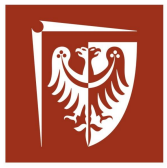


Politechnika
Wrocławska

Politechnika Wrocławska
Wydział Elektroniki

Obrona Inżynierska 2018

Opracował:
Jarosław Ciołek-Żelechowski

Na podstawie:
`https://github.com/december0123/obrona`
`http://lucc.pl/inf/egzamin_inzynierski`

22 stycznia 2018

Spis treści

1 K1 – Paradygmat programowania obiektowego

Paradygmat - zbiór mechanizmów działania, wzorów, które definiują sposób realizacji programu.

Programowanie obiektowe stanowi podejście do implementacji algorytmów, które opiera się na wykorzystaniu tak zwanych **obiektów**. Są to twory, które łączą w sobie *dane* (pola obiektu) i *zachowania* (metody obiektu) oraz komunikują się ze sobą w celu wykonania pewnych zadań.

Paradygmat ten ma stanowić ułatwienie w pisaniu i utrzymywaniu kodu, który może być używany wielokrotnie w różnych projektach jednak nie jest odpowiedzią na wszystkie problemy i oprócz swoich zalet ma również wady – np. przez to, że obiekty posiadają wewnętrzny stan, programowanie współbieżne staje się o wiele trudniejsze, gdyż musimy zapobiegać wyścigom czy zagłodzeniu.

Pojęciami, na których się skupię będą:

- Klasa,
- Abstrakcja,
- Enkapsulacja,
- Dziedziczenie,
- Polimorfizm,
- Wzorce projektowe.

1.1 Klasa

Stanowi pewien zbiór cech i zachowań danego obiektu, który jest jej instancją.

Przykładem może być klasa *Pies,* która zawiera cechy takie jak oczy, pysk oraz umiejętność szczekania.

Języki programowania można podzielić ze względu na to, w jaki sposób pojęcie klasy jest zrealizowane. Na przykład w językach *Java*, *C#* czy *Python* każda klasa zawsze ma swoją klasę bazową (zazwyczaj dziedziczona niejawnie i jest to przeważnie klasa bazowa *Object*). Język *C++* na przykład, można nazwać językiem hybrydowym, gdyż umożliwia tworzenie obiektów, lecz nie zmusza do tego programisty (można programować zarówno strukturalnie jak i obiektowo). Istnieją także języki, których obiektowość opiera się na **prototypach** - nowe obiekty tworzone są w oparciu na istniejące już obiekty, nie zaś na podstawie zdefiniowanej klasy. Przykładem takiego języka jest większość języków interpretowanych, np. *JavaScript*.

Głównym zyskiem obiektowości jest modularność - program można podzielić na mniejsze, gotowe do użycia moduły. Zwiększa to także czytelność kodu - łatwiej jest się odnaleźć w metodach przypisanych do obiektów niż w gąszczu funkcji. Łatwiejsze jest także rozbudowywanie projektów bez integracji w istniejące funkcjonalności.

1.2 Abstrakcja

Czyli poziom ogólności, pozwala nam na upraszczanie problemu poprzez zredukowanie właściwości do jedynie tych kluczowych dla algorytmu (ignorowanie zbędnych detali poprzez wyizolowanie kluczowych aspektów).

Dla przykładu - możemy mówić o *Bazie danych* jako o fragmencie kodu, który będzie stanowił interfejs do komunikacji z pewną bazą danych, jednak nie ma to dla nas większego znaczenia w jaki sposób to będzie realizowane.

Przekładając to na banalny przykład z życia codziennego – kierowca nie musi przechodzić szczegółowych szkoleń za każdym razem, gdy zmienia samochód. Wystarczy mu jedynie podstawowa wiedza o jego działaniu, a takie rzeczy jak znajomość budowy silnika nie są mu potrzebne.

1.3 Enkapsulacja

Inaczej hermetyzacja, polega na celowym ukrywaniu wnętrza obiektów tak, aby zmiany w jego stanie mogły dokonać tylko metody wewnętrzne tego obiektu. Zwiększa to bezpieczeństwo kodu oraz odporność na błędy, a także pozwala podzielić kod na mniejsze fragmenty.

Problem pojawia się, gdy udostępnimy wersję klasy, która ma pewne pola publiczne. Jeśli użytkownicy zaczną z tych pól korzystać, to aktualizacja może spowodować poważne problemy.

Enkapsulacja pomaga ustrzec nas przed niepożądanym korzystaniem z mechanizmów wewnątrz tworzonej przez nas klasy, a na użytek świata zewnętrznego pozwala nam wystawić tylko zdefiniowany przez nas interfejs.

Powracając do przykładu z samochodem – kierowca powinien otwierać okno za pomocą guzika lub korbki, a nie poprzez wybijanie szyby.

Warto wspomnieć, że języki w różny sposób podchodzą do omawianych przeze mnie pojęć. Python jest językiem, w którym pojęcie enkapsulacji praktycznie nie istnieje – wszystko jest dostępne dla wszystkich i jeśli ktoś bardzo chce użyć zmiennych, które wyraźnie oznaczyliśmy jako do użytku wewnętrznego to może to zrobić na własną odpowiedzialność.

Pełna enkapsulacja występuje wtedy, gdy wszystkie pola klasy znajdują się w sekcji prywatnej, a dostęp do nich możliwy jest tylko i wyłącznie poprzez metody.

1.4 Dziedziczenie

Jest narzędziem, które dzięki odpowiednio zaprojektowanej relacji **klasa bazowa - klasa pochodna** pozwala na rozszerzanie funkcjonalności bez duplikowania kodu. Dzięki temu możemy tworzyć hierarchiczne struktury przechodząc od typów najbardziej ogólnych aż do tych szczegółowo opisujących dany obiekt bądź zjawisko.

W odniesieniu do samochodów – możemy dostać wersję podstawową jakiegoś pojazdu, a

następnie ją rozszerzyć o stylowe neony, naklejki z ogniem i spoilery.

1.4.1 Dziedziczenie wielokrotne

Tutaj znowu w zależności od języka możemy mieć możliwość skorzystania z mechanizmu dziedziczenia wielokrotnego lub nie. Językami to umożliwiającymi są na przykład *C++* czy *Python*.

Polega ono na tym, że klasa pochodna może dziedziczyć po kilku klasach bazowych. Jest to potężny mechanizm, jednak należy go używać z głową, gdyż może powodować problemy takie jak np. *the diamond problem* (gdy pewna klasa X dziedziczy po dwóch klasach, które to mają wspólną klasę bazową i obie przeciążają tę samą metodę klasy bazowej, z której zatem klasy wywołana zostanie ta metoda, gdy X jej nie przeciąży?).

1.5 Polimorfizm

Pozwala nam na wyabstrahowanie pewnych zachowań od konkretnych typów danych. Dzięki niemu możemy wybrać zachowanie w zależności od kontekstu. Mówiąc ściślej, do jednej referencji można przypisać obiekty różnych typów, które dziedziczą po typie referencji, i wywołując przeciążoną metodę użyta zostanie metoda przypisanego typu referencji.

Dla odmiany rozpatrzmy przykład, w którym klasą **bazową** będzie **Zwierze,** a klasami **pochodnymi** będą **Pies** oraz **Ryba.** Klasa bazowa ma zadeklarowaną metodę (funkcję, którą możemy wywołać na rzecz obiektu) o nazwie *dajGlos*. Klasy pochodne mogą tę funkcję zdefiniować po swojemu i w ten sposób po wywołaniu metody *dajGlos* na obiekcie klasy *Pies* usłyszymy szczekanie, a po wywołaniu metody na obiekcie klasy *Ryba* usłyszymy tylko ciche bulgotanie, któremu towarzyszyć będzie bezgłośne osądzanie naszych wyborów życiowych.

Polimorfizm można podzielić na statyczny i dynamiczny. Polimorfizm **statyczny** (inaczej zwany *wczesnym wiązaniem*) – decyzja o użytym typie zostaje podjęta już na etapie kompilacji – w języku *C++* jest to realizowane za pomocą szablonów bądź przeciążonych operatorów. Polimorfizm **dynamiczny** (*późne wiązanie*) – wybór zostaje podjęty w czasie wykonywania programu – w języku *C++* zrealizowane przy użyciu wskaźników przeciążając metody wirtualne w klasach pochodnych.

1.6 Wzorce projektowe

Powstały, aby spisać często napotykane podczas programowania problemy oraz zdefiniować sprawdzone rozwiązania.

Jest to temat bardzo atrakcyjny dla początkujących programistów i jest jednocześnie bardzo pomocny i bardzo niebezpieczny, ponieważ początkujący mogą chcieć korzystać ze wzorców gdzie tylko mogą, nie zważając na to czy faktycznie są one potrzebne.

Należy pamiętać o antywzorcach złotym młotku oraz srebrnym pocisku – nie wszystko co się do tej pory sprawdziło gdzie indziej sprawdzi się w naszym przypadku, a to że doskonale znamy jakąś technologię nie znaczy, że jest ona zawsze odpowiednia.

Wzorce projektowe dzielimy wg trzech kategorii:

- konstrukcyjne – opisujące proces tworzenia nowych obiektów, przykładowo **fabryka, singleton,**
- strukturalne – opisujące struktury powiązanych ze sobą obiektów, przykładowo **adapter, dekorator,**
- behawioralne – opisujące zachowanie i odpowiedzialność współpracujących ze sobą obiektów, przykładowo **strategia, null object.**

2 K2 – Arytmetyka stało- i zmiennoprzecinkowa

Ludzie spodziewają się, że komputery będą dokładne i nieomylne. Spodziewają się, że po wpisaniu 0.1 + 0.2 kalkulator odpowie: 0.3.

Prędzej czy później każdy programista próbuje czegoś takiego i natrafia na problem: komputer twierdzi, że wynik tego dodawania to np. 0.30000001. Skąd takie rzeczy?

Każda informacja w komputerze jest przedstawiona za pomocą podstawowych jednostek – bitów. W przypadku liczb, można je reprezentować za pomocą typów {stało-,zmienno-}przecinkowych. Różnią się cechami takimi jak:

- sposób zapisu,
- zakres wartości,
- precyzja operacji.

Typy stałoprzecinkowe posiadają wiele reprezentacji.

2.1 Naturalny kod binarny (NBC)

Jest to system pozycyjny o podstawie 2. Liczby zapisane w tej reprezentacji nie posiadają znaku, nie można za jego pomocą zapisać ułamków. Wartość liczby można zaprezentować za pomocą wzoru (liczba *n-bitowa*):

$$x = \sum_{i=0}^{n-1} 2^i \times x_i \tag{1}$$

Na przykład:

DEC	BIN (NKB)
0	00
1	01
2	10
3	11

2.2 Kod uzupełnieniowy U2

W celu reprezentacji liczby całkowitej potrzebne jest zdefiniowanie znaku liczby. Wartość liczb zapisuje się podobnie jak w wypadku NKB, jednak znak liczby jest określony przez bit najbardziej znaczący (0 – liczba dodatnia, 1 – liczba ujemna). Aby otrzymać wartość n-bitowej liczby zapisanej w kodzie U2 można skorzystać z następującego wzoru:

$$x = -x_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} 2^i \times x_i \tag{2}$$

Jak wynika ze wzoru, o ile liczba dodatnia wygląda tak samo jak w NKB, to postać liczby ujemnej zapisanej w U2 wcale nie jest taka sama jak w NKB z dodaną jedynką na przodzie.

DEC	NKB	U2
-3	—	101
-2	—	110
-1	—	111
0	000	000
1	001	001
2	010	010
3	011	011

Dwa łatwe sposoby na odczytanie wartości liczby ujemnej w U2:

- Neguj wszystkie bity i dodaj 1.
- Potraktuj najstarszy bit jak ujemną wartość na tej pozycji i dodaj do niej pozostałe bity.

W dwójkowym systemie uzupełnieniowym porządek kodów arytmetycznych jest zachowany w całym zakresie. $\{0, x_{n-2}, \ldots, x_1, x_0\}$ reprezentuje dodatnią liczbę x, a $\{1, x_{n-2}, \ldots, x_1, x_0\}$ to zapis liczby **większej o x od najmniejszej liczby** -2^{n-1}.

Podobnie jak dla NKB, nie jest możliwe zastosowanie tutaj zapisu ułamków.

2.3 Typ stałoprzecinkowy

Reprezentacja ta jest podobna do kodu uzupełnieniowego, jednakże można tutaj określić pozycję przecinka. Dla liczb o podstawie β oraz ustalonej liczbie pozycji części ułamkowej – r, wartość każdej liczby jest podana z dokładnością β^{-r}. Można ją przedstawić za pomocą iloczynu liczby całkowitej, wtedy zapis wygląda następująco:

$$\{x_m, x_{m-1}, \ldots, x_1, x_0, \ldots, x_{-r}\} \tag{3}$$

Podczas operacji z liczbami stałoprzecinkowymi można otrzymać przeniesienie w wypadku operacji dodawania lub mnożenia.

2.4 Inne systemy stałoprzecinkowe

- system obciążony,
  - \+ unikatowa reprezentacja zera,
  - \+ zgodność uporządkowania liczb i ich reprezentacji,
  - \- wymagana korekcja wyników działań arytmetycznych,
  - \- problematyczne przy dzieleniu i mnożeniu.
- system ze znakowaną cyfrą.

2.5 Typ zmiennoprzecinkowy

2.5.1 Budowa

Przedstawione do tej pory systemy były **stało**przecinkowe. Znaczy to tyle, że miały z góry określoną precyzję i rozmiar.

Jedną z wad typów stałoprzecinkowych jest ilość miejsca potrzebna do przechowywania liczb: aby przechować liczbę z zakresu <0; 4 294 967 295> potrzebujemy aż 32 bitów pamięci.

Wyobraźmy sobie, że potrzebne nam są liczby rzędu rozmiaru wszechświata albo wielkości atomu.

Gdybyśmy takie liczby chcieli zapisać w typie stałoprzecinkowym to musielibyśmy zmarnować ogromne ilości pamięci na przechowywanie zer.

Rozwiązanie? Notacja naukowa i zmienny przecinek.

Liczba zmiennoprzecinkowa jest reprezentowana przez 4 liczby (S, β, M, E):

- S – znak,
- β – podstawa,
- E – wykładnik,
- M – mantysa.

$$x = (-1)^S \times \beta^E \times M \tag{4}$$

Łatwo zauważyć, że jest to po prostu notacja naukowa, lecz jak tak właściwie taka liczba wygląda?

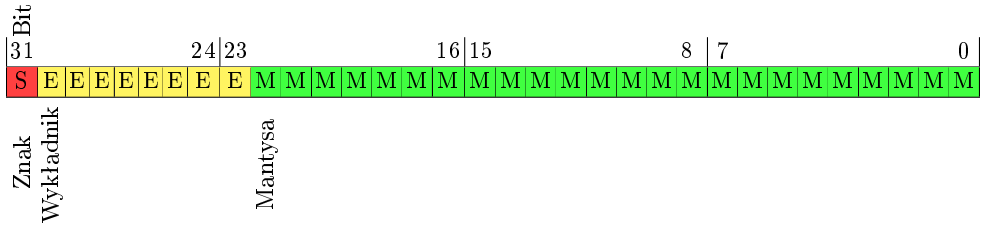
Pierwszy bit jest bitem znaku – 0 dla liczb dodatnich i 1 dla liczb ujemnych. Tak jest zawsze.

Kolejne bity to bity wykładnika i to ile tych bitów jest zależy od typu (precyzji). Dla liczby pojedynczej precyzji, czyli typu 32-bitowego, liczba bitów wykładnika wynosi 8.

Ważne: wykładnik jest zapisany w systemie obciążonym +N ($N = 2^{k-1} - 1$, gdzie k to liczba bitów wykładnika) – dla 32-bitowej liczby jest to system +127.

Pozostałe bity to liczby części ułamkowej. Ważne jest tutaj pojęcie liczby znormalizowanej, czyli takiej, dla której przyjmujemy, że część ułamkowa ma na początku ukrytą jedynkę – jest więc postaci 1.bb...bb, a nie 0.bb...bb. Więcej o tym za chwilę. **Ważne:** mantysa zawsze jest liczbą dodatnią.

Tablica 1: Liczba zmiennoprzecinkowa pojedynczej precyzji (32 bity)



Standard wyróżnia następujące precyzje:

- połowicza (16 bit),
- pojedyncza (32 bit),
- podwójna (64 bit),
- podwójna rozszerzona (80 bit).

Przykład: Liczba 17_{10} zapisana jako 32-bitowa liczba zmiennoprzecinkowa ma postać:

0|10000011|00010000000000000000000

czyli

- Znak – 0 -> liczba dodatnia
- Wykładnik – $10000011_2 == 2_{10}^{131-127} == 2_{10}^4$
- Mantysa – $00010000000000000000000_2 == 1.0625_{10}$

Po złożeniu wychodzi nam: $(-1)^0 * 2^4 * 1.0625 == 1 * 16 * 1.0625 == 17.0$

Jaka jest główna zaleta takiego zapisu? Ogromny wręcz zakres wartości możliwych do zapisania: korzystając z 32-bitowego typu zmiennoprzecinkowego możemy zapisać wartości z oszałamiającego zakresu <-3.40282e+38, 3.40282e+38>.

Powróćmy do pojęcia liczby znormalizowanej.

2.5.2 Liczby znormalizowane i zdenormalizowane

Standard IEEE754 wymaga, aby liczby były znormalizowane (oprócz wartości specjalnych) – mantysa powinna być w zakresie $1 \leq |M| < 2$. Liczbę taką można poznać po tym, że wykładnik jest **różny** od zera. Wtedy mantysa liczby jest postaci 1.bb...bb.

