaną podzielone na jakieś grupy i każde nowe zdjęcie trafi do grupy, gdzie znajdują się kwiatki do niego podobne. Co to znaczy „podobne”? Wybór należy do maszyny uczącej się. Zwykle na początku podaje się informację, na ile grup byśmy chcieli podzielić nasze dane.

5. Jaka jest różnica między klasyfikacją a regresją – oraz gdzie są wykorzystywane, podaj znane ci rozwiązańia.

https://ai.ia.agh.edu.pl/_media/pl/dydaktyka:mbn:uczenie_maszynowe.pdf

<https://pl.gadget-info.com/difference-between-classification>

<https://www.bloginnovazione.it/pl/uczenie-maszynowe/3716/>

Algorytmy klasyfikacyjne – to takie algorytmy, które pozwalają przypisać dane do odpowiednich kategorii. Przykład najbardziej znany, to podział maili na spam i nie-spam. Oprócz tego rozpoznawanie kwiatków po wyglądzie albo ręcznie napisanych cyferek. Jeśli przypisujemy dane do dwóch kategorii, to mamy do czynienia z klasyfikacją dwuklasową (ang. *two-class classification*). Jeśli etykiet jest więcej, to mówimy o klasyfikacji wieloklasowej (ang. *multiclass classification*).

Algorytmy regresyjne – to zupełnie jak w tekście o **regresji liniowej**. Mamy dane wejściowe (np. wielkość czekolady, zawartość kakao, producent itp.), a oczekujemy, że algorytm pomoże nam przewidywać cenę takiej czekolady (oczekujemy tu wartości ciągłych, a nie dyskretnych, w przeciwieństwie do klasyfikacji).

Definicja klasyfikacji

Klasyfikacja to proces znajdowania lub odkrywania modelu (funkcji), który pomaga w rozdzielaniu danych na wiele klas jakościowych. W klasyfikacji identyfikowane jest członkostwo grupy w problemie, co oznacza, że dane są kategoryzowane pod różnymi etykietami według niektórych parametrów, a następnie etykiety są przewidywane dla danych.

Modele pochodne można wykazać w postaci reguł "IF-THEN", drzew decyzyjnych lub sieci neuronowych itp. **Drzewo decyzyjne** jest zasadniczo schematem, który przypomina strukturę drzewa, gdzie każdy węzeł wewnętrzny przedstawia test na atrybutie, a jego gałęzie pokazują wynik testu. Proces klasyfikacji dotyczy problemów, w których dane można podzielić na dwie lub więcej etykiet dyskretnych, innymi słowy, dwa lub więcej zbiorów rozłącznych.

Weźmy **przykład**, przypuśćmy, że chcemy przewidzieć możliwość wystąpienia opadów w niektórych regionach na podstawie pewnych parametrów. Wtedy będą dwie etykiety deszczu i bez deszczu, pod którymi można zaklasyfikować różne regiony.

Definicja regresji

Regresja jest procesem znajdowania modelu lub funkcji do rozróżniania danych w ciągle rzeczywiste wartości zamiast używania klas. Matematycznie, z problemem regresji, próbuje się znaleźć aproksymację funkcji z minimalnym odchyleniem błędu. W regresji przewiduje się, że zależność numeryczna danych będzie ją rozróżniać.

Analiza regresji jest modelem statystycznym używanym do przewidywania danych numerycznych zamiast etykiet. Może również identyfikować ruch dystrybucji w zależności od dostępnych danych lub danych historycznych.

Weźmy również podobny **przykład** w regresji, gdzie za pomocą niektórych parametrów znajdujemy możliwość wystąpienia opadów w niektórych regionach. W tym przypadku istnieje prawdopodobieństwo związane z deszczem. Tutaj nie klasyfikujemy regionów w czasie deszczu i bez etykiet przeciwdeszczowych, a zamiast tego klasyfikujemy je z powiązanym prawdopodobieństwem.

Kluczowe różnice między klasyfikacją a regresją

- Proces klasyfikacji modeluje funkcję, za pośrednictwem której dane są przewidywane w dyskretnych etykietach klas. Z drugiej strony regresja jest procesem tworzenia modelu przewidującego ciągłą ilość.
- Algorytmy klasyfikacji obejmują drzewko decyzyjne, regresję logistyczną, itp. Dla kontrastu, drzewo regresji (np. Losowy las) i regresja liniowa są przykładami algorytmów regresji.
- Klasyfikacja przewiduje nieuporządkowane dane, a regresja przewiduje uporządkowane dane.
- Regresję można oszacować za pomocą błędu średniej kwadratowej. Wręcz przeciwnie, klasyfikacja jest oceniana na podstawie dokładności pomiaru.

6. Podaj wady i zalety rozwiązań opartych o sztuczną inteligencję w znanych ci rozwiązaaniach. <https://pl.techbriefly.com/jakie-sa-wady-i-zalety-sztucznej-inteligencji-tech-17836/>

Zalety sztucznej inteligencji

1. Błędy są zminimalizowane

Po pierwsze, dzięki sztucznej inteligencji błędy ludzkie są redukowane. Maszyny są dokładniejsze i mniej podatne na błędy zewnętrzne.

2. AI wspiera wiele sektorów i branż

Mожет быть stosowany w medycynie, lotnictwie, transporcie itp.

3. Sztuczna inteligencja optymalizuje do maksimum

W taki sposób, aby maszyna w pełni wykorzystywała wydajność, będąc w stanie wykonywać kilka zadań w tym samym czasie, pomijając zbędne funkcje. Na przykład samochody autonomiczne.

Z drugiej strony, skoro już rozmawialiśmy o wszystkich zaletach sztucznej inteligencji, zobaczymy, jakie są główne zagrożenia.

Wady sztucznej inteligencji

1. Wzrost bezrobocia

Pamiętaj, że jeśli maszyny są w stanie wykonywać te same zadania co ludzie, spowoduje to wzrost bezrobocia.

2. Ma wysoki koszt

Inwestowanie w sztuczną inteligencję wymaga dużych nakładów, dlatego to największe organizacje jako pierwsze wdrażają ją w swoich procesach i zarządzaniu.

3. W końcu brakuje mu uczuć i wartości

Choć technologia ta jest skuteczna, nie jest istotą ludzką i brakuje jej uczuć. Tak więc, jak wspomnieliśmy wcześniej, nie ma ograniczeń i nie zna bariery moralnej. Okoliczność, w której jeśli nie jest hamowana, może być bardzo niebezpieczna.

7. Co to jest walidacja? Jakie znasz metody walidacji?

<https://kwalifikacje.edu.pl/poznaj-metody-walidacji/>

Symulacja

Stosowana jest w sytuacjach, w których zastosowanie obserwacji byłoby niewystarczające. Jest wiarygodną metodą pomocną także dla zbadania odporności na stres osoby podlegającej walidacji czy zdolności szybkiego reagowania w sytuacji nietypowej lub kryzysowej. Do jej stosowania mogą natomiast zniechęcić wysokie koszta i nakład pracy związane z przeprowadzeniem symulacji.

Obserwacja

Umożliwia sprawdzenie specyficznych kompetencji i umiejętności pracownika, których nie sposób zweryfikować za pomocą innych narzędzi. Odbywa się najczęściej w miejscu pracy i polega na wykonywaniu danych zadań przez osobę podchodzącą do walidacji. Obserwacja jest wysoce wiarygodną metodą i umożliwia weryfikację wielu efektów uczenia się naraz. Jednocześnie pozwala osobie walidowanej wczuć się w sytuację codziennej pracy bez stresu wywołanego nienaturalnego sztuczną sytuacją walidacyjną. Wadami tej metody są jej czasochłonność i niemożność całkowicie obiektywnej oceny umiejętności pracownika, gdyż wiele czynników może wpływać na wynik podczas obserwacji .

8. Podaj rzeczywiste przykłady zastosowania metod sztucznej inteligencji

<https://automatykaonline.pl/Artykuly/Sterowanie/zastosowanie-metod-sztucznej-inteligencji-w-energetyce>

<https://www.controlengineering.pl/piec-glownych-obszarow-zastosowania-sztucznej-inteligencji/>

- Przewidywanie
- Optymalizacja
- Modelowanie

9. Przygotowujesz oprogramowanie, którego jednym celów będzie rozpoznawanie twarzy użytkownika. Zaproponuj poszczególne kroki budowy takiego modułu.

<https://bulldogjob.pl/articles/1136-czym-jest-deep-learning-i-sieci-neuronowe>

<https://www.sztucznainteligencja.org.pl/glebokie-uczenie-i-sieci-neuronowe-w-rozpoznawaniu-obrazu-tylko-po-co/>

<https://pclab.pl/art79689.html>

<https://aigeekprogrammer.com/pl/konwolucyjne-sieci-neuronowe-klasyfikacja-obrazow-czesc-1/>

BEZPIECZEŃSTWO INFORMACJI

1. Wymień i porównaj szyfry podstawieniowe i przestawieniowe.

<http://home.agh.edu.pl/~horzyk/lectures/bdk/BDK-KryptografiaKryptoanaliza.pdf>

Proste szyfry stosowane były od dawna:

- ✓ **Szyfry przestawieniowe** - polegające na przestawieniu znaków w tekście,
np. Szyfr Atbasha lub Szyfr Cezara:

Alfabet jawny - A B C D E F G H I J K L M N O P R S T U V W X Y Z
↓ ↓
Alfabet tajny - D E F G H I J K L M N O P R S T U V W X Y Z A B C
- ✓ **Szyfry podstawieniowe** – polegające na podstawieniu znaku lub sekwencji znaków w miejscu szyfrowanego znaku lub sekwencji znaków, np.
szyfry stosowane w trakcie 1. i 2. wojny światowej przez wojska niemieckie, tj.
szyfr ADFGX czy ADFGVX, polegające na zastąpieniu oryginalnych/szyfrowanych znaków przez dwuznak składający się z dwóch liter łatwych do przesłania kodem Morsa, kodujących litery oraz cyfry:

Dzięki pracy francuskiego kryptologa Painvina udało się uratować Paryż przed zmasowanym atakiem wojsk niemieckich i doprowadzić szybciej do zakończenia 1. wojny światowej.

	A	D	F	G	V	X
A	c	o	8	x	f	4
D	m	k	3	a	z	9
F	n	w	1	0	j	d
G	5	s	i	y	h	u
V	p	1	v	b	6	r
X	e	q	7	t	2	g

2. Wymień i porównaj szyfry symetryczne i asymetryczne.

<http://home.agh.edu.pl/~horzyk/lectures/bdk/BDK-KryptografiaKryptoanaliza.pdf>

<https://academy.binance.com/pl/articles/symmetric-vs-asymmetric-encryption>

<https://www.slawop.net/blog/szyfrowanie-symetryczne-i-asymetryczne>

Szyfrowanie symetryczne vs asymetryczne

Algorytmy szyfrowania podobnie jak kryptografia dzielą się na dwie kategorie: algorytmy szyfrowania symetrycznego i algorytmy szyfrowania asymetrycznego. Podstawowa różnica między tymi dwiema metodami szyfrowania polega na tym, że algorytmy szyfrowania symetrycznego opierają się na pojedynczym klucz, podczas gdy szyfrowanie asymetryczne polega na wykorzystaniu dwóch różnych, ale powiązanych ze sobą kluczy. Takie rozróżnienie, choć pozornie proste, najlepiej wyjaśnia różnice funkcjonalne występujące między tymi dwiema formami szyfrowania oraz sposobem ich wykorzystania.

Szyfrowanie symetryczne

Ze względu na większą szybkość, szyfrowanie symetryczne jest szeroko stosowane do ochrony informacji w wielu nowoczesnych systemach komputerowych. Najbardziej znanymi przykładami są m.in Advanced Encryption Standard (AES) wykorzystywany przez rząd Stanów Zjednoczonych do szyfrowania poufnych i tajnych informacji. AES zastąpił wcześniejszy standard szyfrowania danych (DES), który został opracowany w latach siedemdziesiątych jako jeden ze standardów szyfrowania symetrycznego.

Szyfrowanie asymetryczne

Szyfrowanie asymetryczne z kolei jest wykorzystywane w systemach, w których wielu użytkowników może wymagać dostępu do zarówno szyfrowania, jak i deszyfrowania wiadomości lub zestawu danych. Jednym z przykładów takiego systemu jest zaszyfrowana wiadomość e-mail, w której klucz publiczny może być użyty do zaszyfrowania wiadomości, a klucz prywatny może zostać użyty do jej odszyfrowania.

3. Wykonujesz testy penetracyjne – przedstaw charakterystykę poszczególnych etapów.

<https://shinsec.pl/test-penetracyjny/>

<https://sekurak.pl/czego-ucza-metodologie-testow-penetracyjnych-cz-1/>

Metodologia

Początkowym etapem jest ustalenie zakresu prac i metod przeprowadzania testu penetracyjnego.

Wyróżniamy trzy możliwe metody:

- **white-box** – osoba przeprowadzająca pentest posiada pełną dokumentację systemu bądź aplikacji. Najczęściej test ten ma na celu sprawdzenie czy system spełnia określone procedury bezpieczeństwa, bądź normy. Metoda ta zajmuje najmniej czasu i jest najtańsza.
Przeprowadzany jest on przy pełnej wiedzy zespołu i zarządu.
- **black-box** – pentester (osoba przeprowadzająca testy penetracyjne) nie wie nic o systemie, o dostępnych usługach ani infrastrukturze. Jest to najbardziej czasochłonna i kosztowna metoda.
Wymaga od pentestera ogromnej wiedzy i umiejętności przełamywania zabezpieczeń.
Przeznacza on największy okres czasu na fazę rekonesansu.
- **grey-box** – połączenie metod białego pudełka i czarnego. W tym przypadku otrzymujemy część informacji o celu. Nie zawsze jest to pełna dokumentacja, ale często są to bardzo cenne wskazówki.

Fazy

Pre-engagement Interactions

Faza wstępna czyli kontakt ze zleceniodawcą mający na celu ustalenie szczegółów testu penetracyjnego. Na tym etapie ustala się sposób przeprowadzania testów, narzędzia jakich pentester będzie mógł użyć, zakres, które systemy podlegają analizie. Należy ustalić, które systemy nie powinny być testowane ze względu na ich wrażliwość. Ważne jest ustalenie, w którym momencie test się kończy.

Intelligence Gathering

Zbieranie informacji na temat celu ataku. W tej fazie wykorzystujemy biały wywiad, czyli szukamy dostępnych informacji na portalach społecznościowych, korzystamy ze skanerów portów, footprintingu, Google hackingu. Staramy się dowiedzieć jakie usługi są uruchomione, które porty są otwarte. Powinniśmy poznać listę dostępnych hostów, jakie aplikacje są uruchomione itd. Im więcej informacji zbierzmy tym większe szanse na wejście do systemu.

Threat Modeling

W fazie modelowania zagrożeń wykorzystujemy dane, które zgromadziliśmy w poprzedniej fazie. Przyjmujemy pozycję atakującego i staramy się myśleć jak on. Opracowujemy strategię w taki sam sposób, jaki zrobiłby to intruz.

Vulnerability Analysis

Analiza podatności polega na zdefiniowaniu, które ataki są możliwe. Wyszukujemy błędy w usługach, aplikacjach i hostach, które zbadaliśmy wcześniej. Na tym etapie korzystamy ze skanerów podatności. Dobieramy exploity, które mają największe szanse na zadziałanie.

Exploitation

Eksplotacja, czyli realna penetracja systemu. Wykorzystujemy dobrane wcześniej exploit i luki, aby dostać się do systemu. Należy pamiętać, że ślepe przeprowadzanie masowych ataków mija się z celem i może tylko spowodować awarię – takie rozwiązanie niesie niewielkie korzyści dla właściciela.

Post Exploitation

Faza poeksplatacyjna zaczyna się po uzyskaniu dostępu do systemu. Przechodzimy z systemu do systemu i zdobywamy coraz więcej informacji. Pentester może uzyskać trwałego dostępu do systemu wykorzystując tylne drzwi (backdoor), spróbować podnieść uprawnienia (privilege escalation). Staramy się złamać zabezpieczenia kolejnych wewnętrznych systemów, aplikacji, infrastruktur. Należy się dostosować, polegać na własnym rozumie i inteligencji, a nie zautomatyzowanych rozwiązań.

Reporting

Test penetracyjny kończymy stworzeniem raportu. Musimy zawrzeć w nim czynności, które przeprowadziliśmy, szczegółową listę luk. Wszystkie informacje, jakie udało nam się zdobyć należy umieścić w dokumentacji. Należy podzielić raport na dwie części – streszczenie dla zarządu i część techniczną. W streszczeniu powinniśmy zawrzeć ogólne zalecenia, ogólne wyniki przeprowadzonych działań i krótki opis każdej z faz testu. W części technicznej należy umieścić szczegółowy opis każdej z faz i oszacować ryzyko związane z każdą podatnością. Pamiętać należy, aby nie oznaczać luk teoretycznych jako krytycznych, jeśli nie stworzono jeszcze odpowiedniego exploitu. Dzięki temu zleceniodawca wie, na których podatnościach powinien się skupić w pierwszej kolejności. W części technicznej ważne jest zamieszczenie rozwiązań, które pomogą wyeliminować ryzyko związane z lukami. Należy pamiętać, aby upewnić się, że luki to naprawdę luki, a nie bazować tylko na wyniku ze skanerów.

4. Ataki Dos/DDoS – scharakteryzuj rodzaje i metody przeprowadzania ataku.

<https://dataspace.pl/blog/dos-rodzaje-atakow-cz-1/>

5. Omów zasadę przeprowadzania ataków *Man in the middle* (MITM), jak się przed tym chronić.

<https://tyrantsthem.com/pl/artykuly/man-in-the-middle/>

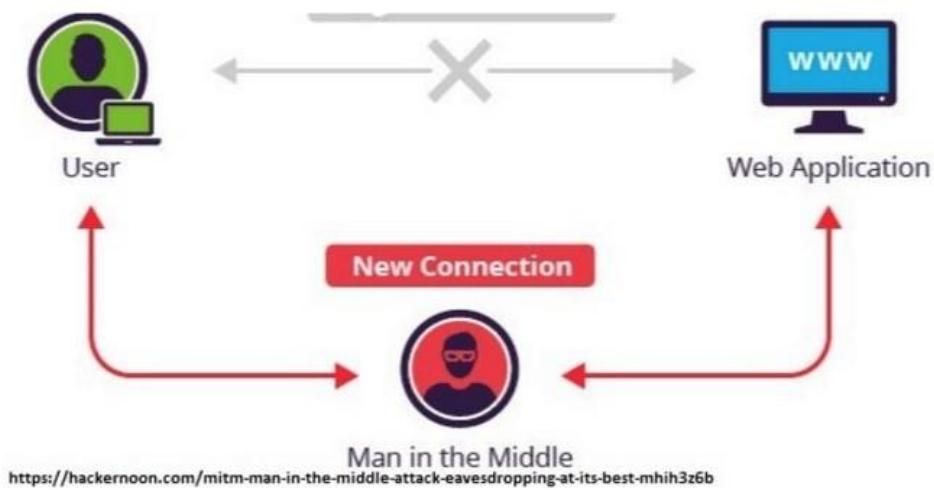
<https://geek.justjoin.it/atak-man-in-the-middle-na-czym-polega-i-jak-sie-przed-nim-bronic>

<https://itfocus.pl/dzial-it/bezpieczenstwo/ataki-man-in-the-middle/>

Definicja ataku MITM

Atak MITM – (ang. Man in the Middle) to nic innego, jak prosty atak sieciowy, którego zamysłem jest **pod słuchiwanie wymiany danych pomiędzy stronami komunikacji** (np. rozmowy Kasi i Tomka) lub ewentualnej jej modyfikacji. Wyróżnia go brak świadomości użytkownika o podsłuchu oraz to, że informacje, które otrzymuje lub, które wysyła mogą być zmodyfikowane przez cyberprzestępca.

Atak Man in the Middle jest atakiem z grupy **ataków pasywnych**, ponieważ polega głównie na podsłuchiwaniu. W chwili, gdy cyberprzestępca dokonuje modyfikacji w komunikacji, przekierowuje ruch na inne strony internetowe czy serwery, wtedy zmienia charakterystykę w atak z grupy ataków aktywnych, czyli mających na celu modyfikację danych lub tworzenie nowych danych.



Rysunek 1 Topologia ataku MITM

Wyżej wymieniony scenariusz ataku **nie jest jedynym zastosowaniem ataku MITM**. Innym scenariuszem jest podsunięcie autorskiego klucza szyfrującego komunikację, przez cyberprzestępca. Haker tworzy dwa klucze, jeden do rozpoczęcia komunikacji oraz do odszyfrowania odpowiedzi. W kolejnym kroku szyfruje wiadomość drugim utworzonym przez niego kluczem, który wysyła do odbiorcy. Dzięki tak sprytnemu zastosowaniu cała komunikacja przechodzi przez cyberprzestępca a efektem tego jest **całkowity wgląd do danych** przesyłanych pomiędzy stronami komunikacji – wrażliwe dane, dostęp do prywatnych zasobów.

Warto wspomnieć, że tego rodzaju atak Man-in-the-middle **może, lecz nie musi** polegać na usuwaniu szyfrowanego połączenia pomiędzy stronami komunikacji przez osobę trzecią. Głównym celem ataku jest spowodowanie braku lub możliwości wzajemnego uwierzytelnienia się pomiędzy stronami komunikacji, co skutkuje także możliwością uzyskania autoryzowanego dostępu przez osobę trzecią.

Atak typu Man in the Middle dzieli się na kilka mniejszych:

Rodzaje ataków Man in the Middle

- najpowszechniejszym z nich jest sniffing, czyli podsłuchiwanie ruchu w sieci Wi-Fi,
- DNS Spoofing, mający na celu przekierowaniu użytkownika na fałszywą stronę aby pozyskać danych logowania,
- ARP Poisoning, czyli przeciążenie przełącznika pakietami ARP.

Co możemy zrobić, by chronić się przed atakami MitM?

- Przede wszystkim należy unikać publicznych punktów dostępu WiFi — szczególnie gdy nie są chronione hasłem. Jeśli jesteś zmuszony do korzystania z takiej sieci, rób to jedynie, do pasywnego korzystania z internetu nie korzystając ze stron, które wymagają podawania danych.
- Wyloguj się po zakończeniu korzystania ze strony, która wymaga logowania. Część stron robi to automatycznie wraz z zamknięciem przeglądarki.
- Używaj wieloetapowego uwierzytelniania, jeśli to możliwe. Praktycznie wszystkie strony związane z finansami mają opcję dwuetapowego uwierzytelniania, które staje się standardem również poza branżą finansową.
- Używaj stron korzystających z HTTPS. Upewnij się, że jesteś na stronie z 'kłódką' gdy podajesz jakiekolwiek dane — HTTPS zapewnia szyfrowaną komunikację. Jeśli masz możliwość, zainstaluj plug-in typu HTTPS Everywhere, który wymusza przeglądarkę do korzystania z bezpiecznej wersji strony.
- Używaj VPN (virtual private network) do przeprowadzania transakcji i wrażliwej komunikacji. VPN jest właściwie konieczne w czasie korzystania z publicznego WiFi.
- Skonfiguruj router. Upewnij się, że nie pozostawiłeś domyślnych danych logowania producenta. Upewnij się także, że router jest zaktualizowany.
- Uważaj na phishing w mailach.
- Zainstaluj oprogramowanie antywirusowe — w ten sposób można uniknąć ataków man-in-the-middle, które bazują na zainstalowanym malware.

6. Wymień i scharakteryzuj wybrane narzędzia pentestera.

<https://kapitanhack.pl/2019/05/21/nieskategoryzowane/jak-przeprowadzic-proste-testy-penetracyjne-w-swojej-sieci/>

Narzędzia pentestera warte uwagi:

- **Frameworki:** Kali Linux, Backtrack5 R3, Security Onion
- **Rekonesans:** Smartwhois, MxToolbox, dnsstuff, nslookup, DIG
- **Rozpoznanie:** Angry IP scanner, nmap, NetResident
- **Skanowanie portów:** Nmap, Megaping, zenmap, Advanced port scanner, Hping3
- **Enumeracja:** Superscan, Netbios enumeration, DumpSec, Snmpcheck, Ps Tools, nslookup
- **Wykrywanie podatności:** Nessus, GFI Languard, Retina, Nexpose
- **Łamanie haseł:** Ncrack, John The Ripper, Hashcat, Cain&Abel
- **Exploitacja:** Metasploit, Core Impact

7. Omów na przykładach wybrane ataki socjotechniczne.

<https://niebezpiecznik.pl/post/socjotechnika/>

<https://www.eskom.eu/blog/hakerskie-ataki-socjotechniczne/>

https://www.sans.org/sites/default/files/newsletters/ouch/issues/OUCH-201411_po.pdf

8. Omów zasadę działania oraz porównaj systemy IDS oraz IPS.

<https://students.mimuw.edu.pl/SO/Projekt04-05/temat5-g2/sikora-kobylinski/idsips.html>

<https://repozytorium.ukw.edu.pl/bitstream/handle/item/3532/Ids%20Ips%20systemy%20wykrywania%20i%20zapobiegania%20wlamaniom%20do%20sieci%20komputerowych.pdf?sequence=1&isAllowed=y>

[http://luk.kis.p.lodz.pl/BSS/wyklad/v2013!/BSS.w05.v2013.\(IDS.IPS%20-%20i%20inne\).pdf](http://luk.kis.p.lodz.pl/BSS/wyklad/v2013!/BSS.w05.v2013.(IDS.IPS%20-%20i%20inne).pdf)

Czym są IDS?