W wypadku, gdy wykładnik liczby jest równy zero, mówimy, że liczba jest zdenormalizowana, a jej mantysa ma postać 0.bb...bb.

Liczby zdenormalizowane są potrzebne do zapisania wartości tak bliskich 0, że ich ukryty bit w mantysie musi wynosić 0.

Będąc przy tym temacie warto wspomnieć o wartościach specjalnych.

2.5.3 Wartości specjalne

Liczby zmiennoprzecinkowe zawierają zarezerwowane wartości dla liczb specjalnych.

Wartości specjalne to:

- Zero – oprócz bitu znaku zawiera same zera (tak więc mamy tu zero dodatnie i ujemne).
- $\pm\infty$ – wszystkie bity wykładnika wynoszą 1, mantysa 0.
- Nie-liczba (NaN) – wykładnik również przyjmie same jedynki, natomiast mantysę $\neq 0$.
- Liczby zdenormalizowane posiadają w wykładniku same zera, natomiast mantysa jest postaci $0.bbb\ldots bb$.

Tablica 2: Liczby w standardzie IEEE754

Znak	**Wykładnik**	**Mantysa**	**Wartość**
0	$00\cdots00$	$00\cdots00$	$+0$
0	$00\cdots00$	$00\cdots01$ $\vdots$ $11\cdots11$	Dodatnia zdenormalizowana $(0.f \times 2^{-N+1})$
0	$00\cdots01$ $\vdots$ $11\cdots10$	$XX\cdots XX$	Dodatnia znormalizowana $(1.f \times 2^{E-N})$
0	$11\cdots11$	$00\cdots00$	$+\infty$
0	$11\cdots11$	$00\cdots01$ $\vdots$ $01\cdots11$	SNaN
0	$11\cdots11$	$1X\cdots XX$	QNaN
1	$00\cdots00$	$00\cdots00$	-0
1	$00\cdots00$	$00\cdots01$ $\vdots$ $11\cdots11$	Ujemna zdenormalizowana $(-0.f \times 2^{-N+1})$
1	$00\cdots01$ $\vdots$ $11\cdots10$	$XX\cdots XX$	Ujemna znormalizowana $(-1.f \times 2^{E-N})$
1	$11\cdots11$	$00\cdots00$	$-\infty$
1	$11\cdots11$	$00\cdots01$ $\vdots$ $01\cdots11$	SNaN
1	$11\cdots11$	$1X\cdots XX$	QNaN

N – obciążenie wykładnika, X – dowolna wartość, QNaN – quiet not-a-number (nie zgłaszają wyjątków), SNaN – signaling not-a-number (zgłaszają wyjątki)

Tablica 3: Specjalne operacje arytmetyczne

Operacja	**Wynik**
$\frac{n}{\pm\infty}$	0
$\pm\infty \times \pm\infty$	$\pm\infty$
$\frac{\pm X}{0}$	$\pm\infty$
$X \times \pm\infty$	$\pm\infty$
$\infty + \infty$ $\infty - -\infty$	$+\infty$
$-\infty - \infty$ $-\infty + -\infty$	$-\infty$
$\infty - \infty$ $-\infty + \infty$	NaN
$\frac{\pm 0}{\pm 0}$	NaN
$\frac{\pm\infty}{\pm\infty}$	NaN
$\infty \pm 0$	NaN

Standard wyróżnia również wyjątki:

- invalid operation — niewłaściwa operacja,
- division by zero — dzielenie przez zero,
- overflow — nadmiar – liczba jest za duża,
- underflow — niedomiar – utrata precyzji przy liczbach bliskich zero,
- inexact — niedokładność – podczas operacji zaokrąglania.

2.6 Działania

2.6.1 Arytmetyka stałoprzecinkowa

- **Dodawanie/odejmowanie**
  - dodawanie kolejnych cyfr, z zachowaniem ich pozycji
  - występują przeniesienia
  - odejmowanie – dodawanie liczby przeciwnej (odjętej od zera)
- **Mnożenie**
  - mnożenie mnożnej kolejnymi cyframi mnożnika, potem dodanie kolejnych sum częściowych
  - w systemie uzupełnieniowym jest podobnie jak w naturalnym, gdy mnożnik ujemny wymagana jest korekcja wyniku
- **Dzielenie**
  - wynik dzielenia jest przybliżony (nie ma gwarancji, że dla dzielnej X oraz dzielnika D istnieje liczba Q, że $X = Q * D$, można to skorygować dodając resztę z dzielenia)

– właśnie ogarnąłem, że opisałem fakty, których uczyliśmy się w podstawówce

2.6.2 Arytmetyka zmiennoprzecinkowa

W przypadku liczb zmiennoprzecinkowych ważna jest kolejność działań! Wynika to z błędów precyzji i zaokrągleń.

- **Dodawanie/odejmowanie**
  - wymaga wyrównania wykładników (mniejszy do większego)
  - denormalizacja jednej liczby, następnie dodanie/odjęcie mantys, potem normalizacja wyniku
  - nadmiar/niedomiar może wystąpić przy normalizacji
- **Mnożenie/dzielenie**
  - mnożenie – przemnożenie mantys, dodanie wykładników, normalizacja
  - dzielenie – podzielenie mantys, odjęcie wykładników, normalizacja
  - może wystąpić nadmiar/niedomiar

2.6.3 Zaokrąglanie

Na koniec powiedzmy sobie o zaokrągleniach liczb.

Są one potrzebne, ponieważ w systemie binarnym nie zawsze da się dokładnie zapisać wszystkie liczby (ten problem występuje również w innych systemach).

Przykładowo, liczba dziesiętna $\frac{1}{10}_{10} == 0.1_{10}$ daje się łatwo zapisać w systemie dziesiętnym, jednak w binarnym jest ona ułamkiem okresowym $0.1(0011)_2$. Ze względu na naturę komputerów, pamięć jest ograniczona, a co za tym idzie nie jesteśmy w stanie zapisać liczb z nieskończoną precyzją.

Kiedy komputer napotyka liczbę, której nie jest w stanie dokładnie zapisać, zaokrągla ją.

Zanim przejdziemy do omawiania konkretnych zaokrągleń, porozmawiajmy o trzech bitach, które są ukryte za mantysą: G,R,S.

Bity G(uard)R(ound)S(ticky) istnieją dla celów zaokrągleń i przechowywane są w nich bity niemieszczące się w mantysie. Już tłumaczę.

Jeśli chcemy naszą mantysę podzielić przez np. 8, czyli przesunąć bity w prawo o 3 pozycje, to wygląda to następująco:

mantysa |GRS

000111010|000

xxx000111|010

001000111|010

Skąd wzięło się 001 na miejscu xxx i czym w końcu są te bity GRS?

Bity 001 na początku wzięły się stąd, że mieliśmy do czynienia z liczbą znormalizowaną, więc na początku stała ukryta 1, a że mantysa zawsze jest dodatnia, to przed nią stoi nieskończenie wiele zer. Jasne? Jasne.

Bity G oraz R to po prostu dwa dodatkowe bity używane przez implementację do przechowywania dodatkowych wartości.

Bit S mówi nam czy po prawej od bitów G oraz R może znajdować się jeszcze jakaś '1' – jeśli jest możliwe, że wśród utraconych bitów była jakaś '1' to bit jest ustawiany na '1', a w przeciwnym wypadku na '0'.

Dobrze, rozpatrzmy różne zaokrąglenia na przykładzie liczby $0.1(0011)_2$

Zaokrąglanie do zera

Na początku wybieramy ile miejsca możemy poświęcić na zapisanie naszej liczby, a resztę kasujemy. Bitami GRS w ogóle się nie przejmujemy. Wybierzmy 4 miejsca po przecinku.

$0.10011001100110011...0011_2 \Rightarrow 0.1001_2$

Już. Proste, prawda?

Zaokrąglanie do $+\infty$

Niezależnie od wartości, zawsze zaokrąglamy w górę.

Zaokrąglanie do $-\infty$

Niezależnie od wartości, zawsze zaokrąglamy w dół – czyli dla ujemnych wartości, liczba będzie jeszcze mniejsza.

Zaokrąglanie do najbliższej, w kierunku parzystej

Jest to domyślny tryb zaokrąglania w IEEE-754. Przy tym zaokrąglaniu bity GRS zaczynają mieć znaczenie. Jeżeli $GRS \geq 101$ (czyli 101, 110 lub 111), to zaokrąglamy w górę – dodajemy 1 do *ulp* (ang. *unit in the last place*, najmłodszy bit). Jeżeli nastąpi przepełnienie mantysy (wyjdzie poza zakres $1 \leq M < 2$), to należy ją znormalizować oraz zwiększyć wykładnik. Jeśli $GRS = 100$ to mamy dwie sytuacje:

- $ulp = 1 \Rightarrow$ zaokrąglamy w górę
- $ulp = 0 \Rightarrow$ nic nie robimy

Zostaje jeszcze ostatnia możliwość: $GRS = 0xx$ (bity R i S nie mają znaczenia) – wtedy nie zmieniamy mantysy.

Zaokrąglanie do najbliższej, w kierunku nieskończoności

Zaokrąglanie to działa podobnie do powyższego dla liczb dodatnich, dla liczb ujemnych zaokrąglenie w dół. Zostało stworzone dla systemu o podstawie dziesiętnej.

2.7 Podsumowanie

Cecha	L. stałoprzecinkowe	L.zmiennoprzecinkowe
Reprezentacja	Tak jak w systemie pozycyjnym lub uzupełnieniowym	Reprezentacja przez: znak, wykładnik, mantysę
Dokładność	Mały zakres, dokładne wyniki	Duży zakres, możliwe zaokrąglenia, utrata precyzji
Obsługa błędów	Może wystąpić przepełnienie	Zaokrąglenia, błędne operacje, przepełnienie, niedomiar
Wartości specjalne	—	+0, -0, $+\infty$, $-\infty$, QNaN, SNaN, liczby zdenormalizowane
Arytmetyka	Wszystkie twierdzenia prawdziwe	Kolejność działań wpływa na wynik

3 K3 – Normalizacja schematu bazy danych

Normalizacja schematu bazy danych, czyli sprowadzanie schematu do jednej z postaci normalnych jest dokonywana, aby przeciwdziałać lub też zapobiegać problemom, które pojawiają się podczas cyklu życia bazy danych.

Przykładowe problemy związane z użytkowaniem i utrzymaniem bazy danych:

- brak spójności danych,
- zbyt skomplikowane zapytania,
- anomalie spowodowane aktualizacją danych.

Normalizacja najczęściej sprowadza się do rozbijania tabel na mniejsze, jednak co ważne, nie wpływa ona na dane, a jedynie na sposób w jaki je przechowujemy.

Sprowadzenie schematu do którejś z postaci normalnych polega na sprawieniu, by schemat spełniał warunki określone przez postać do której dążymy oraz przez wszystkie poprzednie.

Przykładowo: schemat jest w 3NF jeśli spełnia warunki 3NF, 2NF oraz 1NF.

Pan Codd, który był twórcą pojęcia normalizacji, wymyślił trzy postaci normalne, z których trzecia jest przez większość uważana za wystarczającą, jest ich jednak więcej.

Zanim omówimy poszczególne postaci, chciałbym wspomnieć o jednej ważnej rzeczy – **normalizacja bazy danych jest bardzo dobrą praktyką, która nie zawsze jest jednak konieczna, a czasami może być wręcz niepożądana.**

Wynika to z tego, że po rozbiciu na wiele tabel, system bazodanowy musi wykonywać złączenia przy pozyskiwaniu danych, co może niekorzystnie wpływać na wydajność systemu.

1NF

Określa podstawowe zasady, które musi spełnić każda dobrze zorganizowana baza danych:

1. Tabele nie posiadają powtarzających się kolumn.
2. Dane w każdej kolumnie są niepodzielne.
3. Każdy wiersz daje się jednoznacznie zidentyfikować (klucz główny).
4. Kolejność wierszy i kolumn nie ma znaczenia.
5. Dane w jednej kolumnie są tego samego rodzaju.

Czy poniższa tabela spełnia warunki 1NF?

Tablica 4: Transakcje - przed normalizacją

Klient	Towar
Adam Adamowicz	Lalka barbie
Anna Annowska	Lalka barbie, Wino grzane

Widzimy, że tabela nie jest w 1NF, ponieważ kolumna *Klient* zawiera dane, które da się rozbić na dwie mniejsze kolumny – imię oraz nazwisko klienta. Brakuje również unikalnego identyfikatora wierszy. Widzimy też, że w kolumnie *Towar* pojawia się wiele wartości, co jest błędem.

Tabela po sprowadzeniu do 1NF wygląda następująco:

Tablica 5: Transakcje - 1NF

Id	Imię	Nazwisko	Towar
1	Adam	Adamowicz	Lalka barbie
2	Anna	Annowska	Lalka barbie
3	Anna	Annowska	Wino grzane

2NF

Druga postać normalna jeszcze bardziej zagłębia się w usuwanie redundancji:

1. Spełnia warunki 1NF.
2. Atrybuty **niekluczowe** zależą funkcyjnie od **pełnego** klucza.

Co to tak właściwie oznacza?

Znaczy to tyle, że jeśli któraś z kolumn niekluczowych zależy tylko od części klucza, to jest to błąd.

Załóżmy tabelę, w której klucz jest złożony z kolumn **id_kursu** oraz **id_semestru**:

Tablica 6: Kursy - 1NF

id_kursu	id_semestru	sala	nazwa_kursu	id_prowadzacego	nazwisko_prowadzacego
I1	2015-16-z	10	Programowanie	1	Nauczycielowicz
I1	2015-16-l	10	Programowanie	1	Nauczycielowicz
E1	2015-16-z	15	Elektronika	2	Prowadzącywicz

Widzimy, że kolumna *nazwa_kursu* zależy tylko od części klucza - *id_kursu,* ponieważ nazwa kursu zależy od jego id, ale nie ma nic wspólnego z semestrem, w którym się odbywa. Kolumny *sala, id_prowadzacego* oraz *nazwisko_prowadzacego* nie naruszają warunków 2NF, ponieważ

- kolumny *sala* oraz *id_prowadzacego* zależą od całości klucza – prowadzący i sala zmienia się zarówno ze względu na przedmiot jak i na semestr,
- kolumna *nazwisko_prowadzacego* wcale nie zależy od klucza.

Aby sprowadzić tabelę do 2NF należy rozbić ją w następujący sposób:

Tablica 7: Kursy - 2NF

id_kursu	id_semestru	sala	id_prowadzacego	nazwisko_prowadzacego
I1	2015-16-z	10	1	Nauczycielowicz
I1	2015-16-l	10	1	Nauczycielowicz
E1	2015-16-z	15	2	Prowadzącywicz

Tablica 8: Nowa tabela wyekstrahowana z tabeli Kursy - 2NF

id_kursu	nazwa_kursu
I1	Programowanie
E1	Elektronika

3NF

Trzecia postać normalna idzie o krok dalej:

1. Spełnia warunki 2NF.
2. Atrybuty **niekluczowe** zależą **tylko** od **pełnego** klucza.

Różnica między 2NF i 3NF brzmi subtelnie, a polega na tym, że z tabeli wyciągamy wszystkie dane, które w żaden sposób nie zależą od klucza.

Tabela *Kursy* sprowadzona do 3NF wygląda następująco:

Tablica 9: Kursy - 3NF

id_kursu	id_semestru	sala	id_prowadzacego
I1	2015-16-z	10	1
I1	2015-16-l	10	1
E1	2015-16-z	15	2

Tablica 10: Nowa tabela wyekstrahowana z tabeli Kursy - 3NF

id_prowadzacego	nazwisko_prowadzacego
1	Nauczycielowicz
2	Prowadzącywicz

4 K4 – Model warstwowy TCP/IP

Model TCP/IP jest modelem warstwowej struktury protokołów komunikacyjnych. Dzięki niemu dokonywana jest współpraca między różnym rodzajem sprzętu, technologii sieciowych oraz oprogramowania. Funkcje sieci komputerowych są podzielone na grupy, które są realizowane na różnych warstwach.

Stopień skomplikowania sieci komputerowych spowodował, że różne funkcje sieci należy podzielić na grupy (warstwy). Każda warstwa ma swoją określoną rolę. Nawiązanie połączenia między np. serwerami pocztowymi (pracują na tej samej warstwie) na dwóch komputerach w rzeczywistości odbywa się przez wykorzystanie warstw niższych.

W latach osiemdziesiątych stworzono model ISO/OSI. Został zaprojektowany jako „otwarty" model – opublikowany za darmo, bez restrykcji patentowych bądź ograniczeń rozpowszechniania. Aktualnie model ten jest traktowany jako wzorzec dla protokołów komunikacyjnych. Posiada on siedem warstw:

7	aplikacji
6	prezentacji
5	sesji
4	transportowa
3	sieciowa
2	łącza danych
1	fizyczna

Warstwa fizyczna zapewnia przekaz bitów między stacjami połączonymi fizycznym medium (kable miedziane, światłowody, łącza radiowe).

Warstwa łącza danych umieszcza dane w ramkach, które zawierają ciągi kontrolne. Warstwa ta również sprawdza jakość przekazywanych informacji, próbuje naprawić ewentualne błędy.

Warstwa sieciowa jest odpowiedzialna za ustalenie drogi do docelowego urządzenia. Jednostką danych w tej warstwie są pakiety. Występuje tutaj segmentacja danych.