IDS (Intrusion Detection Systems) systemy wykrywania intruzów, jak sama nazwa wskazuje, zajmują się wykrywaniem prób uzyskania dostępu do systemu. Ich zadaniem jest wykrycie takiego zdarzenia i poinformowanie o tym odpowiednich osób. Działanie takich systemów jest podobne do alarmu chroniącego dom przed włamywaczami.

IDS często są używane razem z firewallami, nie należy jednak mylić tych pojęć - zapory sieciowe służą wyłącznie do ochrony systemu przed niepowołanymi osobami. IDS są wykorzystywane do uzupełnienia całości dobrze zorganizowanego systemu bezpieczeństwa, na który oprócz nich powinny się składać m.in.:

- polityka bezpieczeństwa
- szyfrowanie danych
- weryfikacja użytkowników
- kontrola dostępu
- zapory sieciowe

Trzy podstawowe zadania IDS to monitorowanie systemu, detekcja ataków i podejmowanie odpowiednich działań w zależności od zagrożenia, jak: wylogowanie użytkowników, zablokowanie konta lub wykonanie odpowiednich skryptów. Często informacja o ataku przesyłana jest natychmiast do upoważnionych osób np. na pager lub poprzez e-mail.

Skąd potrzeba używania IDS?

W rzeczywistości większość zagrożeń dotyczących bezpieczeństwa pochodzi z wewnętrz sieci, w czym należy uwzględnić ataki przeprowadzane przez nielojalnych pracowników firmy. Z zewnątrz grożą głównie ataki typu Denial of Service bądź próby penetracji struktury sieci. Systemy wykrywania intruzów są zatem bardzo użyteczne, gdyż służą one podniesieniu bezpieczeństwa sieci zarówno od wewnętrz jak i od zewnętrz. Dodatkowo systemy IDS mogą służyć do analizy ruchu sieciowego, a więc zareagować na zagrożenie atakiem DoS.

Host-Based IDS (HIDS)

HIDS to najwcześniej zaimplementowane systemy wykrywania intruzów. Systemy tego typu zbierają i analizują dane na komputerze, który jest gospodarzem systemu, na przykład serwerze sieciowym. Gdy odpowiednie dane są zebrane, mogą zostać przeanalizowane na odrębnej maszynie lub w obrębie komputera-gospodarza.

Przykładowo rolę HIDS może spełniać program zbierający informacje z logów systemowych i aplikacyjnych z innych komputerów. Jest to sposób na efektywne wykrywanie nadużyć wewnętrz sieci jeśli jakiś użytkownik wykona niedozwoloną czynność, informacja o tym zdarzeniu bardzo szybko trafi do systemu wykrywania. To rozwiązanie dobrze sobie radzi także m.in. z nieautoryzowaną modyfikacją pliku.

Taki system może jednak stać się niepraktyczny. Przy dużych sieciach zbieranie informacji od każdej maszyny jest posunięciem nieefektywnym i niewygodnym w obsłudze. W dodatku, w przypadku odcięcia serwera HIDS od reszty sieci, zabezpieczenie przestaje działać.

Czym są IPS?

IPS (Intrusion Prevention Systems) to sprzętowe bądź programowe rozwiązania, których zadaniem jest wykrywanie ataków na system komputerowy z wewnętrz jak i od zewnątrz systemu oraz uniemożliwianie przeprowadzenia takich ataków. Od strony technicznej systemy IPS to mniej więcej połączenie zapory sieciowej i systemu IDS. W dalszej części prezentacji omówionych zostanie kilka różnych sposobów implementacji systemów ochrony przed intruzami.

Inline Network-Based IDS

Sieciowe systemy wykrywania intruzów zwykle posiadają skonfigurowane dwie karty sieciowe jedną do wykrywania zagrożeń, drugą do zwykłych zastosowań. Karta do wykrywania nie ma przypisanego adresu IP, działa niewidoczna dla reszty systemu, co znaczy, że nie można skierować do niej pakietu ani poprosić ją o odpowiedź. Cały ruch sieciowy przechodzi przez inline NIDS, który sprawdza, czy w przesyłanych treściach nie można doszukać się podobieństwa do wzorców sygnatur. Są rozwiązania, (HogWash) które idą krok dalej są w stanie zmieniać treść złośliwych pakietów, aby nie mogła działać (packet scrubbing). W ten sposób atakujący nie będzie wiedział, że jego próby są nieudane, a broniący się będzie mógł zebrać więcej dowodów.

To rozwiązanie łączy możliwości NIDS z blokowaniem niepożądanych pakietów przez zaporę sieciową. Jak w wielu systemach NIDS, można monitorować działanie wielu serwerów i sieci z jednej maszyny. Może to być jednak zarówno wielką zaletą jak i wadą, problemy pojawiają się przy awarii systemu wtedy ruch sieciowy nie musi przepływać przez dane urządzenie. Kolejną niedogodnością takich systemów IPS jest to, że można nimi chronić jedynie pewne aplikacje, które są w użyciu (np. Apache, Internet Information Services, ...) oraz to, że nie ma ochrony przed zachowaniami dla których nie zostały zdefiniowane sygnatury. Są miejsca, gdzie omawiane rozwiązanie przyjęło się bardzo dobrze mainframe'y i duże serwery.

9. Czy jest możliwość bycia anonimowym w Internecie?

- <https://www.vpnpolaczenie.pl/security/anonimowe-surfowanie/>
- <https://techporadnik.pl/jak-byc-anonimowym-w-internecie-w-2020-roku/>
- <https://dkdetektyw.pl/anonimowosc-w-sieci/>

<https://trybawaryjny.pl/anonimowosc-w-sieci/>

TECHNOLOGIE INTERNETOWE I MOBILNE

1. Opisz jak przebiega komunikacja komputera z serwerem HTTP podczas próby odczytu pliku *.php

<https://docplayer.pl/6481398-Php-i-komunikacja-klient-serwer-pawel-rajba.html>

<https://webshake.ru/kurs-php-dlya-nachinayushih/kak-rabotaet-php>

2. Omów budowę i zasadę działania formularza HTML5 oraz przedstaw przynajmniej trzy typy pól.

http://www.zsp1.jedrzejow.com.pl/public/download/HTML_formularz.pdf

<https://grafmag.pl/artykuly/formularze-w-html5>

3. Omów różnice między metodami przesyłania danych GET i POST. Podaj przykłady zastosowań obu metod.

<https://pl.gadget-info.com/difference-between-get>

4. Omów na przykładach budowę i sposoby wykorzystania kaskadowych arkuszy stylów CSS.

<http://staff.uz.zgora.pl/amajczak/sieci-komputerowe/CSS.pdf>

http://www.firma.orawskie.pl/?pl_kaskadowe-arkusze-stylow-css,88

<https://docplayer.pl/50107247-Wyklad-2-kaskadowe-arkusze-stylow-css-czesc-1.html>

5. Technologia Ajax – czym jest i w jakich rozwiązańach jest wykorzystywana

<https://kursjs.pl/kurs/ajax/ajax.php>

https://prophp.pl/advice/show/20/pierwsze_kroki_z_ajax._jak_dziala_i_czym_jest%3F

6. Omów zasadę komunikacji aplikacji klienckiej z serwerem bazodanowym na przykładzie PHP i MySQL

<http://student.pwsz.elblag.pl/~stefan/Dydaktyka/2009-2010/SemDiplomowe/Raporty/Kielkowski/2010-04-26/>

7. Przedstaw sposób wykorzystania mechanizmu plików cookies.

<https://wszystkoociasteczkach.pl/po-co-sa-ciasteczka/>

<http://cookieinfo.hostmonster.pl/>

8. Przedstaw sposób wykorzystania mechanizmu zmiennych sesyjnych

<http://www.lomilowka.pl/upload/file/PAI/php6.pdf>

<http://www.beldzio.com/bezpieczenstwo-mechanizmu-sesji>

<http://webmade.org/porady/sesje-php-system-logowania.php>

<http://shebang.pl/kursy/zabezpieczenia-php/r4-sesje/>

9. Omów podstawowe konstrukcje i znaczenie języka XML

<http://webmaster.helion.pl/starocie/xml/xml.htm>

<https://www.samouczekprogramisty.pl/xml-dla-poczatkujacych/>

<https://mfiles.pl/pl/index.php/XML>

10. Scharakteryzuj możliwości języka HTML5 umożliwiające wygodne tworzenie aplikacji graficznych.

<https://blog.eduweb.pl/html5-wszystko-co-musisz-wiedziec/>

<https://tworcastron.pl/blog/html5-naprawde-taki-fajny/>

<https://proseedmag.pl/aktualnosci/zalety-i-wady-gier-html5>

11. Wymień po jednym przykładzie stosowanego na stronach WWW skryptowego języka programowania, wykonywanego a) po stronie klienta b) po stronie serwera.

<https://kobietydokodu.pl/ktory-jazyk-programowania-wybrac/>

<https://pl.gadget-info.com/difference-between-server-side-scripting>

PROGRAMOWANIE

1. W jaki sposób identyfikujemy miejsce przekazania sterowania w inne miejsce w programie w językach interpretowanych, a w jaki w językach kompilowanych?

<https://edu.pjwstk.edu.pl/wyklady/ppj/scb/PrgJav/PrgJav.html>

https://pl.qaz.wiki/wiki/Interpreted_language

https://pl.qaz.wiki/wiki/Compiled_language

Przepływ sterowania [edytuj | edytuj kod]

1 Osobny artykuł: [przepływ sterowania](#).

Do zaprogramowania każdego diagramu przepływu czy każdego automatu o skończonej liczbie stanów bez duplikacji kodu, a więc do sterowania przepływem kontroli w kodzie, którego żadna część się nie powtarza, wystarczą instrukcje skoku warunkowego (poprzedzone instrukcją porównania) i bezwarunkowego.

Obserwacja ta znalazła zastosowanie w konstrukcji **procesorów** i wyrażenia tego w języku **assemblera**, gdzie dostępne są te właśnie instrukcje (instrukcje skoku bezwarunkowego `JMP` i warunkowego `Jxx` w architekturze `x86`, gdzie `xx` symbolizują stan **flag rejestru stanu** zmienianego przez część instrukcji procesora).

Pierwsze języki, przede wszystkim ze względu na nacisk na łatwość konstrukcji **kompilatorów** nie odbiegały znacząco zasobem instrukcji od assemblera, np. w pierwszych wersjach **Fortranu** jedyną instrukcją warunkową był właśnie skok – nie można było warunkowo przypisać wartości czy wykonać grupy poleceń. Z czasem do popularnych języków (wyrosłych na podstawie Fortranu) zaczęto dodawać inne instrukcje kontroli przepływu: wykonania warunkowego (`if (warunek) { polecenie1 } else { polecenie2 }`), różne rodzaje **pętli** (`while`, `for`), rekurencyjność, instrukcje ponownienia (`redo`), **następnej iteracji** (`next` lub `continue`) lub **zakończenia wykonywania** (`last` lub `break`) pętli, te same instrukcje z wielopoziomowymi pętlami (`x: foreach $a(@A) { foreach $b(@B) { ...; if (...) { last X; } } }`), **wyjątki**, **iteratory**, **funkcje wyższego rzędu**, **wątki** itd., co uczyñoło z instrukcją skoku instrukcję na wskroś przestarzałą.

Kompilator tłumaczy program źródłowy na instrukcje, które mogą być wykonane przez procesor i jednocześnie sprawdza składniową poprawność programu, sygnalizując wszelkie błędy. Proces komplikacji jest więc nie tylko procesem tłumaczenia, ale również weryfikacji składniowej poprawności programu.

Zalety i wady

Programy skompilowane do kodu natywnego w czasie komplikacji są zwykle szybsze niż te tłumaczone w czasie wykonywania ze względu na narzucona związana z procesem tłumaczenia. Nowsze technologie, takie jak komplikacja just-in-time i ogólne ulepszenia w procesie tłumaczenia, zaczynają jednak zmniejszać tę lukę. Rozwiązań mieszanych wykorzystujących kod bajtowy mają tendencję do uzyskiwania średniej wydajności.

Języki programowania niskiego poziomu są zwykle komplikowane, zwłaszcza gdy głównym problemem jest wydajność, a nie obsługa wielu platform. W przypadku takich języków istnieje więcej relacji jeden do jednego między zaprogramowanym kodem a operacjami sprzętowymi wykonywanymi przez kod maszynowy, co ułatwia programistom precyzyjne sterowanie wykorzystaniem jednostki centralnej (CPU) i pamięci.

Przy pewnym wysiłku zawsze można napisać kompilatory, nawet dla języków interpretowanych tradycyjnie. Na przykład Common Lisp można skompilować do kodu bajtowego Java (następnie zinterpretowanego przez maszynę wirtualną Java), kodu C (następnie skompilowanego do natywnego kodu maszynowego) lub bezpośrednio do kodu natywnego. Języki programowania, które obsługują wiele celów komplikacji, dają programistom większą kontrolę nad wyborem szybkości wykonywania lub kompatybilności między platformami.

Interpreter wykonuje bezpośrednio tekst programu. Zatem składniowa poprawność jest sprawdzana zazwyczaj dopiero w trakcie działania programu, aczkolwiek niektóre języki interpretowane udostępniają fazę symbolicznej komplikacji do kodu pośredniego, podczas której sprawdzana jest poprawność źródła.

Niektóre interpretery wewnętrznie komplikują fragmenty kodu do postaci binarnej, aby przyspieszyć wykonanie.

Zalety

Funkcje, które są często łatwiejsze do zaimplementowania w interpreterach niż w kompilatorach, obejmują:

- niezależność platformy (na przykład kod bajtowy Javy)
- Odbicie i odblaskowe zastosowanie oceniającego (na przykład pierwszego rzędu eval funkcja)
- dynamiczne pisanie mniejszy rozmiar wykonywalnego programu (ponieważ implementacje mają elastyczność w wyborze kodu instrukcji)
- dynamiczne określanie zakresu

Ponadto kod źródłowy można odczytywać i kopiować, co daje użytkownikom większą swobodę.

Niedogodności

Wady języków tłumaczonych to:

- Bez statycznego sprawdzania typu , które jest zwykle wykonywane przez kompilator, programy mogą być mniej niezawodne, ponieważ sprawdzanie typów eliminuje pewną klasę błędów programistycznych (choć sprawdzanie typu kodu można przeprowadzić przy użyciu dodatkowych samodzielnego narzędzi. Zobacz TypeScript na przykład)
- Tłumacze mogą być podatni na ataki polegające na wstrzykiwaniu kodu .
- Wolniejsze wykonywanie w porównaniu do bezpośredniego wykonywania natywnego kodu maszynowego na procesorze hosta . Techniką używaną do poprawy wydajności jest komplikacja just-in-time, która konwertuje często wykonywane sekwencje interpretowanych instrukcji na kod maszynowy hosta. JIT jest najczęściej łączony z komplikacją do kodu bajtowego, tak jak w Javie .
- Kod źródłowy można czytać i kopiować (np. JavaScript na stronach internetowych) lub łatwiej poddać inżynierii wstępnej poprzez odbicie w aplikacjach, w których własność intelektualna ma przewagę handlową. W niektórych przypadkach zaciemnianie jest stosowane jako częściowa ochrona przed tym.

Java jest językiem interpretowanym, co umożliwia wykonywanie "binarnych" kodów Javy bez rekomplikacji praktycznie na wszystkich platformach systemowych. Kod źródłowy (pliki z rozszerzeniem ".java") jest kompilowany przez kompilator Javy (program javac) do kodu bajtowego (B-kodu, pliki z rozszerzeniem ".class"), ten ostatni zaś jest interpretowany przez tzw. wirtualną maszynę Javy – JVM (jest to program java wraz odpowiednimi dynamicznymi bibliotekami), zainstalowaną na danej platformie systemowej.

2. Wskaż różnice pomiędzy programowaniem imperatywnym a programowaniem funkcyjnym. Które podejście związane jest ze zmianą stanu?

<http://www.braintelligence.pl/roznica-miedzy-programowaniem-funkcyjnym-a-imperatywnym-kotlinowe-przyklady/>

<https://docs.microsoft.com/pl-pl/dotnet/standard/linq/functional-vs-imperative-programming>

Programowanie imperatywne

Najbardziej pierwotnym sposobem programowania komputerem jest programowanie imperatywne. Polega ono na opisaniu programu jako prosty ciąg poleceń zmieniających stan maszyny, na której jest wykonywany. W uproszczeniu mówiąc skupia się więc na tym *w jaki sposób* działa program, a nie jaki jest jego *cel*. Paradygmat ten powiązany jest z komputerami opartymi na architekturze von Neumanna, gdzie program rozumiany jako ciąg instrukcji wpływa na globalny stan maszyny – zawartość jej pamięci, rejestrów i znaczników procesora. Przykładem języka korzystającego z programowania imperatywnego był oryginalny *FORTRAN* czy też *ALGOL*.

Przeciwnieństwem programowania imperatywnego jest **programowanie deklaratywne**, które z definicji skupia się na tym, co program ma osiągnąć, a nie w jaki sposób. Naturalnymi następcami, czy może bardziej rozszerzeniami programowania imperatywnego są:

- **programowanie proceduralne** – zaleca dzielenie kodu na procedury, czyli ciągi instrukcji wykonujące ściśle określone operacje. Powinny one pobierać wszelkie przetwarzane dane jako parametry wywołania i zwracać wynik na ich podstawie.
- **programowanie strukturalne** -polega na dzieleniu kodu na procedury oraz hierarchiczne bloki przy użyciu **instrukcji sterujących (if/else)** i **iteracji (for/while/repeat)**.
- ostatecznie rozwój powyższych doprowadził do powstania **paradygmatu programowania obiektowego**, które zakłada łączenie danych i procedur w *obiekty*, o czym więcej w dalszej części tego artykułu.

Imperatywny kod to ten z którym najczęściej spotykamy się na początku naszej przygody z programowaniem. Jest to najbardziej naturalny sposób w jaki można pisać aplikację. Tworzymy tutaj ciąg instrukcji jaki nasz program wykonuje (step-by-step). Opisujemy dokładne czynności jakie muszą być wykonane podczas działania programu. Podczas tych kroków zmieniamy stan systemu modyfikując go. Wynikiem końcowym jest zwrócona wartość lub inny efekt.

Imperatywny kod cechuje:

- zmieniający się stan podczas każdej iteracji
- kolejność wykonywania jest ważna (step-by-step)
- najczęściej wywołujemy jakieś funkcje, pętle, warunki

Programowanie funkcyjne

Według tego paradygmatu (odmiana programowania deklaratywnego) program traktujemy jako złożoną, matematyczną funkcję, która dla podanych danych wejściowych zwraca konkretny wynik. W językach **czysto funkcyjnych** nie bierzemy pod uwagę stanu maszyny (nie istnieją zmienne, tylko argumenty funkcji), nie występują również efekty uboczne (żaden efekt uruchomienia funkcji nie wykracza poza zwrócenie wartości). Tradycyjnym przedstawicielem takiej grupy jest język *Haskell*.

Najpopularniejszym jednak podejściem stosowanym dzisiaj są języki mieszane, które pozwalają na stosowanie zmiennych, efektów ubocznych czy użycie tradycyjnego wejścia/wyjścia. Mieszają one **styl funkcyjny** z imperatywnym lub obiektowym. Takimi językami są *Lisp*, *Scala*, czy nowoczesny *F#*. W pewnym stopniu funkcyjne podejście możemy stosować w *JavaScript*, czy *Javie*.

Przykład programowania funkcyjnego w języku Scala – sortowanie a'la quicksort (źródło: [Wikipedia](#))

```
1. def qsort(list : List[Int]): List[Int] = list match {
2.   case Nil => Nil
3.   case pivot :: tail => {
4.     val (smaller, rest) = tail partition (_ < pivot)
5.     qsort(smaller) :::: pivot :: qsort(rest)
6.   }
7. }
```

Funkcyjny kod cechuje:

- stan nie istnieje, immutable objects
- kolejność wykonywania nie zawsze jest ważna (często może być asynchroniczna)

Główną różnicą jest tutaj to, że funkcyjne programy są bardziej ekspresywne (czytelne). Piszemy mniej kodu robiąc to samo co w imperatywnym. Ponadto dzięki niemutowalności oraz większej kontroli nad efektami ubocznymi nasze aplikacje są bardziej deterministyczne. Dzięki czemu czasami uciekniemy od wielowątkowych problemów jak race-conditions, deadlocks oraz inne. Ponadto nie zawsze musimy się przejmować się kolejnością wykonywania działań w naszym kodzie. Oczywiście to zależy od konkretnego przypadku, ale koniec końców FP pomaga nam w wielu kwestiach.

Dwa przykłady imperatywnego/funkcyjnego kodu

W podejściu imperatywnym:

Skupiamy się na tym **co chcemy zrobić** (wykonujemy konkretne czynności step-by-step). Tworzymy wynik.

W podejściu funkcyjnym:

Skupiamy się na tym **co chcemy osiągnąć**.

Promujemy możliwe jak najmniejszą ilość efektów ubocznych oraz nie zmieniamy stanu obiektu.

2 Prosty przykład - liczby nieparzyste

Chcemy tylko nieparzyste liczby. Wrzucamy je do listy `odds` (ang. nieparzyste).

```
// Imperatywny przykład
val numbers = listOf(1, 2, 3, 4, 5)
val odds = ArrayList<Int>()

for (index in 0..numbers.lastIndex) {
    val item = numbers[index]
    if (item % 2 != 0) odds.add(item)
}

// Powyższy kod jest zbudowany z wyrażeń.
// Skupiamy się tutaj na tym co robimy/chcemy zrobić.
// Jest to seria mutacji oddzielonych warunkami.
```

```
// Funkcyjny przykład
val numbers = listOf(1, 2, 3, 4, 5)

val odds = numbers.filter { it % 2 != 0 }

// Skupiamy się na tym co chcemy osiągnąć.
// Do tego taki kod jest czytelniejszy.
```

3. Czym jest postać strukturalna kodu źródłowego i co należy zrobić aby można było ją otrzymać w przypadku kiedy mamy do czynienia nawet z programowaniem funkcyjnym?

https://mfiles.pl/pl/index.php/Programowanie_structuralne

<https://devstyle.pl/2018/04/30/jak-i-po-co-pisac-funkcyjnie-w-c-sharp/>

Programowanie strukturalne - jeden z paradygmatów programowania w którym celem jest konstruowanie programu zmierzające do osiągnięcia takiej jego struktury, aby stanowiła ona odzwierciedlenie struktury rozwiązywanego problemu. Kluczowym zagadnieniem w programowaniu strukturalnym jest systematyczne użycie poziomów abstrakcji w ramach opracowywania kolejnych etapów programu. **Proces** tworzenia programu rozpoczyna się od jego ogólnego opisu, przechodząc następnie do szczegółów.^[1] W przeciwieństwie do **programowania obiektowego**, w programowaniu strukturalnym zaleca się wykorzystanie tylko kilku najważniejszych struktur sterujących:

- **Iteracja** - powtarzanie instrukcji w pętli do momentu w którym zostanie spełniony warunek iteracyjny
- Wybór - realizacja poszczególnych instrukcji programu w zależności od jego stanu
- Sekwencja - wykonanie poszczególnych instrukcji w kolejności
- Pętla
- Podprogram
- **Instrukcja** (przypisania, wprowadzania/wyprowadzania danych, warunkowa, wyboru, złożona, procedury)

Ideą programowania strukturalnego jest **wymuszenie** opisania najważniejszych struktur danych, przemyślenie całej struktury przygotowywanego programu oraz utworzenie niezbędnej dokumentacji, co pozwala na wczesne wykrycie problemów, które programista może napotkać w trakcie realizacji projektu.