Warstwa transportowa ma nadzór nad połączeniem (inicjacja, zrywanie) między dwoma stacjami. Warstwa ta jest również odpowiedzialna za podział danych na bloki, kontrolę poprawności transmisji, rozpoznawanie duplikatów oraz sprawdzanie poprawności adresowania.

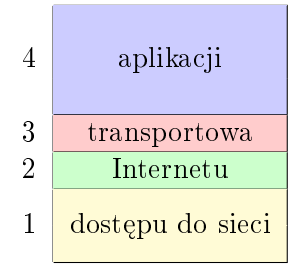
Warstwa sesji synchronizuje przesył danych, wznawia go po przerwaniu połączenia.

Warstwa prezentacji przekształca dane użytkowników do postaci standardowej, używanej w sieci (np. zamiana Little Endian na Big Endian). Innymi cechami tej warstwy jest kompresja oraz szyfrowanie danych.

Warstwa aplikacji zapewnia obsługę użytkowników w dostępie do usług (transmisja plików, zdalny terminal, poczta, etc.).

Koncepcja modelu TCP/IP jest identyczna jak w modelu OSI/ISO. Obecnie jest wyko-

rzystywany jako struktura Internetu. Architektura tego modelu została opracowana w celu umożliwienia komunikacji między systemami pochodzącymi od różnych dostawców. W porównaniu do modelu OSI posiada mniej warstw, analogiczne warstwy zostały zaznaczone identycznym kolorem, jak w poprzedniej tabeli.



Warstwa dostępu do sieci może zawierać protokoły dynamicznego przydzielania adresów IP. Protokoły działające w tej warstwie: Ethernet, WiFi.

Warstwa Internetu przetwarza dane posiadające adresy IP. Protokół działający w tej warstwie: IP.

Warstwa transportowa wykorzystuje protokoły TCP oraz UDP. Przesyłanie danych do odpowiednich aplikacji odbywa się za pomocą portów określonych dla każdego połączenia.

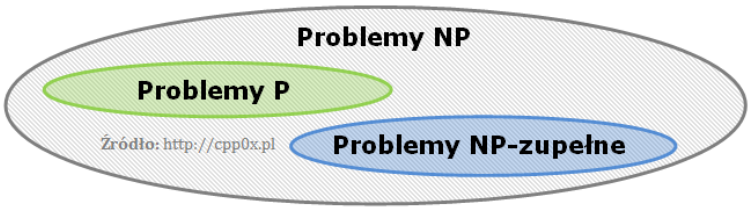
Warstwa aplikacji wykorzystuje różne protokoły (HTTP, XMPP, FTP, IRC, SSH, etc.).

5 K5 – Ocena złożoności algorytmów

Złożoność algorytmu jest miarą ilości zasobów potrzebnych do rozwiązania danego problemu o określonym rozmiarze. Dla przykładu w algorytmie rozkładu liczb na czynniki pierwsze zauważyć można, że im większa liczba, tym dłużej będziemy ją rozbijać i wykonamy więcej obliczeń. Faktem zatem jest, że im większy rozmiar danych wejściowych, tym więcej zasobów potrzebujemy do rozwiązania problemu. Złożoność algorytmu można więc określić mianem **funkcji rozmiaru danych wejściowych**.

Wyróżnić możemy kilka **klas złożoności**, czyli grup zagadnień o podobnej złożoności obliczeniowej. Są to:

1. **NP** (*nondeterministic polynomial*) - problemy decyzyjne rozwiązywalne niedeterministycznym algorytmem wielomianowym.
2. **P** (*deterministic polynomial*) - problemy decyzyjne, które można rozwiązać deterministycznym algorytmem o złożoności wielomianowej.
3. **NP-zupełny** - problemy decyzyjne, których znalezienie rozwiązania nie jest możliwe w czasie wielomianowym.
4. **NP-trudny** (silnie NP-zupełny) - samo sprawdzenie rozwiązania problemu jest co najmniej tak trudne jak każdego innego problemu NP.



Rysunek 1: Klasy złożoności

Aby określić złożoność stosowane są metody **oceny złożoności**. W celu ujednolicenia wyników i uniezależnienia się od środowiska, ocena złożoności rozpatrywana jest modelem abstrakcyjnym nie uwzględniającym platformy czy mocy obliczeniowej maszyny. Wyróżniamy następne modele złożoności:

1. **Czasowe** - czas wykonywania algorytmu wynikająca z liczby operacji elementarnych wykonywanych podczas przebiegu programu.
2. **Pamięciowe** - pamięcio-żerność oraz zasobo-żerność - ilość pamięci operacyjnej, bądź przestrzeni dyskowej potrzebnej do rozwiązania problemu. Zależy od liczby oraz złożoności użytych struktur.

Przy wyborze algorytmu zazwyczaj decydująca jest złożoność czasowa - o wiele bardziej miarodajny model.

Wyróżniamy także podział złożoności algorytmów ze względu na instancję problemu:

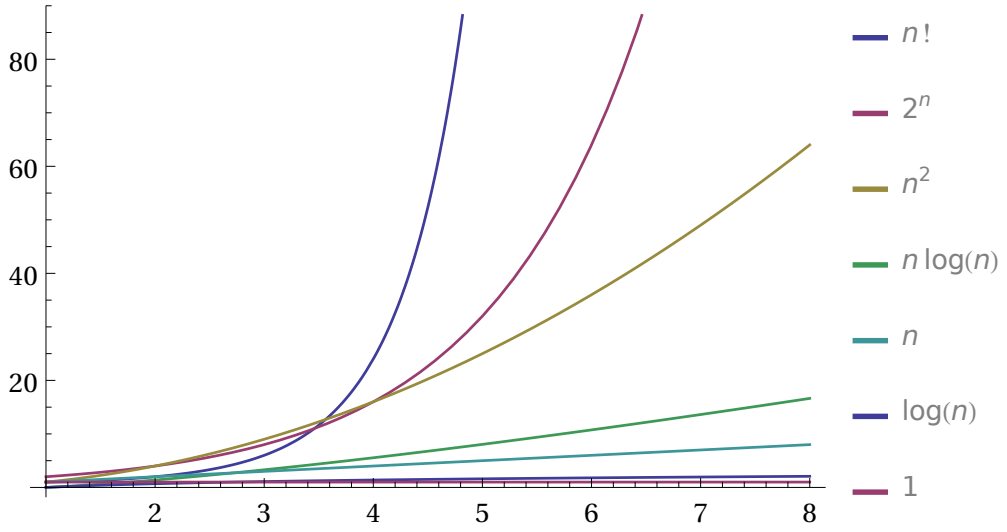
1. **Optymistyczna** - najlepszy możliwy przypadek (minimalna ilość wykonywanych operacji).
2. **Pesymistyczna** - najgorszy scenariusz (wykonywane są wszystkie operacje).
3. **Średnia** - wartość oczekiwana wykonywania algorytmu.

Do oceny złożoności algorytmu potrzebny jest **rząd wielkości** oraz zmienność zależnie od instancji problemu. Nie potrzebne są dokładne wartości, dlatego też stosuje się **asymptotyczne tempo wzrostu**, które to informuje nas o zmianie (wzroście bądź spadku) złożoności algorytmu w zależności od ilości danych wejściowych. Opisuje jak szybko funkcja rośnie, maleje, bądź pozostaje stała.

Asymptotyczne tempo wzrostu określane jest za pomocą *notacji asymptotycznych*. Istnieje kilka notacji *Bachmann-Landaua*:

- $\mathcal{O}$ (omikron, duże O) - Funkcja asymptotycznie **niewiększa**. Jest to ograniczenie górne, czyli taka funkcja, dla której wszystkie wartości funkcji badanej, będą od niej niewiększe (mniejsze bądź równe).
- Ω (omega) - Funkcja asymptotycznie **niemniejsza**. Jest to granica dolna, czyli funkcja, dla której wszystkie wartości funkcji badanej, będą od niej niemniejsze (większe bądź równe).
- Θ (theta) - Funkcja asymptotycznie **podobna**. Jest to oszacowanie dokładne, czyli funkcje ograniczające badaną funkcję od góry i od dołu.
- o - (małe O) - Funkcja asymptotycznie mniejsza. Rzadko stosowana.
- ω - (mała omega) - Funkcja asymptotycznie większa. Rzadko stosowana.

Złożoności kategoryzuje się do rzędów złożoności. Określają one jak szybko złożoność algorytmu rośnie wzraz ze wzrostem instancji problemu.



Rysunek 2: Porównanie rzędów złożoności

Wyróżnić można (n - *liczba danych wejściowych*):

- $\mathcal{O}(1)$ - **złożoność stała** - liczba operacji jest niezależna od rozmiaru problemu. Przykład: Ustalenie, czy liczba binarna jest dodatnia bądź ujemna, obliczenie $(-1)^n$, odwołanie się do n-tego elementu stałej w rozmiarze tablicy.
- $\mathcal{O}(\log n)$ - **logarytmiczna** - liczba operacji rośnie proporcjonalnie do logarytmu z rozmiaru problemu. Przykład: Wyszukiwanie binarne w posortowanej tablicy, dodanie/usunięcie elementu z drzewa binarnego.
- $\mathcal{O}(n)$ - **liniowa** - liczba operacji jest wprost proporcjonalna do rozmiaru problemu. Przykład: Znalezienie min/max elementu w nieposortowanej tablicy, dodanie dwóch n-bitowych liczb całkowitych w sumatorze z przeniesieniem szeregowym.
- $\mathcal{O}(n \log n)$ - **quasi-liniowa** - liczba operacji proporcjonalna do iloczynu rozmiaru problemu i jego logarytmu. Sortowania przez porównania (MergeSort (przez scalenie), HeapSort (kopcowe), QuickSort (szybkie)), FFT (Szybka Transformata Fouriera).
- $\mathcal{O}(n^2)$ - **kwadratowa** - liczba operacji rośnie proporcjonalnie do kwadratu rozmiaru problemu. Przykład: Prosty algorytm mnożenia dwóch n-bitowych liczb, sortowania: BubbleSort (bąbelkowe), SelectionSort (przez wybieranie), InsertionSort (przez wstawianie), znalezienie najkrótszej ścieżki w grafie.
- $\mathcal{O}(n^k)$ - **wielomianowa** - liczba operacji rośnie proporcjonalnie do wielomianu rozmiaru problemu. Przykład: Programowanie liniowe, maksymalne skojarzenie dla grafu dwudzielnego.
- $\mathcal{O}(2^n)$ - **wykładnicza** - liczba operacji rośnie proporcjonalnie do wartości wykładniczej rozmiaru problemu. Przykład: Programowanie dynamiczne dla problemu komiwojażera.
- $\mathcal{O}(n!)$ - **rzędu silnia** - liczba operacji rośnie proporcjonalnie do silni rozmiaru problemu. Przykład: Algorytm brute-force dla problemu komiwojażera.

6 K6 – Język UML w projektowaniu oprogramowania

UML w kontekście projektowania oprogramowania jest językiem pozwalającym na modelowanie i opisywanie systemów i ich części.

Pozwala na standaryzowany, graficzny zapis systemu z uwzględnieniem zarówno jego części konceptualnych, takich jak funkcje i procesy biznesowe, jak i obiektów fizycznych jakimi mogą być bazy danych czy warstwa sprzętowa.

W jaki sposób używa się UMLa do modelowania? Korzysta się z diagramów, które są podzielone ze względu na modelowanie strukturalne – diagramy pakietów, klas, komponentów itd – oraz behawioralne – diagramy sekwencji, przypadków użycia, stanu itd.

UML jest przydatnym narzędziem nie tylko w procesie modelowania, ale również podczas szybkich rozmów między programistami kiedy chce się szybko nakreślić ideę.

Omówmy kilka najczęściej spotykanych diagramów.

6.1 Diagramy strukturalne

6.1.1 Diagram pakietów

Pełni rolę organizacyjną. Zawiera w sobie inne elementy języka, zazwyczaj diagramy klas. Jest to ogólna wizualizacja systemu, z przedstawieniem zależności pomiędzy jego częściami.

6.1.2 Diagram klas

Składa się z zamodelowanych klas oraz relacji między nimi.

Pojedynczą klasę umieszcza się w prostokącie podzielonym na trzy części

- nazwę,
- atrybuty,
- funkcje.

Atrybuty oraz funkcje oznacza się modyfikatorem dostępu (np. + dla public, - dla private), składniki statyczne zaznacza się poprzez podkreślenie, a stereotypy – elementy służące do doprecyzowania semantyki elementu, np. oznaczenie jako klucz główny – umieszcza się w podwójnych nawiasach ostrych «».

Relacje między klasami:

- **Zależność – strzałka przerywana** – podstawowa relacja mówiąca, że obiekt może w jakiś sposób korzystać lub wpływać na inne obiekty.

- **Asocjacja – linia ciągła** – obiekty wykorzystują inne obiekty (człowiek używa magazynu do przechowywania rzeczy).
- **Agregacja częściowa – pusta strzałka z rombem** – obiekt posiada inne obiekty, ale nie są one do niego przypisane na wyłączność (kiedy człowiek umrze, jego samochód dalej istnieje).
- **Agregacja całkowita – pełna strzałka z rombem** – obiekt posiada inne obiekty i jest za nie odpowiedzialny (kiedy człowiek umrze, jego serce umiera wraz z nim).
- **Dziedziczenie – pusta strzałka** – określa hierarchię dziedziczenia.

6.1.3 Diagram wdrożenia

Pokazuje on strukturę fizyczną systemu, z uwzględnieniem obiektów systemu oraz elementami zewnętrznymi (np. maszyna obliczeniowa, czy też baza danych). Diagram ten jest przydatny przy wdrażaniu dużych systemów.

6.2 Diagramy behawioralne

6.2.1 Diagram przypadków użycia

Diagram umożliwia analizę oraz dokumentację wymagań co do funkcjonalności systemu – w jasny sposób pokazuje co jest od niego wymagane i pomaga w opracowaniu projektu oraz przetestowaniu już zbudowanego systemu. Co ważne, diagram nie posiada szczegółowych informacji na temat przypadków użycia – takie informacje są umieszczane np. na diagramie stanu czy aktywności.

Diagram składa się z

- przypadków użycia – ciąg akcji, z uwzględnieniem ich różnych wariantów, które mogą zostać wykonane podczas interakcji systemu z użytkownikiem,
- aktorów – mogą być to ludzie-użytkownicy, ale również inne systemy czy urządzenia,
- związków – powiązanie pomiędzy elementami diagramu.

6.2.2 Diagram sekwencji

Pokazuje w sposób zgodny z intuicją kolejność wywołanych operacji i przepływ sterowania pomiędzy obiektami.

Diagram zbudowany jest z

- prostokątów, które oznaczają obiekty,
- pionowych linii życia tychże obiektów,
- komunikatów wymienianych między obiektami.

Czas jest reprezentowany w postaci pionowej osi diagramu, a zajętość obiektu jest symbolizowana przez prostokąt umieszczony na jego linii życia.

Do komunikatów wymienianych między obiektami zalicza się między innymi

- wywołanie funkcji,
- powrót z wywołania,
- wywołanie asynchroniczne.

Na tym diagramie można również umieścić bloki, które przedstawiają specjalne interakcje (np. pętle, alternatywne przejście, sekcję krytyczną).

6.2.3 Diagram komunikacji

Jest to diagram podobny do diagramu sekwencji, jednak skupia się on na obiektach, które są składnikiem interakcji oraz komunikatów, które przesyłają. W odróżnieniu od diagramu sekwencji, nie skupia się na aspekcie czasowym. Rozmieszczenie obiektów jest kluczowe, aby w jasny sposób przedstawić relacje między sobą. Komunikacja jest przedstawiona za pomocą linii łączącej obiekty, natomiast przesyłane dane znajdują się ponad nimi (np. student zalicza egzamin).

7 K7 – Generowanie realistycznych obrazów scen 3-D za pomocą metody śledzenia promieni

Fotorealizm jest to metodyka starająca się w jak najlepszym stopniu oddać świat rzeczywisty za pomocą wygenerowanego obrazu. Aby to osiągnąć wykorzystuje się różne metody takie jak *fotogrametria* (łączenie zdjęć tego samego obiektu z wielu różnych perspektyw w celu wygenerowania obiektu 3-D - technologia użyta np. przy tworzeniu gry *The Vanishing of Ethan Carter*), metody próbkowania przestrzeni, czy też metoda śledzenia promieni.

Metoda próbkowania przestrzeni polega na analizowaniu toru promieni od źródła światła, poprzez odbicia od obiektów sceny, aby finalne trafić do oka obserwatora albo wylecieć poza scenę. Z każdego źródła światła wyprowadzany jest pęk promieni (**dyskretyzacja światła**), i im więcej promieni, tym dokładniejszy stanie się obraz. Tutaj pojawia się problem, gdyż generowana jest bardzo duża liczba promieni, z czego ogromna większość nie przecina nawet rzutni (nie dociera do oka obserwatora), przez co operacje wykonane zostały bezsensownie. Stąd też wymyślona została metoda śledzenia promieni rozwiązująca ten problem.

Metoda śledzenia promieni (ang. *Ray Tracing*) jest techniką służącą do generowania fotorealistycznych scen trójwymiarowych 3-D. Opiera się na śledzeniu wyłącznie tych promieni, które docierają do obserwatora. Cechą charakterystyczną jest to, że promienie nie są analizowane normalnym torem, tj. od źródła światła, bądź światła odbitego, do obserwatora, lecz właśnie **od oka obserwatora do elementów na scenie**. Zatem dla każdego piksela obrazu wynikowego wyprowadzany jest jeden promień, od którego w następstwie zależy wartość koloru tego piksela.