Przykłady kodu źródłowego

Niemalże wszystkie popularne [języki programowania](#) umożliwiają pisanie programów strukturalnych, jak również znacząca mniejszość z nich zabrania używania skoków. Jednym z popularnych języków w którym można pokazać przykład programu wykorzystującego ideę struktur jest [C++](#). [Prosty program](#) w tym języku wykorzystujący programowanie strukturalne, którego zadaniem jest sumowanie liczb ma postać:

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

int main (){
    int liczbaDodatnia,
    int pierwszaLiczba = 1,
    int i = 0,

    cout << "Podaj liczbę..." << endl,
    cin >> liczbaDodatnia,

    for (int i=0; i < liczbaDodatnia; i++){
        i = pierwszaLiczba + 1,
        cout << i,
    }

    return 0,
}
```

Cechy charakterystyczne [\[edytuj\]](#) [\[edytuj kod\]](#)

Struktury kontrolne [\[edytuj\]](#) [\[edytuj kod\]](#)

Podstawowe struktury kontrolne jakie wymienia [twierdzenie o programowaniu strukturalnym](#), z których możliwe jest zbudowanie dowolnego programu to:

- **Sekwencja** – wykonanie ciągu kolejnych instrukcji
- **Wybór** – w zależności od wartości predykatu wykonywana jest odpowiednia instrukcja. W językach programowania zazwyczaj reprezentowana przez słowa kluczowe `if..then..else`.
- **Iteracja** – wykonywanie instrukcji póki spełniony jest jakiś warunek. Reprezentowana w różnych wariantach jako [pętle](#) oznaczane między innymi przez: `while`, `repeat`, `for` lub `do..until`.

Każda ze struktur kontrolnych może być też rozumiana jako pojedyncza instrukcja. Według pierwotnych założeń struktury kontrolne powinny posiadać jedno wejście i jedno wyjście.

4. Wskaż sprawdzalną metodę zapewniania skalowalności kodu źródłowego tworzonego z zastosowaniem paradymatu programowania obiektowego.

<https://medium.com/webbdev/solid-4ffc018077da>

Инкапсуляция, абстракция, наследование, полиморфизм



Инкапсуляция — объект независим: каждый объект устроен так, что нужные для него данные живут внутри этого объекта, а не где-то снаружи в программе. Например, если у меня есть объект «Пользователь», то у меня в нём будут все данные о пользователе: и имя, и адрес, и всё остальное. И в нём же будут методы «Проверить адрес» или «Подписать на рассылку».

Абстракция — у объекта есть «интерфейс»: у объекта есть методы и свойства, к которым мы можем обратиться извне этого объекта. Так же, как мы можем нажать кнопку на блендере. У блендера есть много всего внутри, что заставляет его работать, но на главной панели есть только кнопка. Вот эта кнопка и есть абстрактный интерфейс.

В программе мы можем сказать: «Удалить пользователя». На языке ООП это будет «пользователь.удалить()» — то есть мы обращаемся к объекту «пользователь» и вызываем метод «удалить». Кайф в том, что нам не так важно, как именно будет происходить удаление: ООП позволяет нам не думать об этом в момент обращения.

Например, над магазином работают два программиста: один пишет модуль заказа, а второй — модуль доставки. У первого в объекте «заказ» есть метод «отменить». И вот второму нужно из-за доставки отменить заказ. И он спокойно пишет: «заказ.отменить()». Ему неважно, как другой программист будет реализовывать отмену: какие он отправит письма, что запишет в базу данных, какие выведет предупреждения.

Наследование — способность к копированию. ООП позволяет создавать много объектов по образу и подобию другого объекта. Это позволяет не копипастить код по двести раз, а один раз нормально написать и потом много раз использовать.

Например, у вас может быть некий идеальный объект «Пользователь»: в нём вы прописываете всё, что может происходить с пользователем. У вас могут быть свойства: имя, возраст, адрес, номер карты. И могут быть методы «Дать скидку», «Проверить заказ», «Найти заказы», «Позвонить».

На основе этого идеального пользователя вы можете создать реального «Покупателя Ивана». У него при создании будут все свойства и методы, которые вы задали у идеального покупателя, плюс могут быть какие-то свои, если захотите.

Идеальные объекты программисты называют классами.

Полиморфизм — единый язык общения. В ООП важно, чтобы все объекты общались друг с другом на понятном им языке. И если у разных объектов есть метод «Удалить», то он должен делать именно это и писаться везде одинаково. Нельзя, чтобы у одного объекта это было «Удалить», а у другого «Стереть».

При этом внутри объекта методы могут быть реализованы по-разному. Например, удалить товар — это выдать предупреждение, а потом пометить товар в базе данных как удалённый. А удалить пользователя — это отменить его покупки, отписать от рассылки и заархивировать историю его покупок. События разные, но для программиста это неважно. У него просто есть метод «Удалить()», и он ему доверяет.

Плюсы и минусы ООП

У объектно-ориентированного программирования много плюсов, и именно поэтому этот подход использует большинство современных программистов.

- ① Визуально код становится проще, и его легче читать. Когда всё разбито на объекты и у них есть понятный набор правил, можно сразу понять, за что отвечает каждый объект и из чего он состоит.
- ② Меньше одинакового кода. Если в обычном программировании одна функция считает повторяющиеся символы в одномерном массиве, а другая — в двумерном, то у них большая часть кода будет одинаковой. В ООП это решается наследованием.
- ③ Сложные программы пишутся проще. Каждую большую программу можно разложить на несколько блоков, сделать им минимальное наполнение, а потом раз за разом подробно наполнить каждый блок.
- ④ Увеличивается скорость написания. На старте можно быстро создать нужные компоненты внутри программы, чтобы получить минимально работающий прототип.

А теперь про минусы:

- ① Сложно понять и начать работать. Подход ООП намного сложнее обычного процедурного программирования — нужно знать много теории, прежде чем будет написана хоть одна строчка кода.
- ② Требует больше памяти. Объекты в ООП состоят из данных, интерфейсов, методов и много другого, а это занимает намного больше памяти, чем простая переменная.
- ③ Иногда производительность кода будет ниже. Из-за особенностей подхода часть вещей может быть реализована сложнее, чем могла бы быть. Поэтому бывает такое, что ООП-программа работает медленнее, чем процедурная (хотя с современными мощностями процессоров это мало кого волнует).

5. Które składowe klasy są najważniejsze i dlaczego?

<https://docplayer.pl/17561576-Obiektowy-php-czym-jest-obiekt-definicja-klasy-skladowe-klasy-pola-i-metody.html>

6. Który mechanizm paradygmatu programowania obiektowego jest najważniejszy i dlaczego?

<https://mansfeld.pl/programowanie/zalety-i-wady-oop/>

7. Jaka jest istotna różnica pomiędzy programowaniem obiektowym a programowaniem prototypowym?

https://pl.qaz.wiki/wiki/Prototype-based_programming

https://pl.qaz.wiki/wiki/Class-based_programming

8. Co jeszcze oprócz funkcji i metod możemy określić jako abstrakcję przy tworzeniu oprogramowania z wykorzystaniem programowania zorientowanego obiektowo?

http://zasoby.open.agh.edu.pl/~09sdczerner/strona/page/Klasy_abstrakcyjne_i_interfejsy.html

<https://bartłomiejchmielewski.pl/klasa-abstrakcyjna-java/>

<https://nullpointerexception.pl/pytania-rekrutacyjne-czym-rozni-sie-klasa-abstrakcyjna-od-interfejsu/>

9. Omów mechanizm polimorfizmu w zakresie możliwości wykorzystania szablonów.

<https://www.modestprogrammer.pl/co-to-jest-polimorfizm-w-programowaniu-obiektowym>

<https://javastart.pl/baza-wiedzy/programowanie-obiektowe/polimorfizm>

<https://strefainzyniera.pl/artykul/933/polimorfizm-dziediczenie-i-hermetyzacja-w-java>

10. Wyjaśnij na czym polega polimorfizm w zakresie możliwości wykorzystania mechanizmu przeciążania operatorów.

<https://www.modestprogrammer.pl/co-to-jest-polimorfizm-w-programowaniu-obiektowym>

https://www.plukasiewicz.net/CSharp_dla_początkujacych/Polimorfizm

<http://www.csharpowezmagania.pl/polimorfizm-w-jezyku-c/>

11. Jaka jest znacząca różnica pomiędzy metaprogramowaniem a tworzeniem kodu źródłowego z wykorzystaniem podejścia generycznego?

<https://www.samouczekprogramisty.pl/typy-generyczne-w-jezyku-java/>

<https://ru.nipponkaigi.net/wiki/Metaprogramming>

12. Omów struktury kontrolne przy pomocy których możliwe jest zbudowanie dowolnego programu. Podaj praktyczne przykłady każdego wskazanego typu struktury.

<https://phpkurs.pl/struktury/>

https://pl.wikipedia.org/wiki/Programowanie_structuralne

13. Co w praktyce oznacza, że kod źródłowy programu spełnia warunek deterministyczności?

<http://www.ebitech.pl/pl/blog/programowanie-c-cpp.html>

<https://isystems.pl/blog/artykul.html?id=346>

Deterministyczny — dla takich samych argumentów wejściowych zawsze daje ten sam wynik.

Ważne jest również, aby raz powstały kod łatwo było analizować i konserwować oraz modyfikować i rozwijać w przyszłości. Warto też zadbać na etapie projektowania i tworzenia kodu źródłowego o to, aby program zachowywał się zawsze w sposób deterministyczny i zgodny z założonym, aby działania tego nie pozostawać przypadkowi oraz żeby już na etapie projektu eliminować źródła potencjalnych błędów czy nieoczekiwanych zachowań w przyszłości. Dodatkowo, podczas realizacji rozległych, komercyjnych projektów, w które zaangażowane są duże zespoły specjalistów, dobrze jest już na samym początku szczegółowo określić standardy tworzenia kodu, aby w rezultacie otrzymać spójne oprogramowanie o wysokiej jakości i niezawodności.

Najpierw wyjaśnijmy, co to znaczy że funkcja jest lub nie jest deterministyczna. Funkcja jest deterministyczna wtedy, kiedy dla takich samych parametrów zwróci zawsze ten sam wynik. To oznacza, że taka funkcja musi działać zawsze w ten sam sposób, a na wynik nie powinny wpływać żadne czynniki zewnętrzne tj. funkcja nie powinna korzystać z żadnych zmiennych pakietowych ani innych źródeł zewnętrznych. Taka funkcja nie może też zmieniać żadnych danych w bazie (w tabelach ani pakietach). W niektórych sytuacjach wymagane jest by funkcja była deterministyczna. Przykładowo jeśli zechcemy użyć własnej funkcji w indeksie funkcyjnym, to funkcja ta musi być deterministyczna. Nie tylko spełniać warunek jako taki, ale też musi to być jasno określone w treści funkcji.

- 14. Omów typy i przypadki zastosowania znanych Ci modyfikatorów parametrów metod języków obiektowych takich jak C++, Java i C#.**

<https://www.samouczekprogramisty.pl/modyfikatory-dostepu-w-jazyku-java/>

http://imeia.elektr.polsl.pl/files/materials/si/Serwisy%20internetowe%20%20-%20PHP_3.pdf

- 15. W jaki sposób możemy zdefiniować metodę synchroniczną w kodzie źródłowym i na czym ona polega?**

<http://szymonwieloch.com/pl/2018/02/09/synchroniczne-vs-asynchroniczne-operacje-wejcia-wyjcia/>
Operacje synchroniczne

Operacje synchroniczne są bardzo proste do zrozumienia. Są one stosowane przez 95% wszystkich programistów nowej daty, co tłumaczy powolność działania wielu współczesnych aplikacji. Operacje te polegają na tym, że kiedy wołamy funkcje czytające z pliku lub gniazda, to wątek jest zatrzymywany do momentu dotarcia tych danych. Ponieważ najlepiej znam język C/C++, więc postanowiłem przykłady umieszczać właśnie w tym języku.

Upraszczając sprawę, można powiedzieć, że operacje synchroniczne (zwane także blokującymi), to takie operacje, które zatrzymują wątek do momentu pobrania lub zapisania danych do wejścia lub wyjścia. W podanym wcześniej przykładzie aplikacji moich kolegów wołali oni funkcję blokującą, pobierającą z gniazda dane. Wywołanie funkcji kończyło się dopiero wtedy, kiedy dane dotarły, a operacje na gniazdach są -podobnie do operacji dyskowych – stosunkowo wolne. Aby jednocześnie obsługiwać 3000 połączeń, potrzebne było 3000 wątków. Narzut związany z wątkami doprowadził ostatecznie do ogromnego spowolnienia aplikacji.

16. Podaj praktyczny przykład obsługi wyjątku kontrolowanego i wyjątku niekontrolowanego. Kiedy możemy stosować wyjątki kontrolowane?

https://achilles.tu.kielce.pl/portal/Members/596d88a5187b42e2bad090bb66227a25/pub/java/new/instrukcja_laboratoryjna_8.html

<https://www.samouczekprogramisty.pl/wyjatki-w-jazyku-java/>

2. Obsługa wyjątków

Wyjątki są **wyrzucane i propagowane** tak długo jak długo pozostają nie przechwycone w kodzie programu. Wyjątki nie przechwycone propagują się aż do metody `main(...)` i jeśli także tam nie są przechwycone powodują przerwanie wykonania programu.



W celu przechwycenia i obsłużenia wyjątków w języku Java korzysta się z bloku **try-catch-finally**.

Kod którego wyjątki chcemy przechwytywać umieszczamy w klauzuli `try {...}`. Wewnątrz klauzuli `catch {...}` umieszczamy **kod obsługi błędu** - kod ten będzie wykonany tylko i wyłącznie wówczas gdy kod programu wyrzuci wyjątek *typu* zgodnego z typem określonym w deklaracji klauzuli `catch`. Kod umieszczony wewnątrz klauzuli `finally {...}` będzie wykonany **zawsze**, niezależnie od tego czy kod programu wyrzucił wyjątek czy nie.

Przykład 1. Przykład definicji bloku try-catch-finally