Wyprowadzony promień może nie trafić w żaden obiekt na scenie - piksel przyjmuje wtedy określony kolor tła. Promień może także trafić na źródło światła - piksel zyskuje kolor źródłowy światła. Promień może również trafić w jakiś obiekt na scenie. Jeśli trafi na taki obiekt, to wyznaczane są **punkty przecięcia**, następnie brany jest punkt przecięcia najbliższy początkowi promienia (gdyż punktów przecięcia dla obiektu może być kilka) i dla niego obliczany jest kolor punktu za pomocą wybranego modelu oświetlenia, na przykład popularnego **modelu Phonga**. Obliczane są w tym momencie także cienie używając pomocniczych promieni biegnących do źródła światła - gdy wiązka przetnie inny obiekt to oryginalny punkt jest zaciemniony. Całą procedurę można następnie powtarzać rekurencyjnie śledząc kolejne promienie odbite (zwane wtórnymi) i załamane tak, aby uzyskać efekt przedmiotów odbijających się w sobie nawzajem.

Przebieg działania algorytmu zapisywany jest w postaci **drzewa** (graf nieskierowany). Skrócony przebieg algorytmu przedstawić można w kilku krokach:

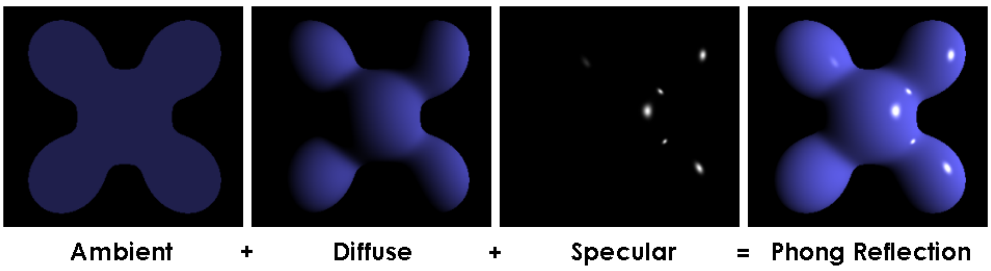
1. Dla każdego piksela obrazu wyprowadź **promień pierwotny**.
2. Dla każedgo napotkanego obiektu oblicz odbicia i wyprowadź **promienie wtóre**. Każdy punkt odbicia zapisz jako nowy węzeł drzewa.
3. Dla każdego węzła wyznacz oświetlenie lokalne korzystając z wybranego modelu oświetlenia.
4. Przechodząc od liści drzewa dodawaj kolejne wartości oświetlenia lokalnego ustalając tym samym ostateczną wartość piksela, od którego wyszedł promień pierwotny.

Algorytm kończy swe działanie w momencie, gdy:

- Promień nie trafia w żaden obiekt na scenie
- Promień trafia w obiekt całkowicie rozpraszający światło
- Promień trafia na obiekt, w którym następuje całkowite wewnętrzne odbicie
- Wyczerpana została ilość rekursywnych odbić

Model Phonga służy do wyznaczania oświetleń lokalnych. Na model ten składają się trzy rodzaje oświetlenia:

1. **Światło ambientowe** - światło otoczenia, jest to wartość stała przypisana do danej sceny.
2. **Światło rozproszone** - podstawowy kolor obiektu, wyliczany na podstawie **modelu Lamberta** (*Lambert, Lambert ty chuju* - Wiedźmin Geralt z Rivii) i widoczny przede wszystkim na powierzchniach matowych (drewno, papier).
3. **Światło odbite** - odpowiada za efekt połysku dla powierzchni śliskich, błyszczących (Passat metallic nówka nie śmigana trzy razy klepana). Wskazuje kierunek promienia odbitego.



Rysunek 3: Model Phonga

Owe składowe są mnożone przez procentowy współczynnik wpływu każdej składowej a następnie sumowane według poniższego wzoru:

$$I = k_a I_a + k_d I_d + k_s I_s \tag{5}$$

Gdzie:

- I – natężenie światła w punkcie
- I_a – (ang. *ambient*) natężenie światła otoczenia
- I_d – (ang. *diffuse*) natężenie światła rozproszonego
- I_s – (ang. *specular*) natężenie światła odbitego lustrzanie
- k – procentowy współczynnik wpływu składowych (przypisane do danego obiektu)

Prawo Lamberta mówi, że natężenie światła rozproszonego jest wprost proporcjonalne do kosinusa kąta pomiędzy wektorem promienia a wektorem normalnym do powierzchni. W skrócie - im mniejszy kąt, tym mniej światła jest rozproszone (powierzchnia skierowana prostopadle do promienia jest najjaśniejsza).

W modelu Phonga używany jest także dodatkowy efekt zway **tłumieniem atmosferycznym**, polegające na spadku natężenia światła wraz z rosnącą odległością od obserwatora. Zjawisko to można wykorzystać do generacji realistycznych warunków pogodowych, takich jak deszcz, zamieć, mgła.

Metoda śledzenia promieni mimo, że daje zdumiewające rezultaty to ma w sobie kilka poważnych wad:

- Jest **bardzo kosztowna obliczeniowo** przez dużą liczbę obliczeń potrzebnych do wyliczenia koloru każdego piksela. Najbardziej kosztowne jest liczenie kolizji, co optymalizować można np. algorytmem brył otaczających, czyli wpisaniem zaawansowanych brył w większe, prostsze kształty - jeśli nie wystąpi kolizja z prostym kształtem, to na pewno nie nastąpi także z tym zaawansowanym.
- Może występować **zjawisko aliasingu**, które sprawia, że niektóre małe obiekty są pomijane, a obiekty o ostrych krawędziach mogą być zniekształcone (poszarpane). W celu uniknięcia tego problemu stosowanych jest wiele metod **antyaliasingu** na przykład nadpróbkowanie (*supersampling - SSAA*), gdzie jeden promień pierwotny zastępowany jest **wiązką promieni** i kolor piksela jest zazwyczaj średnią z tej wiązki
- Przyrost ilości źródeł światła i obiektów znacznie pogorsza czas renderowania
- Nie wszystkie kierunki padania promieni są rozpatrywane
- Każdą klatkę trzeba generować od nowa - nie można wykorzystać informacji z poprzednich klatek

Wartym zaznaczenia jest fakt, że **śledzenie promieni dla każdego z pikseli odbywa się niezależnie** od innych pikseli, także metodę tę można **zrównoleglić** (zarówno dla grupy pikseli bądź promieni) na przykład wykorzystując zestawy instrukcji MMX oraz SSE z architektury SIMD równoległące te same operacje arytmetyczne na liczbach całkowitych w procesorze, bądź używając do tego celu dobrodziejstw najnowszych procesorów graficznych - na przykład technologii NVIDIA CUDA, która pozwala na otrzymanie obrazu generowanego tą metodą w czasie rzeczywistym.

8 K8 – Mechanizmy systemu operacyjnego wspomagające synchronizację procesów

System operacyjny jest oprogramowaniem, którego jedną z głównych ról jest synchronizacja procesów. Czym jest proces? Jest to wykonujący się program, czyli zbiór instrukcji dla procesora przechowywany w postaci pliku.

Każdy proces wymaga przydziału zasobów komputera. Czas procesora, pamięć, dostęp do urządzeń – w mniejszym lub większym stopniu, proces zawsze wymaga pewnych zasobów.

Procesy mogą wykonywać się zupełnie niezależnie od siebie – korzystając np. z innych urządzeń – lub współbieżnie obok siebie korzystając z np. tych samych plików.

Jak łatwo się domyślić, procesy współbieżne wymagają synchronizacji ze strony zarządcy zasobów – systemu operacyjnego.

Typowe problemy związane z procesami współbieżnymi to na przykład:

- problem sekcji krytycznej – kiedy procesy współdzielą strukturę danych, a operacje na nich muszą być atomowe, tj. nie mogą zostać przerwane w środku wykonywania się,
- problem czytelników i pisarzy – synchronizacja dostępu do zasobów dla procesów dokonujących i niedokonujących w nich zmian,
- problem producenta i konsumenta – przetwarzanie danych przez proces może się odbyć jedynie, po otrzymaniu ich od innego procesu,
- problem ucztujących filozofów – procesy korzystają ze wspólnych zasobów, które pobierają i zwalniają wg potrzeb.

Sekcja krytyczna to sekwencja operacji wykonywanych na zasobie (np. pamięci, pliku), które muszą zostać wykonane w trybie wyłącznym przez jeden proces, tj. podczas ich wykonywania, inny proces nie może działać na zasobach przez niego zajętych. Aby poprawnie rozwiązać problem sekcji krytycznej, należy zaimplementować algorytm, który będzie spełniał kilka warunków:

- instrukcje w sekcji krytycznej nie mogą być przeplatane – może w niej być wyłącznie jeden proces,
- nie można zakładać w jakiej kolejności i z jaką szybkością wykonają się dane procesy,
- proces nie może zatrzymać się w sekcji krytycznej,
- nie mogą występować zakleszczenia (jeden proces czeka na zwolnienie zasobów przez drugi, a drugi czeka na zwolnienie zasobów pierwszego, więc obydwa się zatrzymują),
- każdy proces musi wejść do sekcji krytycznej (nie może wystąpić zagłodzenie).

Istnieją przeróżne mechanizmy synchronizacji implementowane przez jądro.

Semafory są typem danych, które służą do kontroli dostępu zasobów przez wiele procesów. Semafory są zmienną całkowitą, która przyjmuje wartości nieujemne (tj. ≥ 0) lub dla semaforów

binarnych – wartości logiczne. Na semaforach można wykonywać dwie operacje – zmniejszenie (zajęcie, podniesienie,) oraz zwiększenie (zwolnienie, opuszczenie). Synchronizacja polega na blokowaniu procesu w operacji zajęcia semafora, gdy jego wartość po zajęciu jest ujemna, do czasu, aż wartość ta nie zostanie zwiększona do nieujemnej (np. przez inny proces, który zwolni semafor). Wyróżnia się rodzaje semaforów takie jak:

- binarne – zmienna przyjmuje tylko wartości *true* (semafor otwarty) lub *false* (semafor zamknięty),
- zliczające – zmienna przyjmuje wartości całkowite nieujemne, a aktualna wartość jest zwiększana/zmniejszana o 1 w wyniku zwolnienia/zajęcia semafora,
- uogólnione – semafor zliczający, ale może zwiększać/zmniejszać wartość o dowolną liczbę (oczywiście, wartość wciąż musi być nieujemna).

Muteksy (`mutual exclusion` – wzajemne wykluczanie) są szczególnym przypadkiem semaforów. Muteks obsługują operacje blokowania i zwalniania, jednak mogą być zwolnione tylko przez proces, które je zajął.

Spinlocki – podobne do semaforów, jednak oczekiwanie na zwolnienie blokady odbywa się na zasadzie aktywnego czekania, przez co zajmują czas procesora, może wystąpić zagłodzenie lub czekać w nieskończoność.

Monitory są strukturalnym narzędziem synchronizacji wątków. Składają się ze zmiennych oraz procedur operujących na nich, zebrane w jeden moduł. Dostęp do zmiennych jest możliwy wyłącznie za pomocą procedur monitora, a w danej chwili tylko jeden proces może wywoływać procedury monitora. Gdy inny proces wywoła procedurę monitora, to będzie on zablokowany do chwili opuszczenia monitora przez pierwszy proces. Istnieje możliwość wstrzymania i wznowienia procedur monitora za pomocą zmiennych warunkowych. Na zmiennych warunkowych można wykonywać operacje `wait` (wstrzymanie procesu i umieszczenie go na końcu kolejki) oraz `signal` (odblokowanie jednego z oczekujących procesów). Procesy oczekujące na wejście do monitora zorganizowane są w kolejkę FIFO.

9 K9 – Programowalne scalone układy cyfrowe PLD, CPLD oraz FPGA

Programowalne układy scalone to układy scalone, których funkcjonalność jest definiowana przez użytkownika końcowego, a nie producenta. Wszystkie wyprodukowane przez producenta układy są identyczne (co pozwala zmniejszyć koszty). Pozwala to na zaprojektowanie, uruchomienie i testowanie urządzenia. Programowanie odbywa się za pomocą tworzenia połączeń w istniejącej sieci ścieżek sygnałowych.

Układy programowalne można podzielić ze względu na ich strukturę:

- PLD – (programmable logic device) najprostsze,
- CPLD – complex PLD,
- FPGA – field-programmable gate array.

Układy PLD mogą realizować funkcje logiczne (tworząc układy kombinacyjne bądź sekwencyjne). Programowanie takich układów bazuje na ustawieniu odpowiednich bitów, by zostały zrealizowane dane funkcje. Każde urządzenie posiada dany zbiór wejść oraz wyjść do układu.

Najprostsze układy PLD (SPLD) pozwalają na tworzenie prostych układów. Do nich możemy zaliczyć układy PLE, PAL, PLA. Ich budowa opiera się na matrycach funkcji AND oraz OR – realizują funkcje postaci $(x_1 \wedge x_2 \wedge ...) \vee (x_m \wedge x_{m+1} \wedge ...) \vee ...$ lub $(x_1 \vee x_2 \vee ...) \wedge (x_m \vee x_{m+1} \vee ...) \wedge ...$ (dla sklerotyków: $\wedge$ – AND, $\vee$ – OR).

Układy PLA posiadają programowalne matryce AND oraz OR. Dowolna linia iloczynu logicznego z matrycy AND (tzw. *term*) może zostać podłączony do realizacji dowolnej operacji sumy logicznej OR. Inna architektura, PAL posiada programowalną matrycę AND oraz stałą (nieprogramowalną) matrycę OR. Natomiast układy PLE posiadają stałą matrycę bramek AND oraz programowalną matrycę bramek OR.

Ze względu na ograniczone możliwości logiczne – małą liczbę bramek oraz małą liczbę wejść/wyjść, układy SPLD mogły zastępować klasyczne obwody logiczne (czyli coś typu jak kleciliśmy na LUC-u).

Układy CPLD, jak nazwa wskazuje są bardziej złożone. Koncepcyjnie są podobne do SPLD, ale mają większe możliwości logiczne i funkcjonalne. Zbudowane są z zespołu struktur PAL połączonych ze sobą programowalną matrycą (switching matrix). Blok funkcjonalny CPLD posiada matrycę AND, makrokomórki oraz zadaną liczbę wyjść. Makrokomórki składają się zazwyczaj z bramek (np. AND, OR) oraz przerzutników. Sygnały sterujące makrokomórkami można podzielić na globalne (wspólne dla wszystkich makrokomórek), lokalne (wspólne dla zespołu połączonych ze sobą makrokomórek) oraz indywidualne (wpływają na działanie jednej makrokomórki). Pojedyncza makrokomórka realizuje prostą funkcję logiczną, większa ich liczba może być połączona ze sobą w bloki funkcyjne, tworząc funkcje z większą liczbą zmiennych.

Najbardziej skomplikowane są układy FPGA. Mają one jeszcze większe możliwości logiczne w stosunku do CPLD oraz są szybsze w działaniu. Zbudowane są one z konfigurowalnych elementów logicznych (CLB). W skład elementu logicznego wchodzą zazwyczaj generator funkcji logicznych,

przerzutnik i programowalne multipleksery. Generator funkcji logicznych określany jest jako LUT (look up table). Elementy logiczne są łączone w bloki. Połączenia między CLB są programowalne, podobnie jak w układach CPLD za pomocą programowalnej matrycy. LUT-y mają inne podejście do generowania funkcji – zamiast programowania połączeń między bramkami, LUT-y działają bardziej jak tabele prawdy – dla zadanego wejścia generują odpowiednie wyjście — **to jest znacząca różnica między FPGA a CPLD**.

Do zalet struktur FPGA można zaliczyć elastyczność architektury (równoległość przetwarzania, dowolną szerokość ścieżki danych), wielokrotne użycie tych samych zasobów sprzętowych, czy też rekonfigurowalność.

Do programowania programowalnych układów scalonych wykorzystywane są języki opisu sprzętu, takie jak VHDL lub Verilog. Opis układu jest tworzony w sposób behawioralny. Narzędzie syntezy przetwarza taki zapis na konkretną realizację sprzętową. W przeciwieństwie do mikrokontrolerów, układy potrafią przetwarzać równolegle, a μC przetwarzają sekwencyjnie. Nie są one ograniczone zbiorem funkcji – μC posiadają określony zestaw instrukcji. Wadą układów programowalnych w stosunku do nieprogramowalnych jest większy pobór mocy. Na układach programowalnych FPGA można zaprogramować procesor programowy – PicoBlaze, MicroBlaze.

Układy PLD mają szerokie spektrum zastosowań. Można do tego zaliczyć logikę scalającą – interfejs dla mikrokontrolerów umożliwiający współpracę z innymi modułami (pamięć, układy peryferyjne). Poprzez możliwość przetwarzania równoległego, układy PLD nadają się jako akceleratory sprzętowe. Akceleratory są wykorzystywane przy przetwarzaniu grafiki, dźwięku lub wideo.

10 K10 – Optyczne nośniki informacji

Dyski optyczne zaliczają się do nośników informacji. Zostały wprowadzone jako alternatywa dla dysków magnetycznych. Odczyt danych z dysku optycznego odbywa się za pomocą wiązki światła (a konkretniej promienia lasera) – nazwa opiera się na zasadzie działania takiego nośnika.

W dzisiejszych czasach nośniki optyczne mają różne pojemności, stosowane są w różnych dziedzinach (np. na nich dystrybuowana jest muzyka, czy też filmy).