```
try {  
    //kod programu generujacy wyjątek  
} catch (Exception e) {  
    //obsługa wyjątków klasy Exception  
} catch (Throwable t) {  
    //obsługa wyjątków klasy Throwable  
} finally {  
    //kod zawsze wykonywany  
}
```

Gdy w wyniku wykonania instrukcji w bloku `try` powstanie wyjątek typu `Exception` lub `Throwable` to sterowanie zostanie przekazane do kodu umieszczonego w w/w klauzulach `catch`.



Klauzula `finally` jest opcjonalna, za to klauzul `catch` może być dowolnie wiele.

Przez pojęcie wyjątków w językach obiektowych rozumiemy mechanizm kontroli przepływu służący do obsługi zdarzeń wyjątkowych (w szczególności błędów). Wyjątek w języku Java to obiekt, który opisuje pewną sytuację błędą lub nieprawidłową – wyjątkową. Jest to obiekt odpowiedniego typu, tj. obiekt klasy `Throwable` lub jej dowolnej podklasy (np. `Exception`, `Error` itd.).

Ważną częścią każdej aplikacji jest sygnalizowanie oraz obsługa pojawiających się błędów. Wystąpienie błędu w aplikacji zaimplementowanej w języku Java sygnalizujemy poprzez rzucenie (wyrzucenie) wyjątku. Obsługa błędów to tzw. łapanie (przechwytywanie) wyjątków.

Wyjątki dzielą się na dwa rodzaje:

- kontrolowane - to takie które musimy deklarować i obsługiwać,
- niekontrolowane - to takie które możemy obsługiwać, ale nie musimy.

Wyjątki niekontrolowane to instancje klas `Error` i `RuntimeException` oraz ich dowolnych podklas. Wyjątki kontrolowane to instancje klas `Throwable` i `Exception` oraz ich podklas, z wyłączeniem podklas klas `Error` i `RuntimeException`. Wyjątki mogą być rzucane (zgłoszane) przez Wirtualną Maszynę Javy (JVM) oraz przez programistę. Najpopularniejszym wyjątkiem rzucanym przez JVM jest `NullPointerException`. Wyjątek ten rzucany jest przy próbie odwołania się do obiektu (np. próba uruchomienia metody) z użyciem referencji, która ma wartość `null`. `NullPointerException` jest podklassą klasy `RuntimeException` tak więc jest wyjątkiem niekontrolowanym – nie musimy go w żaden sposób deklarować czy obsługiwać.

17. W jaki sposób możemy sterować procesem serializacji obiektów w tworzonym oprogramowaniu?

<https://docs.microsoft.com/pl-pl/dotnet/csharp/programming-guide/concepts/serialization/>

<https://www.samouczekprogramisty.pl/serializacja-w-jezyku-java/>

http://www.javaexpress.pl/article/show/J2ME_Serializacja_obiektow_cz_I?lang=pl

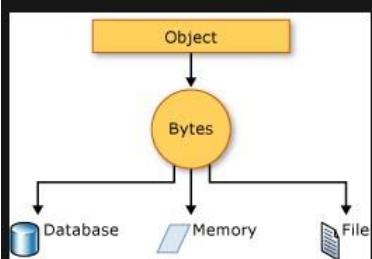
Serializacja (C#)

02.01.2020 • Czas czytania: 4 min •  

Serializacja jest procesem konwertowania obiektu do strumienia bajtów w celu przechowywania obiektu lub przesyłania go do pamięci, bazy danych lub pliku. Jego głównym celem jest zapisanie stanu obiektu, aby można było go odtworzyć w razie potrzeby. Proces odwrotny nazywa się deserializacją.

Jak działa Serializacja

Na tej ilustracji przedstawiono ogólny proces serializacji:



Obiekt jest serializowany do strumienia, który przenosi dane. Strumień może również zawierać informacje o typie obiektu, takie jak jego wersja, kultura i nazwa zestawu. Z tego strumienia obiekt może być przechowywany w bazie danych, pliku lub pamięci.

Używa do serializacji

Serializacja umożliwia deweloperowi zapisanie stanu obiektu i ponowne utworzenie go w razie potrzeby, zapewniając przechowywanie obiektów oraz wymianę danych. Dzięki serializacji programista może wykonywać takie działania, jak:

- Wysyłanie obiektu do aplikacji zdalnej przy użyciu usługi sieci Web
- Przekazywanie obiektu z jednej domeny do innej
- Przekazywanie obiektu przez zaporę jako ciąg JSON lub XML
- Utrzymywanie zabezpieczeń lub informacji specyficznych dla użytkownika w aplikacjach

18. Omów uwarunkowania dostępu do składowych prywatnych obiektu przy zastosowaniu mechanizmu odzwierciedlen (refleksji).

https://pl.wikipedia.org/wiki/Mechanizm_refleksji

http://www.programowanieobiektywne.pl/java_objekty_refleksyjne.php

<https://edu.pjwstk.edu.pl/wyklady/mpr/scb/W3/W3.htm>

Co to jest obiekt niezmienny?

Obiekt niezmienny (immutable) to taki, który po utworzeniu i inicjalizacji pozostaje niezmienny i co ważne, nie ma możliwości jego zmiany (oczywiście w konwencjonalny sposób). Czyli taki obiekt nie udostępnia metod, które pozwalają na zmianę jego stanu. A jego wszystkie pola są prywatne.

Niezmienna ta jest trochę pozorna, ponieważ, przy użyciu refleksji można zmieniać wartości pól w każdej klasie. Wprawdzie wymaga to odrobinę wysiłku i znajomości api refleksji, ale nie jest to *rocket science*. Jednak to wcale nie oznacza, że z tego powodu powinniśmy zrezygnować z niezmiennych obiektów.

Mechanizm refleksji – pojęcie z dziedziny informatyki oznaczające proces, dzięki któremu program komputerowy może być modyfikowany w trakcie działania w sposób zależny od własnego kodu oraz od zachowania w trakcie wykonania. Paradymat programowania ściśle związany z mechanizmem refleksji to *programowanie refleksyjne*.

Refleksja pozwala w łatwy sposób zarządzać kodem tak, jakby był danymi. Używa się jej najczęściej do zmieniania standardowego zachowania już zdefiniowanych metod lub funkcji, a także do tworzenia własnych konstrukcji semantycznych modyfikujących język. Z drugiej strony kod wykorzystujący **refleksję** jest mniej czytelny i nie pozwala na sprawdzenie poprawności składniowej i semantycznej w trakcie kompilacji (niewygodne śledzenie błędów).

Mechanizm ten jest częściej spotykany w językach wysokiego poziomu, zwykle opartych na maszynie wirtualnej.

Poniższy przykład w języku Java wykorzystuje pakiet `java.lang.reflect`.

```
// bez refleksji
Foo foo = new Foo();
foo.hello();

// z refleksją
Class cl = Class.forName("Foo");
Method method = cl.getMethod("hello");
method.invoke(cl.newInstance());
```

Oba fragmenty tworzą instancję klasy `Foo`, następnie wywołują metodę `hello()` tej klasy. Różnica polega na tym, że w pierwszym fragmencie nazwa klasy i metody są częścią kodu źródłowego, podczas gdy w drugim fragmencie możliwe jest przeniesienie ich do zmiennych, których wartość jest ustalana w czasie wykonania kodu.

Mechanizm refleksji pozwala także na zdobywanie informacji o klasach w trakcie wykonania programu. W poniższym przykładzie klasa `Main` sprawdza jaki jest typ zwracany przez metody klasy `Bar`.

```
public class Bar {
    public String fun(Integer i) {
        return "0" + i + ", zglos sie!";
    }
}

import static java.lang.System.out;
import java.lang.reflect.*;

public class Main {
    public static void main(String[] args) throws Exception {
        String className = "Bar";
        Class c = Class.forName(className);
        Method[] methodArr = c.getDeclaredMethods();
        for (Method m : methodArr) {
            out.print("Klasa " + className + " ma metode '" + m.getName() + "' ");
            out.println(" ktora zwraca wartosc typu " + m.getReturnType());
        }
    }
}
```

19. Odwołując się do znanego Ci języka programowania obiektowego wskaz problemy związane z wykorzystywaniem kolekcji przechowujących obiekty.

<https://www.javappa.com/kurs-java/kolekcje-wprowadzenie>

<https://blog.it-leaders.pl/rozmowa-kwalifikacyjna-javy-zaden-problem-cz-ii-kolekcje/>

Porównać dynamiczne arrays vs collections

20. W jaki sposób w kodzie źródłowym należy odzwierciedlić brak modyfikatora metod, który jest widoczny w diagramie klas UML

<https://javastart.pl/baza-wiedzy/java-podstawy-języka/modyfikatory-dostępu>

<https://www.p-programowanie.pl/uml/diagramy-klas-uml>

Składniki klasy mogą posiadać różne **modyfikatory dostępu**:

- `+` to składnik publiczny (public)
- `#` to składnik chroniony (protected)
- `-` to składnik prywatny (private)
- `~` to składnik dostępny w obrębie projektu (package)
- metody abstrakcyjne w klasie abstrakcyjnej są pochylone lub podkreślone.

domyślny (package private)

W przypadku gdy nie podamy żadnego modyfikatora użyty zostanie domyślnie package private. Oznaczony element widoczny będzie tylko w ramach tego samego pakietu. Modyfikator domyślny jest raczej rzadko wykorzystywany, a warto po niego sięgać jako domyślnego specyfikatora dla klas. Dzięki temu możemy z pakietów tworzyć swego rodzaju moduły.

INTERNET RZECZY, SIECI KOMPUTEROWE

1. W jaki sposób zbudować prostą sieć do małego biura.

<https://itbiznes.pl/poradnik/siec-w-malej-firmie/>

2. W jakich sytuacjach wskazane byłoby zastosowanie medium: kable miedziane/światłowód/transmisja bezprzewodowa.

https://pl.wikipedia.org/wiki/Medium_transmisyjne

3. Jakie problemy pojawiają się z adresacją IPv4.

<https://www.komputerswiat.pl/aktualnosci/wydarzenia/koncza-sie-adresy-ipv4-internet-jedzie-na-rezerwie/mprw1zj>

4. Co jest potrzebne do wdrożenia IPv6 w sieci LAN.

<https://www.komputerswiat.pl/aktualnosci/wydarzenia/koncza-sie-adresy-ipv4-internet-jedzie-na-rezerwie/mprw1zj>

<https://www.ibm.com/docs/pl/i/7.1?topic=6-comparison-ipv4-ipv6>

5. Jakie problemy występują w użytkowaniu i zarządzaniu sieciami Wi-Fi.

<https://www.hotgear.pl/jak-to-zrobic/problemy-z-siecia-wi-fi-oto-te-najczestsze-i-proste-sposoby-jak-sobie-z-nimi-poradzic/>

6. Jak zabezpieczyć urządzenia sieciowe przed zagrożeniami.

Jak zabezpieczyć router?

Wystarczy wykonać kilka czynności, żeby nasza sieć była bezpieczniejsza. Żadne zabezpieczenie routera nie wyeliminuje całkowicie ryzyka włamania, ale zdecydowanie je zmniejszy. Oto podstawowe kroki, które warto podjąć:

- Ustawienie trudnego hasła – to absolutna podstawa nie tylko w przypadku routera. Zawsze powinniśmy stosować skomplikowane hasła dostępowe. Warto przestrzegać zasad, żeby miało ono co najmniej 8 znaków, składało się z małych i wielkich liter, cyfr i znaków specjalnych. Często na sprzętach, które otrzymujemy od dostawców internetu, jest już założona sieć Wi-Fi, a hasło znajduje się na naklejce z tyłu urządzenia. Warto je zmienić na takie, które będziemy w stanie zapamiętać, ale nie będzie ono oczywiste. Niestety, ale w dalszym ciągu wśród najpopularniejszych haseł krążą takie jak 123456, password czy qwerty. Trudno je traktować jako dobre zabezpieczenie sieci Wi-Fi.
- Zmiana nazwy sieci – najczęściej producenci korzystają ze standardowych ciągów znaków w przypadku nazw Wi-Fi. Przy zakładaniu takiej sieci, również otrzymamy standardową propozycję. Lepiej ją zmienić na własną, żeby nie ułatwiać pracy przestępcy, a jeszcze lepszym rozwiązaniem jest ukrycie rozgłaszenia SSID.
- Stworzenie sieci dla gości – kolejny sposobem na to, jak zabezpieczyć Wi-Fi, może być założenie sieci dla gości. Wówczas nie będą oni mieć dostępu do naszego Wi-Fi, w tym np. do urządzeń sieciowych, a będą mogli korzystać z internet w takim zakresie, w jakim im to umożliwimy. To bardzo dobre rozwiązanie, dzięki któremu osoby z zewnątrz nie będą znać naszego hasła, a jeśli ukryjemy rozgłaszanie sieci, to również nie sprawdzą, jak się ona nazywa.
- Zmiana hasła dostępu do routera – żeby skonfigurować urządzenie, musimy zalogować się do panelu administracyjnego routera. Wykorzystać należy do tego dane, które znajdziemy w instrukcji do sprzętu. Często nazwą użytkownika będzie admin, a hasłem np. 12345. Koniecznie należy zmienić te dane, żeby utrudnić cyberprzestępcom włamanie się do naszej sieci. Nie zawsze da się zmodyfikować login, ale jeśli jest taka możliwość, to warto z niej skorzystać.
- Ustawienie protokołu szyfrowania – wysoką ochronę zapewniają standardy WPA2 oraz WPA3, natomiast WPA, WEP czy TKIP mają jednak wiele luk i nie są zalecane. Szyfrowanie ma bardzo duże znaczenie, gdyż utrudnia hakerom przejęcie danych.

- Wyłączenie funkcji WPS – jest to łatwa metoda na podłączenie danego urządzenia do sieci bezprzewodowej, poprzez wpisanie prostego PIN-u albo naciśnięcie odpowiedniego przycisku. Jeśli zależy nam na bezpieczeństwie sieci, to lepiej dezaktywujmy taką możliwość, tym bardziej, jeśli z niej nie korzystamy. Jeśli taka opcja będzie dostępna, to ułatwimy cyberprzestępcom włamanie się do naszej sieci.
- Aktualizowanie oprogramowania – jest to niezbędne z dwóch względów. Po pierwsze, dzięki temu uzyskamy dostęp do najnowszej wersji oprogramowania, co z reguły zwiększa możliwości sprzętu, a po drugie, aktualizacje są wydawane po to, aby wyeliminować luki bezpieczeństwa w urządzeniu.
- Korzystanie z oprogramowania antywirusowego – solidny program, który działa w tle i eliminuje zagrożenia na naszym komputerze to podstawa. Jeśli na dysku naszego urządzenia są wirusy, to efektem ich działania może być kradzież danych, w tym tych do logowania.
- Filtrowanie adresów MAC – każda karta sieciowa ma swój adres MAC. W panelu administracyjnym routera możemy dodać, jakie urządzenia mogą korzystać z naszej sieci. To dobry sposób na to, jak zabezpieczyć sieć, ale nie daje on 100 proc. gwarancji, że z naszego Wi-Fi nie skorzysta ktoś inny. Obecnie karty sieciowe dają możliwość zmiany numeru MAC.

7. Jak wykorzystać sieci VLAN w sieciach korporacyjnych.

<https://eia.pg.edu.pl/documents/1113028/0/W12%20-%20VLAN%2C%20VPN.pdf>

8. W jaki sposób zapewnić wysoką niezawodność sieci LAN.

<https://docplayer.pl/2320042-Niezawodnosc-sieci-lan.html>

9. Jak ograniczyć ataki na sieci LAN.

1.1. Bezpieczeństwo sieci lokalnej

Na styku sieci lokalnej z Internetem instaluje się specjalizowane urządzenia do ochrony sieci. Może okazać się, że sam firewall nie wystarcza. Skala i różnorodność zagrożeń wymaga od administratorów dużej wiedzy i stosowania rozbudowanych mechanizmów bezpieczeństwa. Takim rozwiązaniem są urządzenia UTM (ang. Unified Threat Management) czyli kompleksowy, a przede wszystkim zintegrowany sposób zarządzania siecią i jej bezpieczeństwem. Sprzętowe rozwiązania UTM integrują w jednej obudowie wszystkie elementy niezbędne do kompletnego zabezpieczenia sieci lokalnej, np.:

- firewall - administrator ma możliwość zdefiniowania wielu różnych zestawów reguł określających jaki ruch powinien być przez firewall przepuszczany a jaki blokowany. Pozwala także na ustalenie np. innych zasad filtrowania ruchu w godzinach pracy, innych w godzinach popołudniowych, a jeszcze innych w dni wolne od pracy.
- IPS – (ang. Intrusion Prevention System) system Intrusion Prevention wykorzystuje technologię wykrywania i blokowania ataków ASQ (Active Security Qualification). Analizie w poszukiwaniu zagrożeń i ataków poddawany jest cały ruch sieciowy od trzeciej (warstwa sieciowa) do siódmej (warstwa aplikacji) warstwy modelu OSI. Kontroli poddawane są nie tylko poszczególne pakiety ale także połączenia i sesje.
- serwer VPN (ang. Virtual Private Networks) - pozwala na tworzenie bezpiecznych połączeń, tzw. kanałów VPN. Kanały VPN mogą być tworzone pomiędzy użytkownikami pracującymi w terenie (tzw. zdalnymi użytkownikami) a siedzibą firmy lub pomiędzy centralą a oddziałami firmy.
- ochrona antywirusowa – na obecność wirusów sprawdzana jest poczta przychodząca, wychodząca, odwiedzane strony, pobierane pliki,
- ochrona przed spamem
- filtrowanie URL - filtr URL może być ustawiany dla wszystkich lub wybranych grup użytkowników definiowanych przez administratora. Dodatkowo określone filtry mogą działać tylko w wyznaczonych godzinach, dzięki czemu użytkownicy mogą np. w godzinach popołudniowych mieć zapewniony szerszy dostęp do Internetu niż w czasie godzin pracy.

10. Kiedy routing statyczny ma przewagę nad routingiem dynamicznym.

<https://pl.fondoperlaterra.org/comdifference-between-static-and-dynamic-routing-14>

11. Jakie narzędzia stosuje się do rozwiązywania problemów w sieci.

<https://www.manageengine.com/pl/network-monitoring/network-management.html>

Kali linux

12. Jakie znaczenie ma zgodność rozwiązania ze standardem.

13. W jaki sposób dobrą technologię dostępu do Internetu.

W Polsce mamy wiele możliwości podłączenia się z Internetem.

POŁĄCZENIE MODEMOWE (Dial-up) - powoli odchodzący do lamusa, ale wciąż stosowany sposób na dostęp do Internetu. Jest to łącze analogowe o teoretycznej przepustowości 56 kbit/s. Ze względu na niską prędkość oraz konfliktowy sposób połączenia (połączenie jest realizowane na tej samej linii co połączenie telefoniczne, i najczęściej jest ono zrywane przy próbie korzystania z telefonu) jest ono wysoce niewystarczające w dzisiejszych czasach, i nie pozwala na komfortowe korzystanie z wielu usług dostępnych w sieci. Połączenie modemowe miało zasadnicze znaczenie w początkach Internetu w naszym kraju, tj. w początkach lat 90 - tych, kiedy to nie było tak wielu technicznych możliwości, dla innych metod dostępowych. Krokiem naprzód, było wprowadzenie cyfrowej łączności ISDN, gdzie został wyeliminowany problem jednociesnego połączenia z Internetem i korzystania z telefonu.

ŁĄCZE SZEROKOPASMOWE - zapewnia stały dostęp do Internetu z prędkością od 128 kb do 2048 kbit/s, są to łącza ADSL (Neostrada) i DSL. W zasadzie monopolistą na rynku w Polsce jest Telekomunikacja Polska SA. Dla połączeń ADSL charakterystyczny jest sposób łączenia, Mamy do dyspozycji modem podłączany do łącza USB w komputerze, a połączenia dokonujemy z pomocą aplikacji dostarczonej przez TPSA. Cechą charakterystyczną dla DSL, jest posiadanie stałego, zewnętrznego adresu IP. Oprócz powyższych, istnieją łącza o znacznie szybszym transferze, około 100 Mbit/s, lecz ze względu na wysoki koszt mają zastosowanie jedynie w dużych firmach i przedsiębiorstwach, oraz instytucjach bezpośrednio związanymi z siecią - np. portale internetowe.

LOKALNA SIEĆ - bardzo często wykorzystywana metoda w miastach i większych skupiskach mieszkaniowych. Jest to zorganizowane zwykle tak, że serwer sieci, jest na stałe podpięty do Internetu, zwykle za pomocą DSL lub podobnego, pełniąc rolę bramy internetowej. Łącze to jest udostępniane wszystkim użytkownikom sieci, za pomocą infrastruktury sieci (swiche, kable lub bezprzewodowo). Dodatkowym atutem takiego rozwiązania jest możliwość wymiany plików w obrębie sieci, bezpieczeństwo oraz możliwość łatwej rozbudowy sieci. Dość znaczącym czynnikiem jest tutaj możliwość łączności za pomocą fal radiowych. Coraz popularniejsze stają się urządzenia do bezprzewodowej łączności, (pracujące w zakresie 2,4GHz i 5,4GHz) co daje możliwości dotarcia do użytkowników nawet bardzo odległych.

ŁĄCZA ŚWIATŁOWODOWE - sposób dostępu, w którym użyto światłowodów, zapewniających przesył danych oszałamiającą prędkością, około 7 Tb/s. Sieci zbudowane na światłowodach, noszą nazwę FDDI (Fiber Distributed Data Interface) i dają możliwość stosowania wielu protokołów jednocześnie, co przekłada się w wysokowydajnym transferze.

14. Jak zapewnić właściwy przesył danych głosowych w sieci.

Jakość głosu

Realizowana z zastosowaniem protokołu IP usługa VoIP umożliwia przesyłanie głosu w postaci pakietów w sieci z komutacją pakietów. W porównaniu z dotychczasową techniką przenoszenia sygnału, postać cyfrowa sygnału zawarta w pakietach pozwala na nieporównywalnie łatwiejsze sposoby rejestracji i przetwarzania, skuteczniejszą kompresję oraz łatwiejsze transportowanie. Szczególnie ważną właściwością cyfrowej reprezentacji sygnału jest możliwość regeneracji treści użytkowej sygnału mowy bez wzmacniania szumów kanału z jego otoczenia, wraz z możliwością kompensacji echa. Cały etap przetwarzania analogowego sygnału akustycznego składa się z cyfryzacji, kompresji i formowania do postaci pakietów IP, które następnie są transmitowane przez sieci IP.