Do dysków optycznych zaliczamy:

LaserDisc – analogowy dysk optyczny, który miał 30 cm średnicy, składał się z dwóch jednostronnych aluminiowych dysków pokrytych plastikiem. Na nośniku był zapisywany analogowy sygnał wideo oraz analogowy i/lub cyfrowy sygnał dźwiękowy. Posiadał formaty kodowania:

- CAV – stała prędkość kątowa, dyski pracowały ze stałą prędkością kątową 1800rpm, z odczytem jednej klatki na obrót, pojemność wynosiła 30 min na stronę, CAV posiadała funkcje stop klatki, zmiany prędkości odtwarzania oraz cofanie,
- CLV – stała prędkość liniowa, nie posiada funkcji z CAV, przez zmniejszenie prędkości obrotowej zwiększyła się pojemność do 60 min obrazu lub dźwięku,
- CAA (wariant CLV) – stałe przyspieszenie kątowe, wynalezione by wyeliminować zjawisko wchodzenia jednego kanału na drugi z CLV, różnica między CAA a CLV polega na dopasowaniu kąta obrotu dysku, zamiast stopniowego liniowego zwalniania, dzięki CAA znacznie poprawiła się jakość obrazu na nośniku,

CompactDisk – początkowo był stworzony tylko do przechowywania muzyki, później dodano możliwość przechowywania danych. Płyta CD ma średnicę 12 cm i może pomieścić do 80 min dźwięku lub 700 MB danych. Istnieje również format mini-CD, jednak jest on mniej popularny (do 24 min dźwięku, 210 MB pojemności; inny format – „wizytówka" CD – do 6 min dźwięku, do 65 MB pojemności). Formaty CD:

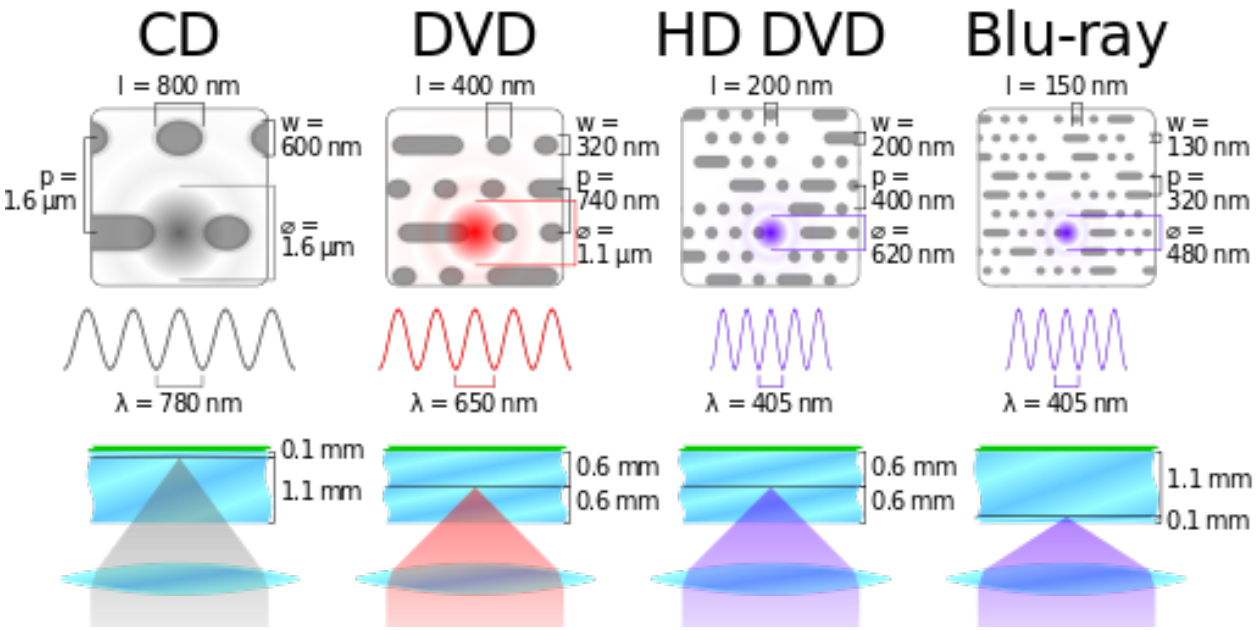
- Audio-CD – format to dwukanałowy 16-bitowy PCM, z próbkowaniem 44.1 kHz na kanał, dźwięk mono nie występuje w implementacji, zazwyczaj dźwięk mono jest przedstawiony jako dwa takie same kanały w stereo. CD-Text jest rozszerzeniem Audio-CD, zezwala na zachowanie informacji tekstowych (np. nazwa albumu, artysty, ścieżek).
- Video-CD – cyfrowy standard do przechowywania wideo na płytach CD. W porównaniu z kasetami VHS jakość wideo nie pogarsza się po każdym użyciu. Wykorzystywane rozdzielczości to 352x240 oraz 352x288.
- CD-R – płyta może zostać raz zapisana, posiada warstwę barwnika, który jest stapiany przy zapisywaniu,
- CD-RW – płyta do zapisu wielokrotnego, kasowanie danych odbywa się za pomocą lasera (następuje topnienie stopu, traci swoją strukturę polikrystaliczną).

DVD – medium do przechowywania różnego typu danych, zazwyczaj do oprogramowania oraz filmów. Nośniki posiadają takie same wymiary jak płyty CD, jednak można na nich zapisać

więcej danych. Zwiększenie danych jest możliwe dzięki zwiększeniu gęstości zapisu danych (użyto lasera o krótszej długości fali). Powstały również płyty dwustronne oraz dwuwarstwowe. Podobnie jak w przypadku płyt CD, istnieją tutaj płyty jednokrotnego oraz wielokrotnego zapisu. Płyty DVD również posiadają formaty Audio oraz Video. Płyty audio posiadają dużą możliwość konfiguracji kanałów, z różnym próbkowaniem (nawet do 24-bit, 192 kHz). W porównaniu do CD-Audio pozwala zapisać więcej dźwięku w lepszej jakości. Zapis wideo na płytach DVD-Video wykorzystuje kodek MPEG-2. Najczęściej stosowane rozdzielczości to 720x576 oraz 720x480. Następcą DVD jest HD DVD, który zawierał gęstszy zapis danych, jednakże został porzucony na rzecz płyt Blu-Ray (wykorzystują tę samą długość fali lasera).

Dostęp do drugiej warstwy możliwy jest przez przepuszczenie lasera przez pierwszą, częściowo przepuszczalną warstwę. Zmiana ścieżki może potrwać kilka sekund, przez co może spowodować chwilowe zatrzymanie pracy napędu.

Blu-ray jest kolejnym formatem dysków optycznych. Dyski posiadają 25 GB na warstwę, przy zapisie dwu-warstwowym daje to 50 GB. W 2008 roku przedstawiono szesnasto-warstwową płytę BD, o pojemności 400 GB. Istnieją również formaty trzy- i cztero-warstwowe płyt BD. Nazwa technologii wywodzi się od koloru wykorzystywanego lasera.

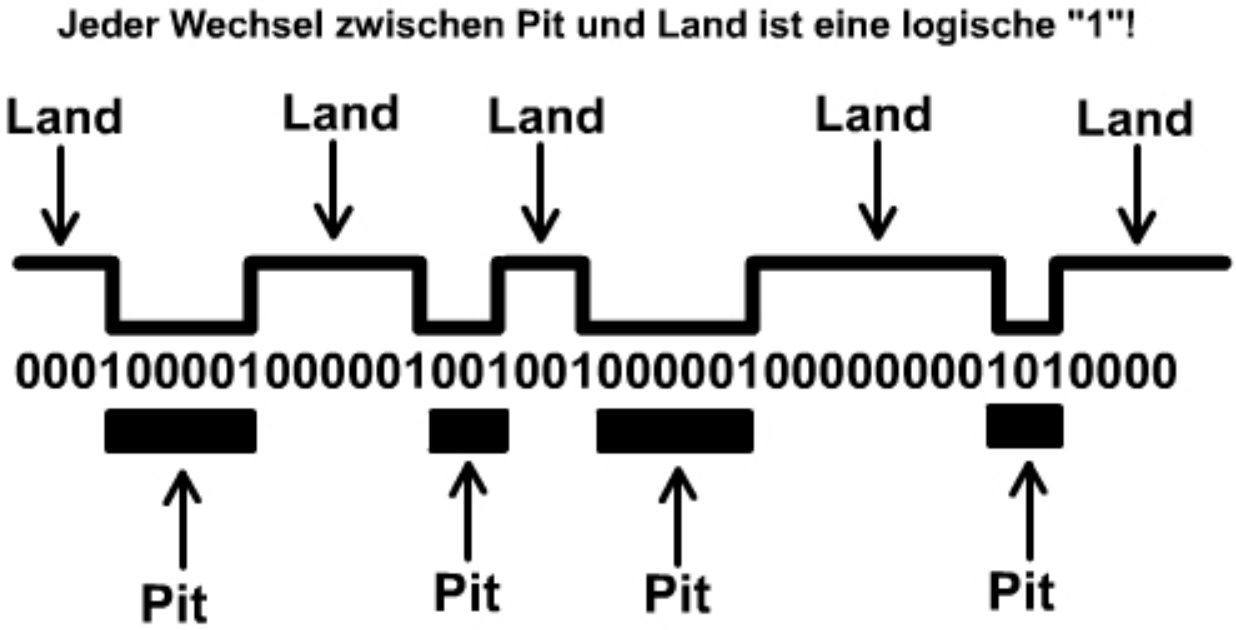


Rysunek 4: Porównanie formatów płyt. Źródło: Wikipedia

HVD (Holographic Versatile Disc) – technologia dysków optycznej nowej generacji, teoretycznie pozwala pomieścić do 6 TB na jednej warstwie. Zapis danych odbywa się w trójwymiarowej przestrzeni dysku. Wykorzystywane są dwa lasery – zielony oraz czerwony.

NRZI – metoda odwzorowania sygnału binarnego, bez powrotów do zera. Wykorzystuje dwa stany logiczne. Przejście między nimi następuje, gdy transmitowany bit wynosi 1, w przypadku bitu 0 zmiana nie następuje.

Stany logiczne na płycie CD są reprezentowane przez pity (wgłębienia) oraz landy (płaskie powierzchnie, przerwy między wgłębieniami). Gdy wiązka trafia na pit, to jest rozpraszana i nie wraca do czytnika. Przy landzie wiązka jest odbijana i trafia z powrotem do czujnika. Każda przejście z landu na pit o odwrotnie powoduje zmianę stanu (logiczne 0 lub 1). Dane zapisywane są od środka na zewnątrz płyty.



Rysunek 5: Stany logiczne odczytywane z powierzchni płyty

Kodowanie wykorzystywane w płytach to EFM (eight to fourteen modulation). Jest wykorzystywane ze względu na ograniczenia technologiczne (szybkości światła i odpowiedzi modulatora). Przez to długości pitów i landów muszą znajdować się w przedziale od 3 do 11 bitów kanałowych (ciąg kilku pitów/landów nie może być krótszy niż 3 bity kanałowe oraz dłuższy niż 11). Dlatego każde 8 bitów jest zapisywane za pomocą 14 bitów kanałowych. Do konwersji jest wykorzystywana tabela, która jest zapisana w firmware napędu.

Dla płyt DVD, ze względu na mniejsze odległości stosowany jest zapis bajtów na 16 bitowych sekwencjach. Długości pitów i landów muszą być w przedziale od 3 do 14 bitów kanałowych.

Płyty Blu-ray wykorzystują kodowanie podobne do EFM, 17PP (one seven parity preserving). Wykorzystuje kodowanie, w którym każde 2 bity są zapisywane na 3 bitach, a samo kodowanie odbywa się w grupach po 2 albo 4 bity. Dodatkowo występują bity kontrolne (DC), po których wystąpieniu (jeśli DC == 1), zmieniana jest polaryzacja strumienia NRZI.

A teraz z innej beczki... Innym optycznym nośnikiem informacji są **kody kreskowe**. Skanery odczytują zapisane dane. Mogą być wydrukowane na różnych powierzchniach. Składają się z jasnych oraz ciemnych elementów. Istnieje wiele rodzajów kodów kreskowych, można je pogrupować według zastosowania, np. identyfikacja towarów (EAN, Code 11), kody pocztowe (POSTNET, PLANET, etc.). Istnieją również dwuwymiarowe kody kreskowe – składają się one z macierzy ele-

mentów (jasnych/ciemnych). Do nich można zaliczyć kody takie jak: kody QR, czy Aztec Code (wykorzystywany w polskich dowodach rejestracyjnych).

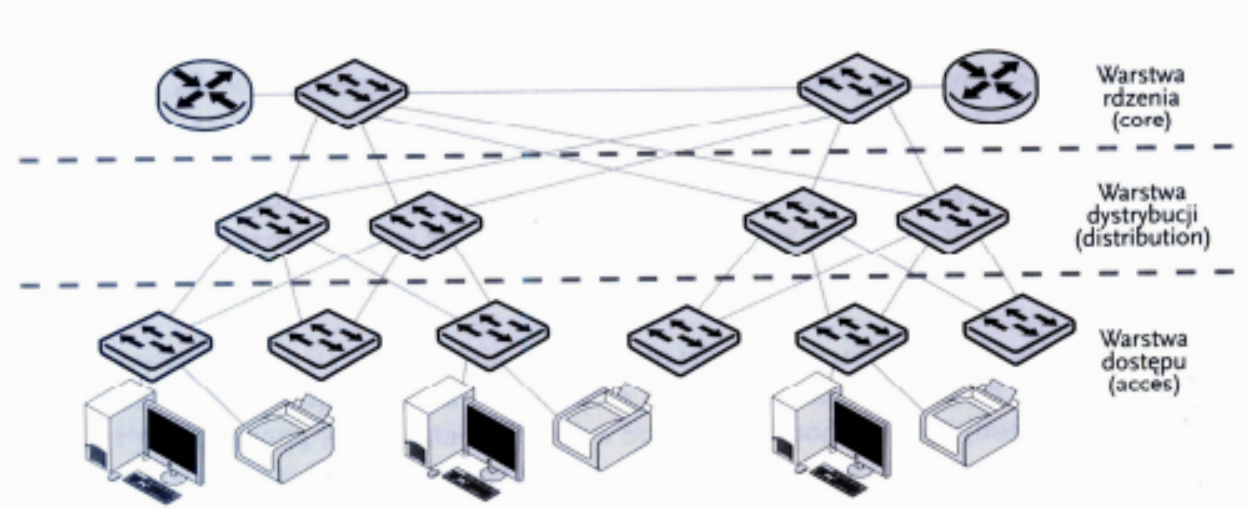
Ten temat to temat rzeka, można opowiedzieć o wielu rzeczach (IrDA, światłowody i pewnie wiele innych). Ostatnim rodzajem, o którym chcę wspomnieć to dyski magnetooptyczne. Są stworzone z tworzywa sztucznego oraz warstwy materiału magnetycznego, zabezpieczone w kasecie, chroniąc przed uszkodzeniami mechanicznymi. Podobnie jak w wypadku płyt omówionych wcześniej, istnieją dyski magnetooptyczne zapisywane jedno- oraz wielokrotnie. Odczyt danych z takiego dysku wykorzystywane jest zjawisko Kerra (zmiana współczynnika załamania światła przez pole magnetyczne). Przy odbiciu lasera od namagnesowanego miejsca następuje „skręcenie" w kierunku zależnym od pola magnetycznego i ta zmiana jest wykrywana przez głowicę napędu i traktowana jako stan logiczny.

11 S1 – Zasady projektowania sieci komputerowych

Podczas projektowania architektury komutowanej sieci LAN (switched LAN) używa się modelu hierarchicznego sieci. Sieci w modelu hierarchicznym dzieli się na odrębne warstwy. Każda z nich realizuje określone funkcje, które definiują rolę danej warstwy w ogólnym modelu sieci. Budowa sieci przyjmuje postać modułową, co zwiększa jej skalowalność i efektywność działania.

W modelu hierarchicznym można wyróżnić trzy warstwy:

- warstwa dostępu (access layer),
- warstwa dystrybucji (distribution layer),
- warstwa rdzenia (córę layer).



Rysunek 6: Model hierarchiczny budowy sieci przełączanej

Warstwa dostępu jest sprzężona z takimi urządzeniami końcowymi, jak komputery PC, drukarki, telefony IP, a jej celem jest zapewnienie dostępu do pozostałych składników sieci. Jej głównym zadaniem jest:

- umożliwienie połączenia urządzeń z siecią,
- umożliwienie kontroli nad komunikowaniem się urządzeń w sieci.

W warstwie dostępu mogą występować przełączniki, mosty, koncentratory i bezprzewodowe punkty dostępowe.

Warstwa dystrybucji gromadzi dane otrzymywane z przełączników z warstwy dostępu przed ich transmisją do warstwy rdzenia. Warstwa ta kontroluje przepływ danych w sieci oraz

wyznacza domeny rozgłoszeniowe. Może również realizować routing między wirtualnymi sieciami LAN (VLAN - Virtual LAN), jeżeli na poziomie warstwy dostępu utworzono takie sieci.

Warstwę rdzenia stanowią szybkie łącza szkieletowe. W warstwie tej gromadzi się ruch sieciowy ze wszystkich urządzeń warstwy dystrybucji, a zatem musi być ona w stanie szybko przekazywać duże ilości danych. Warstwa rdzenia może być połączona z zasobami inernetowymi.

Aby zapewnione zostały maksymalne korzyści przy minimalnym nakładzie pracy i środków, sieć komputerowa powinna posiadać następujące cechy:

- skalowalność,
- nadmiarowość,
- wydajność,
- bezpieczeństwo,
- łatwość zarządzania i utrzymania.

Skalowalność możemy rozumieć jako podatność sieci na rozbudowę. Rozrastanie się sieci o dużej skalowalności można łatwo zaplanować i realizować. Na przykład, przyjmijmy założenie, że na przełącznik z warstwy dystrybucji może przypadać dziesięć przełączników z warstwy dostępu. Wtedy dodatkowy przełącznik w warstwie dystrybucji trzeba będzie dodać do topologii sieci dopiero po przekroczeniu maksymalnej liczby dziesięciu podłączonych przełączników warstwy dostępu.

Zwiększenie niezawodności sieci można osiągnąć, wprowadzając **nadmiarowość** (redundancję) urządzeń lub/i ścieżek. W celu zapewnienia nadmiarowości poszczególne przełączniki z warstwy dostępu są łączone z więcej niż jednym przełącznikiem z warstwy dystrybucji (np. z dwoma różnymi przełącznikami z warstwy dystrybucji). Jeśli jeden z przełączników warstwy dystrybucji ulegnie awarii, to przełącznik z warstwy dostępu może współpracować z drugim przełącznikiem z tej warstwy. Z kolei przełączniki z warstwy dystrybucji są łączone z co najmniej dwoma przełącznikami z warstwy rdzenia. W warstwie dostępu nie występuje nadmiarowość - urządzenia końcowe (komputery, drukarki itp.) nie mogą być przyłączone do więcej niż jednego przełącznika. Jeśli w warstwie dostępu wystąpi awaria przełącznika, to będzie mieć wpływ tylko na urządzenia, które są do niego podłączone.

Wydajność komunikacji można poprawić, unikając transmisji danych przez niskowydajne przełączniki pośredniczące. W warstwie dystrybucji powinny być stosowane przełączniki o wydajności większej niż w warstwie dostępu. Przełączniki w warstwie rdzenia powinny mieć najwyższą wydajność, aby zapewnić szybkie przesyłanie dużej ilości danych. Zastosowanie w warstwie dostępu tańszych przełączników i zwiększenie nakładów na przełączniki z warstw dystrybucji i rdzenia pozwala uzyskać jednocześnie wysoką wydajność sieci i oszczędności finansowe.

Profesjonalne przełączniki umożliwiają **zwiększenie bezpieczeństwa** poprzez wprowadzenie zasad ograniczających dostęp do sieci. Przełączniki z warstwy dostępu można kon-figurować, stosując różne opcje zabezpieczeń portów (port security) oraz sieci wirtualne VLAN, zapewniające kontrolę nad tym, które urządzenia mogą się łączyć z siecią.

W trakcie rozrastania się sieci jej utrzymanie staje się coraz bardziej skomplikowane. Urządzenia stosowane w danej warstwie powinny posiadać podobne parametry techniczne i konfigurację.

Dla zapewnienia **łatwości zarządzania i utrzymania** można stosować jednakowe urządzenia. Jeśli trzeba będzie zmienić funkcjonalność jakiegoś przełącznika, np. z warstwy dostępu, to zmianę tę można powielić we wszystkich przełącznikach z tej warstwy, gdyż najprawdopodobniej wykonują one te same funkcje. Wdrażanie nowych przełączników jest ułatwione, ponieważ ich konfiguracje można skopiować z innych urządzeń i ewentualnie wprowadzić niewielkie modyfikacje.

Przykładowe Etapy projektowania sieci:

1. Inwentaryzacja,
2. Poznanie i analiza potrzeb użytkownika,
3. Wynikające z analizy założenia projektowe,
4. Projekt sieci:
   (a) Projekt logiczny sieci,
   (b) Projekt okablowania (poziome – użytkownik punkt dystrybucyjny, pionowe – pomiędzy punktami dystrybucyjnymi),
   (c) Konfiguracja adresacji,
   (d) Podłączenie do Internetu,
   (e) Bezpieczeństwo i niezawodność sieci,
   (f) Kosztorys

12 S2 – Protokoły rozległych sieci komputerowych.

Sieć WAN (z ang. *Wide Area Network*) – sieć komputerowa znajdująca się na obszarze wykraczającym poza miasto, kraj, kontynent.

- Łączą ze sobą urządzenia rozmieszczone na dużych obszarach geograficznych (np. kraju, kontynentu).
- W celu zestawienia łącza lub połączenia między dwoma miejscami korzystają z usług operatorów telekomunikacyjnych, np. Netia, TP S.A., NASK, Exatel.
- Wykorzystują różne odmiany transmisji szeregowej.

Sieć WAN działa w warstwie fizycznej oraz warstwie łącza danych modelu odniesienia OSI. Łączy ona ze sobą sieci lokalne, które są zazwyczaj rozproszone na dużych obszarach geograficznych. Sieci WAN umożliwiają wymianę ramek i pakietów danych pomiędzy routerami i przełącznikami oraz obsługiwanymi sieciami LAN.

Protokoły sieci rozległych można je podzielić na następujące grupy technik łączenia:

- **komutacja kanałów** - polega na przydzieleniu wybranemu połączeniu wybranej sekwencji połączonych kanałów od terminala źródłowego do terminala docelowego. W sieciach z komutacją kanałów przesyłanie danych następuje dopiero po ustanowieniu połączenia, czyli uzyskaniu specjalnej trasy pomiędzy systemem nadawcy a systemem odbiorcy. Trasa jest sekwencją kolejno połączonych kanałów. Kanały te zostają zajęte przez cały czas, w którym trwa połączenie. Zarezerwowane kanały nie mogą być używane przez inne połączenia. Samo przesyłanie informacji odbywa się w 3 fazach: ustanowienie połączenia, transfer danych, rozłączenie połączenia.

  Jedną z wad techniki komutacji kanałów jest konieczność ustanowienia połączenia między użytkownikami zanim zostaną przesłane dane. Powoduje to powstawanie opóźnień w przesyłaniu informacji oraz to, że kanały są niewykorzystywane podczas zestawiana połączenia, rozłączania połączenia oraz w przerwach między transmisją przy zestawionym połączeniu. W efekcie wzrasta koszt utrzymania takiej sieci a spada jej efektywność. Na plus można zaliczyć wysoką jakość transmisji poprzez trwały kanał (tzn. ustanowiony wcześniej kanał o niezmiennych parametrach). Technika ta wykorzystywana jest w sieciach gdzie przesyła się głos i dane.

- **łącza dzierżawione** - łącze telekomunikacyjne (miedziane lub światłowód) wydzierżawione od innego operatora lub osoby indywidualnej po to, aby móc przesłać po nim swój sygnał. Łącze dzierżawione transmisji danych ma stałą przepustowość.

  Opłaty za dzierżawę łącza uiszczane są niezależnie od stopnia jego wykorzystania. Na przykład niezależnie od ilości przesyłanych danych i czasu transmisji w ustalonym okresie, którego dotyczy opłata.

  Łącza tego typu wykorzystywane są w sytuacjach wymagających zagwarantowanej dyspozycyjności, na przykład dostępu do internetu o gwarantowanej przepustowości. Umożliwiają one klientom uruchamianie własnych usług internetowych dostępnych w trybie całodobowym.

- **komutacja komórek** - jest odmianą komutacji pakietów, w której pakiety zastąpiono krótkimi komórkami o stałej długości, co pozwala na sprzętową realizacje komutacji. Z komutacją łączy wiąże komutacje ATM konieczność zestawiania połączenia między stacjami końcowymi, przed rozpoczęciem przesyłania informacji użytkowej, tzn. zapewnienia (w sensie statycznym) dostępności zasobów sieci (łączy, buforów w węzłach) w czasie przesyłania tej informacji. W komutacji komórek ATM są aktualizowane wartości wirtualnych ścieżek i kanałów (VPI/VCI).
- **komutacja pakietów** - technika komutacji pakietów należy do najbardziej elastycznych technik komutacji stosowanych we współczesnych sieciach. Polega ona na przesyłaniu danych przez sieć w postaci pakietów. W odróżnieniu od techniki komutacji łączy pozwala użytkownikom nawiązywać połączenia z wieloma innymi użytkownikami jednocześnie.

  Pakiety powstają w wyniku podzielenia informacji użytkownika na części o stałej długości (wyjątkiem jest ostatnia część, której długość może być mniejsza) i opatrzenie tych „wycinków" w nagłówek N, też o stałej długości. Nagłówek zawiera informacje, które umożliwiają pakietowi dojście z punktu źródłowego do docelowego, węzłom sieci sprawdzenie poprawności zawartych w pakiecie danych, a punktom docelowym właściwie zestawić i odtworzyć podzieloną informację. W nagłówku zatem, w zależności od organizacji sieci, znajdują się następujące informacje: adresy źródłowy i docelowy, numer portu źródłowego i docelowego, numer pakietu, wskaźnik ostatniego pakietu, a także identyfikator zawartej w pakiecie informacji.

  Efektem tego jest kilka cech komutacji pakietów:

  - odporność na uszkodzenia sieci (uszkodzone urządzenia są po prostu omijane)
  - możliwość docierania pakietów w przypadkowej kolejności (ze względu na różne ścieżki transmisji)
  - opóźnienia związane z buforowaniem pakietów w routerach
  - duża przepustowość efektywna sieci
  - Alternatywą dla komutacji pakietów jest komutacja łączy.

  Przykłady sieci wykorzystujących komutację pakietów:

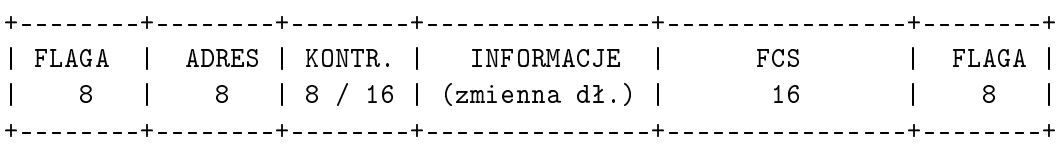
  - IPv4
  - IPv6

  Przykładowe protokoły:

- **komutacja kanałów:**
  - **ISDN** - technologia sieci telekomunikacyjnych mająca na celu wykorzystanie infrastruktury PSTN do bezpośredniego udostępnienia usług cyfrowych użytkownikom końcowym (bez pośrednictwa urządzeń analogowych) (ang. end-to-end circuit-switched digital services). Połączenia ISDN zalicza się do grupy połączeń wdzwanianych (komutowanych). ISDN jest znormalizowana w zaleceniach ITU-T oraz standardach ETSI. Europejskie zostały wprowadzone do systemu Polskich Norm jako grupa ICS 33.080 - Sieć Cyfrowa z Integracją Usług
  - **PPP** - protokół warstwy łącza danych, łączy ze sobą dwa węzły sieci tak, że widzą się one, jakby były bezpośrednio ze sobą połączone; protokół ten tworzy tunel, którym idą już właściwe dane w innym protokole (np. dalej IP i TCP).

- **komutacja pakietów**
  - **HDLC** bitowo zorientowany protokół warstwy łącza danych, używany do transmisji punktpunkt, jak i jeden-do-wielu, w sieciach z komutacją pakietów. Charakterystyczne jest to, że stacja może pracować w jednym z trzech trybów:
    * stacja *primary* – może inicjować transmisję ze stacją *secondary*, kontroluje ją, może utrzymywać wiele sesji ze stacjami podrzędnymi
    * stacja *secondary* – wykonuje polecenia stacji nadrzędnej, utrzymuje tylko jedną sesję
    * stacja uniwersalna – taki mix dwóch powyżej, może utrzymywać tylko jedną sesję

    Podczas transmisji przesyłane są ramki o stałej strukturze:

  - **X.25** Protokół połączeniowy, łączycy węzły sieci rozległych, z komutacją pakietów: DCE *Data Circuit-Terminating Equipment* – wewnętrzne węzły sieci rozległej, DTE *Data Terminal Equipment* – system komputerowy użytkownika. Protokół X.25 definiuje tzw. połączenie wirtualne – czyli mamy kilka połączeń w jednym fizycznym kanale, które postrzegane są tak, jakby naprawdę istniało kilka kabli. Wśród połączeń wirtualnych wyróżnia się:
    * PVC – czyli stałe połączenie wirtualne
    * SVC – tymczasowe połączenie wirtualne

    Protokół ten ma bardzo rozbudowane mechanizmy kontroli błędów (np. potwierdzanie każdej ramki), stąd nadal jest używany tam, gdzie wymaga się pewności transmisji kosztem prędkości (bankomaty, bankowość ogólnie, medycyna, monitoring waż- nych danych i zasobów). Aktualnie maksymalna prędkość uzyskiwana przy pomocy tego protokołu to 2 Mb/s.
  - **Frame Relay** następca X.25, posiada prostsze mechanizmy kontroli poprawności, przez co transmisja jest o wiele szybsza niż w X.25. Reszta ogólnie tak, jak w X.25.

    Maksymalna prędkość transmisji to 45 Mb/s.
- **komutacja komórek**
  - **ATM** protokół połączeń w sieciach rozległych, wykorzystuje komutacje komórek. Dane przesyłane są w małych małych porcjach: nagłówek (5 bajtów) + dane (48 bajtów). Też występują tu kanały wirtualne oraz ścieżki wirtualne (orientacja połączeniowa). Kanał fizyczny dzielony jest na ścieżki wirtualne, a te dopiero na kanały wirtualne. Mechanizmy kontroli i sterowania przepływem muszą być realizowane po stronach systemów użytkowników końcowych. Maksymalna prędkość transmisji to 622 Mb/s.

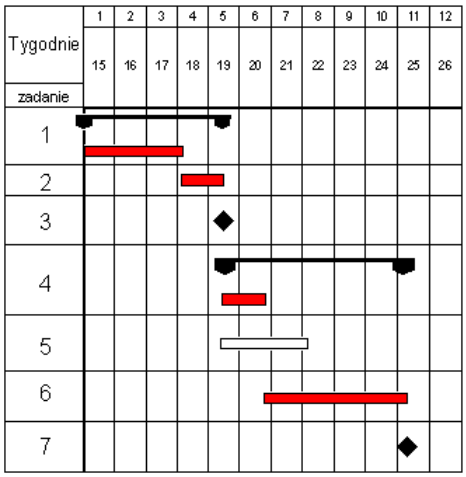
13 S3 – Metody tworzenia harmonogramów w projekcie informatycznym

Obecnie istnieje wiele metodyk pozwalających na efektywne zarządzanie projektem, nie można jednoznacznie wybrać najlepszej, każda ma charakterystyczne elementy. Projekt można podzielić na etapy, określić przewidywany czas realizacji, zasoby, zależności między etapami.

Stosowaną np. przez Nas na seminarium dyplomowym metodą do określenia stanu zaawansowania prac nad naszą pracą inżynierską był wykres Gantta.

Wykres Gantta jest graficznym sposobem planowania i kontroli. Planowanie i koordynowanie przebiegu różnych czynności w przekroju czasowym odgrywa istotną rolę w tworzeniu i funkcjonowaniu organizacji. Wykresy Gantta służą do planowania działań wielopodmiotowych zarówno zespołowych, jak i grupowych. Przedstawiają następstwo kolejnych zdarzeń, uwzględniając również zadania wykonywane równolegle. Dzięki tej technice można także kontrolować realizację zaplanowanego przedsięwzięcia.

Podstawowym celem jest wspomaganie pracy kierownika projektu dzięki podkreślaniu związków między zadaniami oraz wpływu potencjalnych zmian na cały projekt. Pozwala również przeprowadzać symulację, która umożliwia określenie proponowanych zmian na specyfikację, dostępność zasobów i ustalone terminy. Wykres Gantta pełni także zasadniczą rolę w czasie optymalizowania pierwszej „realistycznej" wersji planu projektu.

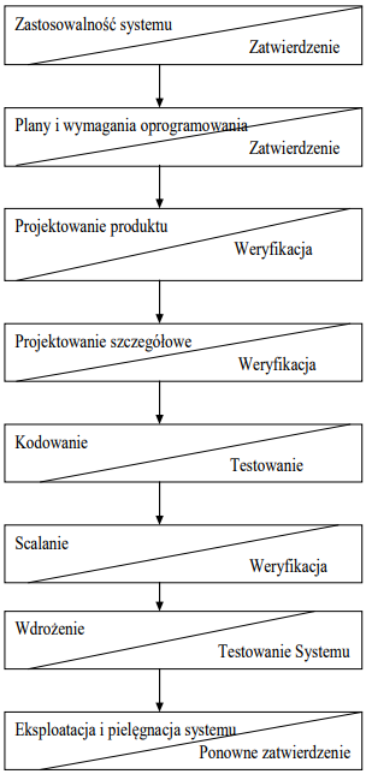


Rysunek 7: Przykładowy wykres Gantta z zadaniami krytycznymi, niekrytycznymi, kamieniami milowymi i podsuowaniami

Oprócz tego sposób organizacji czasu wyznaczają metodyki stosowane w projekcie informatycznym:

- **Model kaskadowy** - projekt jest dzielony na sześć do ośmiu faz, które następują kolejno po sobie. Danymi wejściowymi dla każdej fazy są dane wyjściowe fazy poprzedniej. Model

stosowany jest w projektach o dobrze zdefiniowanych wymaganiach, stosowany najczęściej w projektach małych.