Usługa przesyłania głosu w postaci pakietów w sieci IP wymaga zmiany wcześniej przyjętych założeń dotyczących protokołu IP. Został on bowiem zaprojektowany do przenoszenia danych, zapewnia więc tylko usługę *best effort* (o najwyższej możliwej jakości), nie gwarantując poprawności realizacji usług w czasie rzeczywistym. W przypadku transmisji danych brak mechanizmów zapewnienia jakości transmisji w globalnej sieci IP ma stosunkowo niewielkie znaczenie. Istnieje możliwość uszeregowania pakietów danych odebranych w niewłaściwej kolejności, a uszkodzone lub utracone fragmenty informacji mogą zostać powtórnie przesłane. Użytkownik nawet może nie zauważyc ewentualnego opóźnienia przychodzących pakietów.

Zupełnie inaczej wygląda sytuacja w przypadku transmisji głosu, dla której jakość usługi ma zasadnicze znaczenie. Nie ma tu możliwości, aby uszkodzone lub utracone pakiety głosowe były retransmitowane w akceptowalnym czasie. Aby usługa VoIP była przyjęta przez użytkowników, występujące opóźnienia pakietów nie mogą być zbyt duże. Nadmierne stosowanie mechanizmu buforowania pakietów w punkcie docelowym, w celu przywrócenia właściwej ich kolejności, będzie powodowało, że pakiety głosowe staną się bezużyteczne.

Dla skutecznego wdrożenia usługi VoIP w istniejących sieciach danych najistotniejsze jest zapewnienie właściwej jakości głosu. Znane są różne mechanizmy, które umożliwiają uzyskanie poziomu niezawodności i jakości mowy porównywalnej z wynikami otrzymywanyimi w sieci telekomunikacyjnej użytku publicznego PSTN/ISDN z komutacją łączny.

15. W jaki sposób ograniczyć dostęp do wybranych hostów lub usług.

Blokowanie użytkowników po adresie IP

W sieci Internet robi się od użytkowników, którzy obierają sobie za cel przeszkadzanie w prowadzeniu witryn internetowych. Potocznie nazywani są oni "trollami". Pod pojęciem "trolla" ukrywa się definicja użytkownika, który w zamierzony sposób wpływa na innych użytkowników w celu ich ośmieszenia lub obrażania – czego następstwem jest najczęściej wywołanie kłów.

Moderatorzy oraz autorzy stron WWW stosują najczęściej proste narzędzia do blokowania takich użytkowników (np. blokada adresu e-mail). Takie zabezpieczenie jest skuteczne na krótką metę. Najbardziej efektywnym sposobem zablokowania niechcianych użytkowników jest skorzystanie z pliku .htaccess, który pozwoli na zablokowanie problematycznego użytkownika "u źródła", czyli po adresie IP lub host.

16. Czym różni się Arduino od Raspberry Pi.

Już na wstępie warto podkreślić, że tak naprawdę ciężko wykonać bezpośrednie porównanie, bo Raspberry Pi oraz Arduino to zupełnie różne platformy. Główną i najważniejszą różnicę między nimi stanowi to, że:

- Raspberry Pi jest w pełni funkcjonalnym komputerem z systemem operacyjnym,
- Arduino jest zestawem uruchomieniowym z prostym mikrokontrolerem.

Obie platformy zaprojektowano jako urządzenia do nauki, dla osób, które zaczynają przygodę z elektroniką i programowaniem. Później okazało się, że sprawdzają się w tym doskonale, a ich zastosowania wykraczają znacznie poza tematy edukacyjne.

17. Jakie języki programowania używane są w technologiach IoT.

Szacuje się, że obecnie na całym świecie projektowaniem sieci IoT para co najmniej 6 mln osób. W opracowanym niedawno raporcie Eclipse Foundation's 2018 IoT Developer Survey można przeczytać, że w zeszłym roku największym wzięciem wśród deweloperów sieci IoT cieszył się język programowania Java. Kolejne miejsca zajęły następujące języki: C, JavaScript, Python, C++, PHP, C, Assembler, LUA, Go, Swift, Ruby i Rust.

Pierwsze cztery miejsca na tej liście okupują od dwóch te same języki i nawet ich kolejność nie uległa zmianie (Java, C, JavaScript i Python wygrały również tę rywalizację w 2017 roku). Tegoroczny ranking różnił się jednak od poprzedniego tym, że języki programowania podzielono na trzy grupy.

18. Jakie komponenty będą potrzebne do zbudowania systemu inteligentnego domu.

<https://www.fhome.pl/pl/poradniki/co-powinien-miec-kazdy-inteligentny-dom.html>

19. W jakim celu stosuje się przetwarzanie w Chmurze.

Przetwarzanie w chmurze (ang. cloud computing) to rodzaj obliczeń w oparciu o Internet, który zapewnia udostępniane zasobów obliczeniowych i przetwarzanie danych na żądanie. Jest to model umożliwiający dostęp na żądanie do wspólnej puli konfigurowalnych zasobów obliczeniowych (na przykład sieci komputerowych, serwerów, pamięci masowych, aplikacji i usług), które mogą być szybko dostarczone lub zwalniane przy minimalnym nakładzie na zarządzanie nimi. Przetwarzanie w chmurze i rozwiązania pamięci masowej zapewniają użytkownikom i przedsiębiorstwom, przechowywanie i przetwarzanie ich danych w centrach danych stron trzecich, które mogą być zlokalizowane daleko od użytkownika, w dowolnym miejscu na świecie. Przetwarzanie w chmurze polega m.in. na współdzieleniu zasobów i optymalizację wynikającą z efektu skali.

Zwolennicy twierdzą, że przetwarzanie w chmurze pozwala organizacjom uniknąć wysokich kosztów przygotowania infrastruktury (na przykład zakupu serwerów). Umożliwia również organizacjom skupienie się na swojej podstawowej działalności, zamiast tracić czas i pieniądze na infrastrukturę komputerową. Zwolennicy twierdzą również, że przetwarzanie w chmurze pozwala przedsiębiorstwom uzyskać dostęp do niezbędnych aplikacji szybciej, z poprawą zarządzania nimi i mniejszymi nakładami na ich utrzymanie oraz umożliwia zespołom IT na szybsze dostosowanie zasobów do spełnienia zmiennych i nieprzewidywalnych oczekiwani biznesu.

20. Jakie rozwiązania stosowane są do wymiany informacji w systemach IoT.

Sieci komórkowe

Jednym z najbardziej tradycyjnych i najstarszych sposobów zestawiania bezprzewodowego połączenia M2M jest wykorzystanie sieci komórkowych. Do zalet tego rozwiązania zaliczyć można niemal globalne pokrycie zapewniane przez istniejącą już infrastrukturę stacji bazowych, wysoką prędkość transmisji, jak również ciągły rozwój technologii (zbliżająca się era 5G). Największą wadę stanowi wysokie zużycie energii, przez co opcja ta ograniczona jest praktycznie jedynie do urządzeń zasilanych z sieci elektrycznej lub łatwo i często ładowanych.

Sieci krótkiego zasięgu

Znaczna większość różnego typu systemów komunikacji dla IoT zapewnia łączność na znacznie krótszym dystansie, rzędu dziesiątek lub co najwyżej setek metrów. W tekście omówione zostaną następującego rozwiązania: Bluetooth, Wi-Fi, ZigBee, Z-Wave, NFC, RFID, 6LoWPAN, Thread oraz WirelessHART.

Bluetooth

Technologia Bluetooth jest jedną z najważniejszych w świecie IoT. W związku z jej popularnością w branży urządzeń mobilnych (prawdopodobnie każdy produkowany obecnie smartfon zdolny jest do obsługi tego standardu) umożliwia bardzo łatwe zestawienie sieci urządzeń osobistych o zasięgu do

kilkudziesięciu metrów. Z tego powodu świetnie nadaje się jako podstawowy rodzaj komunikacji dla wszelkiego typu urządzeń noszonych.

Wi-Fi

Niewątpliwą zaletą standardu Wi-Fi jest wysoka prędkość transmisji oraz duża dostępność infrastruktury. Najpopularniejsza obecnie odmiana tego standardu, czyli 802.11n, zapewnia łączność z maksymalną prędkością 600 Mbit/s na odległościach dochodzących do 50 m. Technologia Wi-Fi w świecie komputerów oraz urządzeń mobilnych jest podstawowym sposobem uzyskiwania bezprzewodowego dostępu do Internetu, przez co istniejącą infrastrukturę sieciową znaleźć można w większości nowoczesnych budynków, zarówno w prywatnych domach, jak i miejscach użyteczności publicznej. Wi-Fi wykorzystuje nielicencjonowane pasma 2,4 oraz 5 GHz.

RFID oraz NFC

Technologia RFID stosowana jest przede wszystkim do śledzenia obiektów, co przydaje się w logistyce, handlu (np. metki na produktach sklepowych), ale też w naukach przyrodniczych (monitorowanie dziko żyjących zwierząt), opiece zdrowotnej czy przemyśle.

Standard NFC (Near Field Communication) również oparty jest na technologii RFID. Główna różnica polega na tym, że każdy układ czytnika NFC może pracować również jako tag, czyli emulować działanie karty NFC. Dwa tego typu układy mogą w ten sposób wymieniać informacje pomiędzy sobą. Standard NFC wykorzystuje tę samą częstotliwość co HF RFID (13,56 MHz), dlatego umożliwia komunikację jedynie na niewielkim dystansie. Krótki zasięg, rzędu centymetrów, jest jednocześnie atutem tego rozwiązania, ponieważ stanowi swego rodzaju zabezpieczenie. Z tego powodu NFC wykorzystywane jest w dość wrażliwych i krytycznych z punktu widzenia rozwiązań, jak np. płatności zbliżeniowe czy systemy kontroli dostępu.

21. W jaki sposób systemy IoT minimalizują zapotrzebowanie na zasilanie.

Jednym z najważniejszych wymagań dla węzła czujnika IoT jest małe zużycie energii. W wielu przypadkach węzeł czujnika będzie komunikował się bezprzewodowo, co upraszcza jego montaż. Całe zasilanie węzła musi być zapewnione przez wbudowaną baterię lub pobrane z otoczenia. W celu zminimalizowania kosztów konserwacji wielu użytkowników chciałoby, aby bateria umożliwiała zasilanie urządzenia przez cały okres jego eksploatacji, który może wynosić pięć, dziesięć, a nawet więcej lat.

Zużycie energii w okresie eksploatacji czujnika IoT zależy od zużycia energii podczas aktywnego przetwarzania danych i komunikacji z siecią bezprzewodową oraz w okresach braku aktywności. Zużycie energii przez urządzenia wykorzystujące logikę CMOS, takie jak mikrokontroler, zależy w dużym stopniu od ładunku wymaganego do przełączania poszczególnych stanów. Należy także uwzględnić stałą utratę energii spowodowaną przez prąd upływu.

Energię zużywaną podczas przełączania można obliczyć na podstawie wzoru $C \times V^2 \times f$, gdzie C oznacza całkowitą pojemność ćwierćek obwodu w urządzeniu, V oznacza napięcie zasilania, a f częstotliwość roboczą. Energia tracona przez upływ zależy od użytej technologii przetwarzania i jest znacznie mniejsza od energii wymaganej do przełączania. Jednak ze względu na ciągły charakter upływu może on prowadzić do znacznej utraty energii w długim okresie.

Jedyną metodą zapobiegania upływowi jest całkowite wyłączenie zasilania obwodu. Wiele mikrokontrolerów jest zaprojektowanych do obsługi takiej funkcji. Gdy układ nie jest aktywny, stan uśpienia mikrokontrolera umożliwia odłączenie szyn zasilania od dużych fragmentów układu scalonego (IC) i zasilanie jedynie niektórych bloków funkcjonalnych, takich jak zegar czasu rzeczywistego. W energooszczędnym stanie uśpienia urządzenie zużywa o wiele mniej energii niż w stanie aktywnym.

Wiele zastosowań IoT może wykorzystać funkcję obsługi trybów uśpienia przez mikrokontroler. Odstępy pomiędzy okresami aktywności mogą być względnie długie. Każdy pomiar wykonywany przez czujnik trwa zwykle kilkaset mikrosekund, a często jest nawet jeszcze krótszy. Nawet w przypadku, gdy pomiar musi być wykonywany dziesięć razy na sekundę, węzeł czujnika może pozostawać uśpiony przez większą część swojego czasu eksploatacji.

Czas eksploatacji zależy w dużym stopniu od cyklu przełączania pomiędzy trybem aktywności a trybem uśpienia. W przypadku czasu aktywności wynoszącego 5 procent czasu eksploatacji, urządzenie może wybudzać się pięć razy na sekundę na 10 ms. Mikrokontroler zużywający 1 mA w stanie pełnej aktywności oraz obsługujący tryb głębokiego uśpienia, w którym zużywa 2 mA, może działać przez cztery lata dzięki zasilaniu baterią o pojemności 1800 mAh. Jeśli czas aktywności wzrośnie do 10%, czas eksploatacji ulegnie skróceniu o niemal połowę.

Projektanci mikrokontrolerów przeznaczonych do zastosowań IoT przywiązują ogromną wagę nie tylko do zredukowania energii zużywanej w trybie aktywności oraz w trybie uśpienia, ale również do zwiększenia stosunku czasu uśpienia do czasu największej aktywności. Szczytowa aktywność mikrokontrolera z reguły występuje w czasie, gdy centralny procesor przetwarza oprogramowanie.

GRAFIKA KOMPUTEROWA W GRACH

1. Scharakteryzuj dwie wybrane metody prototypowania graficznego. Podaj ich mocne i słabe strony.
2. Omów narzędzia modelowania i prototypowania graficznego, podaj ich mocne i słabe strony.
3. Wyjaśnij na czym polega przygotowanie obiektów do animacji.
4. Na czym polega przygotowanie wirtualnych przestrzeni gier.
5. Porównaj dwa wybrane narzędzia do przygotowania graficznej prezentacji gry.
6. Jakie czynniki mają wpływ na opracowanie koncepcji, analizy założeń i wymagań projektowych gry.
7. Przedstaw kolejne kroki procesu projektowania gry komputerowej.
8. Porównaj projektowanie gry jednopłatformowej z wielopłatformową.
9. Podaj różnice pomiędzy silnikami gier, a platformami gier.
10. Na czym polega dostosowanie projektu gry do różnych platform
11. Dokonaj porównania sprzętu (konsol, urządzeń mobilnych itp.) na których w rzeczywistych warunkach tworzy się gry.