Rysunek 8: Przykładowy proces kaskadowy

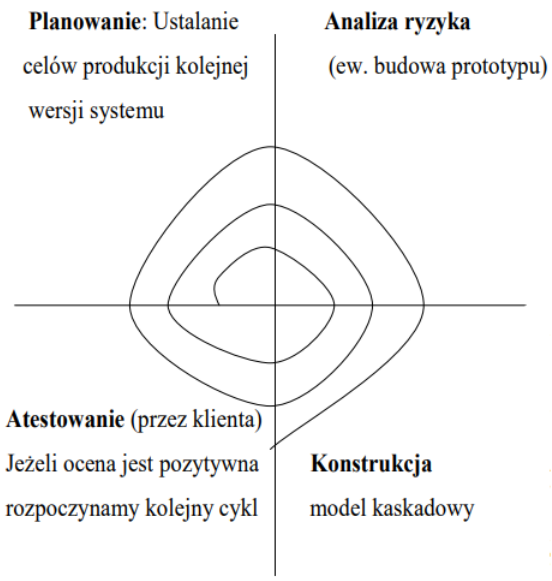
Zalety metody:

- Cały proces jest dobrze udokumentowany i ustalony z klientem.
- Ułatwia organizację: planowanie, harmonogramowanie i monitorowanie projektu
- Wszystkie wymagania są analizowane i zatwierdzane przez klienta, klient sam może oszacować korzyści z wdrożenia systemu
- Wymusza dokończenie dokumentacji po każdej fazie

Wady:

- Ścisła kolejność wykonywanych prac – realizatorzy poszczególnych faz nie mogą pracować równolegle
- Mała możliwość modyfikacji projektu
- Wysokie koszty błędów popełnionych we wczesnych fazach
- Testowanie systemu dopiero po zakończeniu programowania – może powodować poprawianie „na skróty”
- Niedopasowanie do rzeczywistości – rzeczywiste projekty rzadko są sekwencyjne

- **Model spiralny** - rozwinięcie metody kaskadowej – polecany w przypadku dużych, drogich i skomplikowanych projektów. Istota sprowadza się do tego, że jest to iteracyjna metoda kaskadowa.



Rysunek 9: Przykładowy proces spiralny

Zalety metody:

- Cały system powstaje we współpracy z klientem
- Wszystkie braki fazy ustalania wymagań są identyfikowane w miarę postępu prac

Wady:

- Wymagana jest dyscyplina po stronie klienta
- Kontrola wykonawcza jest trudna, bo klient nie odpowiada ani za budżet ani za harmonogram

- **SCRUM** jest metodyką adaptacyjną, która w szybkim tempie zyskuje popularność na całym świecie - również w Polsce. Jako jeden z powodów sukcesu SCRUM podaję się fakt, iż daje on konkretny i spójny przepis na prowadzenie projektów informatycznych oferując przy tym dużą swobodę jeśli chodzi o sposób implementacji w konkretnej organizacji

  SCRUM definiuje cztery główne role w projekcie:

  - Właściciel produktu – jest to klient lub sponsor projektu odpowiedzialny za kwestie biznesowe
  - ScrumMaster - jest to osoba odpowiedzialna za kontrolę przebiegu procesów
  - Członek zespołu - osoba należąca do zespołu realizującego projekt bez względu na wykonywane czynności. (programista, analityk etc.)
  - Interesariusze – osoby odpowiedzialne za podejmowanie decyzji strategicznych, których taktyczną implementację prowadzi właściciel produktu. Interesanci nie biorą czynnego udziału w codziennych spotkaniach zespołu projektowego – mogą jednak pełnić rolę obserwatorów.

  Zespół projektowy pracuje w określonym przedziale czasowym zwanym przebiegiem (ang. **sprint**). Efektem przebiegu za każdym razem powinno być dostarczenie użytkownikom kolejnego działającego produktu. Zasadą jest to, że zmiany wprowadzane w jednym przebiegu

muszą być namacalne dla użytkowników. Muszą wnosić wartość funkcjonalną. Przebieg może trwać **od 2 do 6 tygodni**. Zaleca się stosowanie przebiegów o stałych długościach.

W pierwszym etapie tworzona jest lista wymagań użytkownika, są one gromadzone w postaci **"historyjek"**. Każda historyjka opisuje jedną cechę systemu. Właściciel projektu jest też zobowiązany do przedstawienia priorytetów wymagań oraz głównego celu przebiegu. Po tym formułowany jest rejestr wymagań. Cel przebiegu jest **zapisywany w widocznym miejscu** w pokoju członków zespołu.

Następnie wybierane są zadania o najwyższym priorytecie, a jednocześnie przyczyniające się do realizacji celu projektu. Szacuje się czas realizacji każdego zadania. Lista zadań wraz z oszacowaną czasochłonnością nosi nazwę rejestru zadań przebiegu (**sprint backlog**).

14 S4 – Urządzenia sieci komputerowych

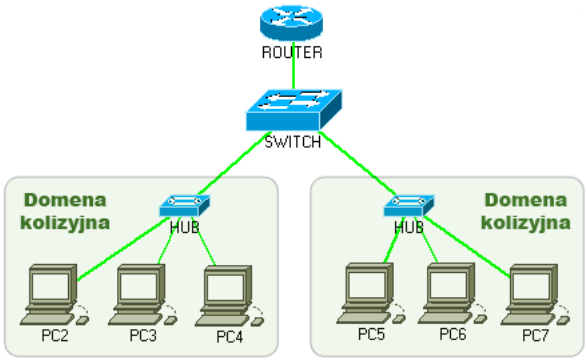
Sieci komputerowe zbudowano, aby wymieniać dane między komputerami. Wymianę tą zapewnia zastosowanie odpowiedniego sprzętu oraz oprogramowania. Podstawowymi urządzeniami stosowanymi do budowy sieci komputerowych są:

- **Modem** (ang. *Modulator DEModulator*) to urządzenie, które zamienia cyfrowe dane, generowane przez komputer, na sygnały analogowe i wysyła je za pomocą sieci. Podczas odbierania danych z sieci sygnały analogowe są zamieniane na cyfrowe i przekazywane do komputera. Modem może być wykorzystywany do połączenia komputera lub sieci LAN z Internetem za pośrednictwem stacjonarnej linii telefonicznej lub do przesyłania danych pomiędzy sieciami LAN. Zaletą modemu jest powszechna dostępność do usługi. Modemy są stosowane również w sieciach telewizji kablowej i telefonii komórkowej (np. modemy 3G/4G).
- **Karta sieciowa** (*Network Interface Card*) to urządzenie łączące komputer z lokalną siecią komputerową. Głównym zadaniem karty sieciowej jest przekształcanie ramek danych w sygnały, które są przesyłane w sieci komputerowej. Karta sieciowa w standardzie **Ethernet** (najczęściej spotykanym) ma unikatowy w skali światowej **adres fizyczny MAC** (ang. *MAC adress*), przyporządkowany jej podczas produkcji i zapisany w pamięci ROM. Karty mogą pracować z różnymi prędkościami. Obecnie standardem w przypadku sieci przewodowych są karty sieciowe pracujące z prędkością 100 Mb/s lub 1 Gb/s. W bezprzewodowych kartach sieciowych do przesyłania danych wykorzystywane są fale radiowe. Karta sieciowa może być wlutowana w płytę główną komputera lub innego urządzenia, albo montowana w różnych odmianach złączy PCI, PCMCIA, ExpressCard, USB, PCIExpress.
- **Wzmacniacz** (ang. *repeater*), zwany również regeneratorem, wykorzystuje się w miejscach, w których jest wymagane wzmocnienie lub regeneracja sygnału, niezbędne do zwiększenia zasięgu sieci. Rzadko jest to samodzielne urządzenie. Najczęściej funkcję wzmacniaka pełni urządzenie sieciowe posiadające własne zasilanie w energię elektryczną, np. koncentrator.
- **Koncentrator** (ang. *hub*) to urządzenie posiadające wiele portów służących do przyłączania stacji roboczych lub innych urządzeń. Koncentratory mogą być pasywne i aktywne. Pasywny pełni tylko funkcję skrzynki łączeniowej, rozsyłającej sygnał otrzymany na jednym porcie do wszystkich pozostałych. Aktywny dodatkowo wzmacnia sygnały. **Tworzy domenę kolizyjną**.
- **Most** (ang. *bridge*) to urządzenie posiadające dwa porty, służące do łączenia segmentów sieci. W swojej pamięci zapamiętuje adresy MAC urządzeń przyłączonych do poszczególnych portów. Po otrzymaniu ramki danych sprawdza adres miejsca docelowego i określa, do jakiego segmentu należy przesłać daną ramkę. Gdy komputer z jednego segmentu wysyła wiadomość, most analizuje zawarte w niej adresy MAC i na tej podstawie podejmuje decyzję, czy sygnał przesłać do drugiego segmentu, czy go zablokować. W sieci nie są wtedy przesyłane zbędne ramki, dzięki czemu zwiększa się jej wydajność.
- **Punkt dostępowy** (ang. *Access Point*) to urządzenie zapewniające stacjom bezprzewodowym dostęp do zasobów sieci za pomocą bezprzewodowego medium transmisyjnego. Pełni funkcję mostu łączącego sieć bezprzewodową z siecią przewodową. Do sieci bezprzewodowych są przyłączane laptopy, palmtopy, smartfony oraz komputery stacjonarne wyposażone w karty bezprzewodowe. Punkt dostępowy może być połączony w jedno urządzenie z routerem.

- **Przełącznik** (ang. *switch*) oferuje te same funkcje, co koncentrator, a dodatkowo pozwala, podobnie jak most, podzielić sieć na segmenty. Urządzenie posiada wiele portów przyłączeniowych, pozwalających na podłączenie komputerów, innych przełączników lub koncentratorów. Porty w przełączniku mogą pracować z jednakowymi prędkościami (przełączniki symetryczne) lub z różnymi prędkościami (przełączniki asymetryczne). Przełączniki mogą być wyposażone w funkcje zarządzania i monitoringu sieci. W odróżnieniu do huba **nie tworzy domeny kolizyjnej**.
- **Router** to urządzenie stosowane do łączenia sieci, np. do przyłączania sieci LAN do Internetu. Jest urządzeniem konfigurowalnym, pozwala sterować przepustowością sieci i podnosi jej bezpieczeństwo.
- **Brama sieciowa** (ang. *gateway*) to urządzenie, za pośrednictwem którego komputery z sieci lokalnej komunikują się z komputerami w innych sieciach. W sieci TCP/IP domyślna brama oznacza router, do którego komputery sieci lokalnej mają wysyłać pakiety adresowane do innej sieci, np. internet. Niektóre bramy umożliwiają komunikację między sieciami, w których działają różne protokoły.
- **Bramka VolP** (ang. *Voice over Internet Protocol*) to urządzenie, którego zadaniem jest umożliwienie wykonywania połączeń telefonicznych tradycyjnym aparatem telefonicznym za pośrednictwem sieci komputerowej wykorzystującej protokół IP. Bramka VoIP zamienia analogowy sygnał mowy oraz sygnały wybierania numeru telefonicznego na sygnały VoIP.
- **Zapora sieciowa** (ang. *firewall*) to dedykowany sprzęt komputerowy wraz ze specjalnym oprogramowaniem, blokujący niepowołany dostęp do sieci. Jego zadaniem jest filtrowanie połączeń wchodzących (ochrona przed nieuprawnionym dostępem z zewnątrz) i wychodzących do sieci (ochrona przed nieuprawnionym wypływem danych z sieci lokalnej na zewnątrz). Rolę zapory może pełnić również komputer wyposażony w system operacyjny, np. Linux z odpowiednim oprogramowaniem.

Urządzenia sieciowe mogą być ze sobą łączone, np. router z przełącznikiem i punktem dostępowym, zintegrowana brama sieciowa zawierająca router, przełącznik, firewall, bramkę VoIP i punkt dostępowy.

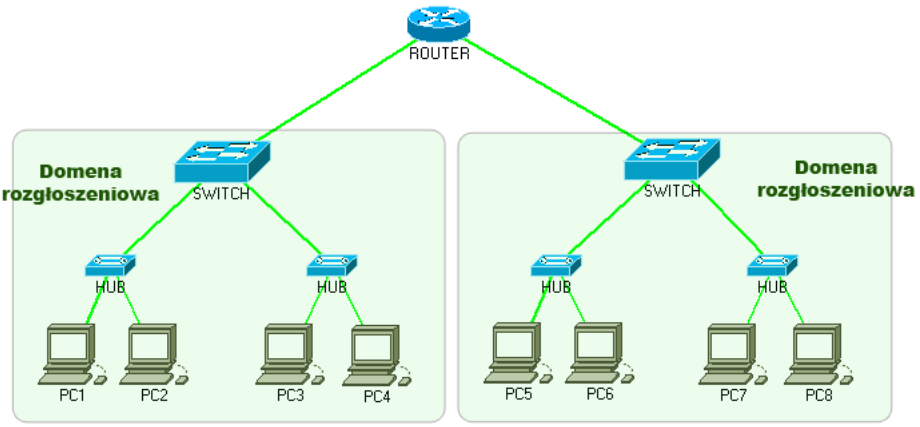
Domena kolizyjna



Rysunek 10: Domena kolizyjna

Jeżeli poprzez jedno medium transmisyjne (kabel) co najmniej dwa urządzenia transmitują dane może dojść do kolizji. Kolizje są zjawiskiem niepożądanym – powodują, iż żadna z wysłanych w tym momencie informacji nie dotrze do adresata. Jednak kolizji w sieciach, w których dostęp do nośnika przyznawany jest na zasadzie rywalizacji, nie da się uniknąć. Obszar sieci, w którym może dojść do kolizji nazywamy domeną kolizyjną. Możemy zmniejszać ryzyko kolizji odpowiednio projektując sieć komputerową. Należy również pamiętać, iż maksymalna ilość urządzeń w domenie kolizyjnej wynosi 1024. Przy czym im więcej urządzeń, tym większe prawdopodobieństwo wystąpienia kolizji. Domenę kolizyjną ograniczają urządzenia pracujące w warstwach wyższych niż pierwsza modelu OSI. Tym samym koncentratory i huby będą wewnątrz domeny kolizyjnej, a switch (przełącznik) czy router będą ją ograniczać.

Domena rozgłoszeniowa

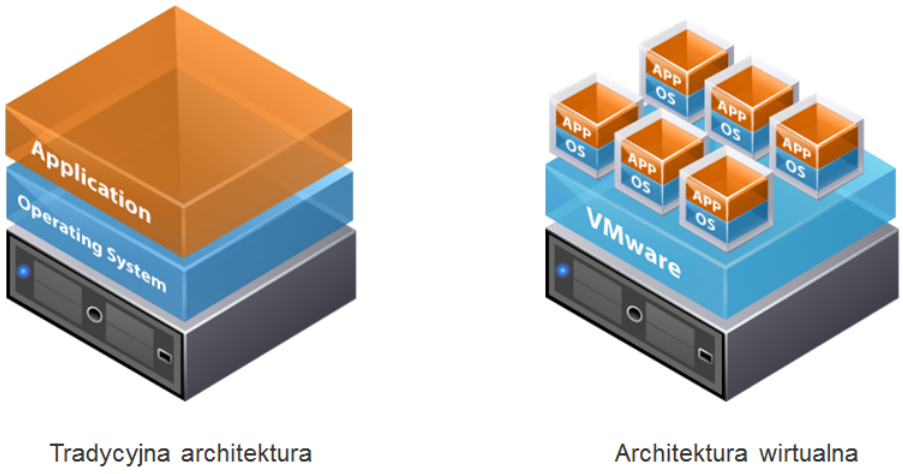


Rysunek 11: Domena rozgłoszeniowa

Domena rozgłoszeniowa to taki obszar sieci, do którego dotrze informacja wysyłana przez jedno urządzenie do wszystkich innych – broadcast. Ruch rozgłoszeniowy jest przekazywany poprzez urządzenia pierwszej i drugiej warstwy modelu OSI, tj. koncentratory, huby, mosty czy switche. One zwiększają obszar domeny rozgłoszeniowej. Ograniczają go natomiast urządzenia trzeciej warstwy – routery. Można również utworzyć sieć wirtualną VLAN (sieć komputerowa wydzielona logicznie w ramach innej, większej sieci fizycznej), która również ograniczy obszar domeny rozgłoszeniowej.

15 S5 – Charakterystyka wybranej techniki wirtualizacji

Wirtualizacja jest to metodologia polegająca na dzieleniu zasobów komputera na wiele środowisk. Działają one poprzez zastosowanie jednej lub wielu technologii, np. sprzętowe lub softwarowe partycjonowanie, tworzenie maszyny wirtualnej, przydzielanie czasu.



Rysunek 12: Wirtualizacja serwerów

- **Obszary wirtualizacji:**
- **Wirtualizacja serwerów** (ang. *Server Virtualization*) – technologia ta umożliwia wielu aplikacjom działanie na wielu systemach operacyjnych uruchomionych na tym samym fizycznym serwerze z wykorzystaniem wolnej mocy. Dzięki temu można w pełni wykorzystać moc obliczeniową i zasobową serwerów, stawiając kolejne maszyny wirtualne na jednym fizycznym serwerze – zamiast kupować kolejne serwery fizyczne. Dodatkową zaletą jest też szybsze dostarczenie usług dla biznesu w ciągu bardzo krótkiego czasu tworzenia nowego wirtualnego systemu operacyjnego gotowego do pracy. Ponadto podstawowe zadania konserwacyjne związane z serwerami są znacznie uproszczone i przyśpieszone.
- **Wirtualizacja stacji roboczych** (utrzymywana na kliencie) (ang. *Client-HostedDesktop Virtualization*) – to rodzaj wirtualizacji, która oddziela system operacyjny kliencki od sprzętu fizycznego i umożliwia uruchomienia maszyn wirtualnych na jednym komputerze PC obok klienckiego systemu operacyjnego hosta. Może być centralnie zarządzana, a obrazy maszyn wirtualnych mogą być dostarczane również centralnie. Technologię tę stosuje się wtedy, gdy zachodzi potrzeba zapewnienia zgodności aplikacji z systemem operacyjnym. Aplikacje są zainstalowane w maszynie wirtualnej i tam wykonuje się ich kod. Dostarczenie aplikacji do użytkownika odbywa się najczęściej w trybie seamless, czyli bez dodatkowej otoczki pulpitu – jedynie widok samej aplikacji.
- **Wirtualizacja stacji roboczych** (utrzymywana na serwerze)(ang. *Server-Based Desktop Virtualization*) – znana bardziej pod nazwą **Wirtualna infrastruktura stacji roboczych** (ang. *Virtual Desktop Infrastructure – VDI*). Jest to przeniesienie klasycznego środowiska pracy użytkownika, opartego na stacjach roboczych, do centrum przetwarzania danych, gdzie stacja robocza jest uruchomiona w postaci maszyny wirtualnej, a dostęp do niej zapewniony

jest drogą sieciową. Połączenie może być zrealizowane z dowolnego komputera PC, laptopa lub cienkiego klienta, który jest preferowaną formą dostępu do VDI. Sam w sobie VDI nie jest jedną technologią – to połączenie kilku niezależnych rozwiązań, które w podstawowej architekturze łączy ze sobą wirtualizację serwerów z wirtualizacją prezentacji.

- **Wirtualizacja sieci** (ang. *Network Virtualization*), którą można podzielić na dwa rodzaje – zewnętrzną oraz wewnętrzną. Zewnętrzna wirtualizacja sieci to znana od lat technologia VLAN (802.1Q), czyli logiczny podział segmentów sieci na fizycznym sprzęcie sieciowym przez znakowanie ramek. Wewnętrzna natomiast to rozwiązanie polegające na tworzeniu wirtualnych przełączników i portów na poziomie hiperwizora. Takie zwirtualizowane przełączniki dostarczają komunikację sieciową dla maszyn wirtualnych oraz łączą je z zewnętrzną fizyczną siecią.
- **Chmura prywatna** (ang. *Private Cloud*) to realizacja koncepcji chmury wewnątrz własnej firmy, czyli na własnych serwerach i własnym oprogramowaniu. W wersji podstawowej w chmurze prywatnej można zrealizować model infrastruktury jako usługi (ang. Infrastructure as a Service – IaaS). Bardziej złożone chmury prywatne mogą realizować model platformy jako usługi (ang. Platform as a Service – PaaS). Odbiorca mocy obliczeniowej lub zasobów nie wie tak naprawdę, w którym miejscu infrastruktury firmy się znajduje, gdyż jego przestrzeń może dynamicznie wędrować między serwerami oraz dynamicznie się skalować. Podsumowując, chmura prywatna to nic innego, jak usługi przetwarzania w chmurze świadczone przez dział IT dla własnej firmy.

Zalety Wirtualizacji:

- redukcja kosztów w aspekcie długoterminowym,
- ograniczenie rozrostu serwerowni,
- konsolidacja zasobów,
- elastyczność przydziału zasobów,
- łatwe testowanie nowych aplikacji oraz sieci komputerowych,
- możliwość uruchomienia większej ilości usług na jednym sprzęcie.
- korzystanie ze starszych aplikacji, niekompatybilnych z obecnie produkowanym sprzętem czy nowymi systemami,
- dostępność na bardzo wysokim poziomie.

Wady:

- Infrastruktura informatyczna, musi być bardzo wydajna i niezawodna,
- koszt inwestycji w mocne serwery,
- zakup licencji,
- wirtualne maszyny mogą być w niektórych przypadkach mniej wydajne od tradycyjnych rozwiązań,
- potrzeba wykwalifikowanej kadry zarządzającej.

My podczas zajęć korzystaliźmy z wirtualizacji VirtualBoxa, emulując np. Windowsa 7, czy CentOS-a.

Typy wirtualizacji

- **Parawirtualizacja** – technika wirtualizacji, w której wirtualizowany system operacyjny (Gość – ang. *Guest*, Partycja – ang. *Partition* lub Domena – ang. *Domain*) współpracuje ze środowiskiem operacyjnym komputera w zakresie obsługi tych elementów sprzętowych, których obsługa kolidowałaby z działalnością innych środowisk wirtualizowanych.

  Parawirtualizacja jest to jedna lub wiele maszyn wirtualnych działających obok systemu host. Parawirtualizacja korzysta z hypervisora typu 1, monitor działa tutaj bezpośrednio na sprzęcie i udostępnia go (lub nie) systemom guest. Jeden z wirtualizowanych systemów pełni rolę zarządzającą, może rozdzielać zasoby pośród pozostałych systemów i pełnić pełną kontrolę nad maszyną wirtualną i jej zasobami. System, który mógłby być uruchomiony jako guest za pomocą parawirtualizacji, musi mieć zmodyfikowane jądro w taki sposób, aby zamiast odwołań do sprzętu, jak również standardowych odwołań do sprzętu, które są niedozwolone podczas parawirtualizacji, trzeba używać odwołań typu hypercall. Odwołanie to po prostu interfejs pomiędzy monitorem a systemami guest. Mechanizm zapewnia tutaj bardzo dobrą wydajność, bardzo zbliżoną do natywnego działania systemu. Bardzo dobra wydajność systemów guest oraz łatwość w modyfikowaniu jądra systemu guest w celu przystosowania go do parawirtualizacji jest jego zaletą. Natomiast wadą jest konieczność modyfikowania systemu guest, ponieważ nie zawsze jest możliwe uzyskanie takiej wydajności. Niestety modyfikacja jądra niesie za sobą również koszty związane z dostosowaniem systemu guest do odpowiedniego hypervisora oraz musimy tę procedurę ponowić dla każdego innego hypervisora. Implementacją parawirtualizacji w postaci VMI (Virtual Machine Interface) jest interfejs komunikacji pomiędzy systemem guest a hypervisorem proponowanym przez VMware. Można również powiedzieć, że VMI jest to próba ujednolicenia interfejsu dla wirtualizacji. Celem było także stworzenie jednego interfejsu komunikacji, który byłby zaimplementowany na wielu hypervisorach i w miarę łatwa byłaby implementacja na systemach guest. Zaletą VMI jest to, że implementacja ta nie jest przynależna do jednego hypervisora tylko jest otwarta. Implementacja systemu guest dałaby raz możliwość uruchomienia go we wszystkich liczących się hypervisorach.

- **Pełna wirtualizacja** – technika wirtualizacji, w której wirtualizowany system operacyjny (gość) ma wrażenie, że działa na prawdziwym, fizycznym sprzęcie (komputerze). W rzeczywistości odwołania wirtualizowanego systemu operacyjnego (gościa) do tych elementów fizycznych komputera, które kolidowałyby z działalnością innych środowisk wirtualizowanych lub systemu operacyjnego gospodarza (ang. host), są przechwytywane przez oprogramowanie wirtualizacyjne, a następnie emulowane. Emulacja taka spowalnia pracę wirtualizowanego środowiska, dlatego pożądane jest sprzętowe wspomaganie wirtualizacji.

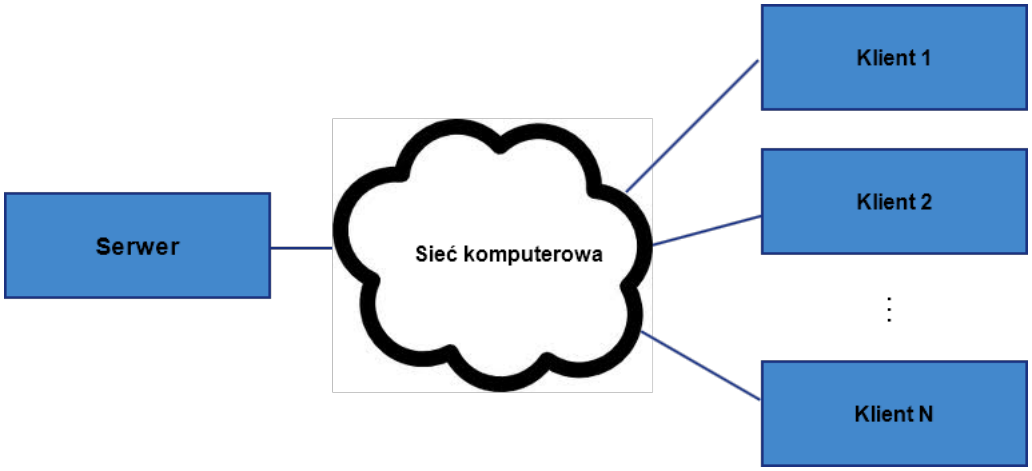
  Pełna wirtualizacja pozwala nam na wirtualizowanie dowolnie wybranego niezmodyfikowanego systemu operacyjnego. Największą zaletą pełnej wirtualizacji jest to, iż nie narzuca ona nam żadnych ograniczeń ani wymagań co do systemu guest. Wydajność w takim rozwiązaniu jest niestety mniejsza niż w przypadku parawirtualizacji. Zarówno za pomocą hypervisora typu 1, jak i typu 2 może być zrealizowana pełna wirtualizacja. Głównym wyzwaniem pełnej wirtualizacji jest realizowanie i przechwytywanie uprzywilejowanych instrukcji jądra guest, tak jak na przykład operacja I/O, przerwań czy operacji na pamięci. Większość instrukcji systemu guest wykonywana jest bezpośrednio na procesorze przez hypervisora, natomiast te

instrukcje, które wychodzą poza uprawnienia systemu guest, jak też odwołują się do sprzętu, powinny być przechwycone i emulowane. W przypadku kiedy zostanie użyty hypervisor typu 1 wymagane jest odpowiednie rozszerzenie sprzętowe, które umożliwia wirtualizację. Pełną wirtualizację określa się również pojęciem natywna wirtualizacja oraz wirtualizacja wspierana sprzętowo. Jeśli do pełnej wirtualizacji użyjemy hypervisora typu 2, to będziemy ją realizować przez mechanizmy takie jak binary translation oraz direct execution, często ze wsparciem pamięci podręcznej tych transakcji w postaci cache, można również użyć kompilacji instrukcji just in time.

16 S6 – Architektura warstwowa w internetowych aplikacji bazodanowych

Jeżeli mówimy o architekturze warstwowej (ang. *tier architectrue*) to zawsze mamy na myśli architekturę n warstwową, gdzie n $>$ 0. Najbardziej popularne architektury warstwowe to:

- **Architektura jednowarstwowa** - całe oprogramowanie zamknięte jest w jednej "warstwie". Przykładem są proste aplikacje desktopowe.
- **Architektura dwuwarstwowa** - oprogramowanie podzielone jest na dwie warstwy - przykładem architektury dwuwarstwowej jest klient-serwer. Oprogramowanie klienta to jedna warstwa a serwera druga.



Rysunek 13: Architektura klient-serwer

Komunikacja pomiędzy serwerem oraz klientami odbywa się za pomocą konkretnego protokołu rozumianego zarówno przez serwer jak i klienta. Mogą to być standardowe protokoły typu np. HTTP, SMTP, FTP.... lub dedykowane - zaprojektowane na potrzeby konkretnego rozwiązania.

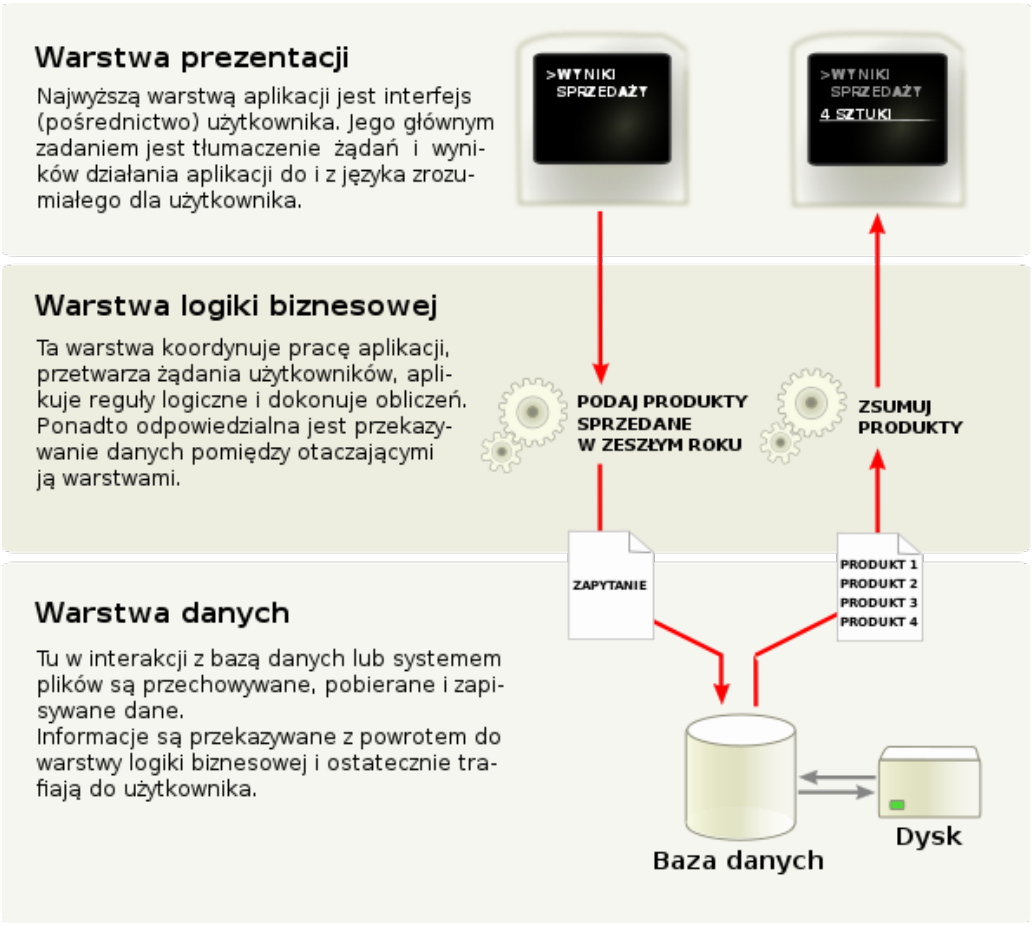
Klient z reguły zawiera interfejsy użytkownika, a serwer posiada dostęp do bazy danych. Serwer to nie tylko skomplikowana dedykowana aplikacja na potrzeby jakiegoś rozwiązania. Rolę serwera może pełnić np. RDBMS bez dodatkowej logiki aplikacyjnej.

Najbardziej popularne dzisiaj rozwiązania architektury klient-serwer to oczywiście sieć www, gdzie rolę serwerów pełnią serwery www a rolę klientów: przeglądarki internetowe.

- **Architektura trójwarstwowa** - w architekturze tej wyróżnia się następujące warstwy:
  - **Warstwa prezentacji** (ang. *presentation tier*) - odpowiedzialna za interakcję z użytkownikiem końcowym (wyświetlanie i wprowadzanie danych). W tej warstwie działają aplikacje klienckie takie jak np. przeglądarki internetowe
  - **Warstwa biznesowa** (ang. *business tier*) - odpowiedzialna za przetwarzanie danych (żądań) od użytkownika. Tutaj też przygotowywane są dane wysłane do warstwy prezentacji (aplikacji klienckich). W tej warstwie realizowane są wszelkiego rodzaju funkcjonalności

biznesowe. Z drugiej strony logika warstwy odpowiedzialna jest za komunikację z warstwą danych. Warstwa biznesowa stanowi swego rodzaju pomost pomiędzy warstwą aplikacji i warstwą danych dodając elementy przetwarzania.

– **Warstwa danych** (ang. *persistance tier*) - odpowiedzialna za przechowywane danych. W tej warstwie mamy np. bazę danych.



Rysunek 14: Architektura trójwarstwowa

Zalety podziału architektury na warstwy:

- podział na niezależne komponenty,
- łatwość podmiany,
- rozdzielenie całego systemu na dedykowane dla danej warstwy zadania,
- ułatwia skalowanie i zarządzanie.

17 S7 – Współczesne algorytmy kryptograficzne

Algorytmy kryptograficzne podzielić możemy na algorytmy symetryczne (z kluczem prywatnym) (symmetric algorithm) lub asymetryczne (z kluczem publicznym). W przypadku tych pierwszych do zaszyfrowania i odszyfrowania wiadomości używamy takiego samego klucza. Wśród algorytmów symetrycznych istnieje podział na algorytmy blokowe(block algorithm) jak i strumieniowe (stream algorithm). W pierwszym przypadku tekst jawny dzielony jest na bloki o odpowiedniej długości (najczęściej 64 lub 128 bitów) a następnie każdy blok jest szyfrowany oddzielnie). W algorytmach strumieniowych - tekst jest szyfrowany bit po bicie lub bajt po bajcie. W przypadku algorytmów asymetrycznych do szyfrowania używamy klucza publicznego (public key) osoby, do której przesyłamy wiadomość a do odszyfrowania osoba ta używa własnego klucza prywatnego(private key).

- **symetryczne** – ten sam klucz używany do szyfrowania i deszyfrowania po obu stronach komunikacji:
  - DES – dane szyfrowane są w blokach 64-bitowych kluczem 56-bitowym, algorytm dość bezbezpieczny, ale już go zbruteforce'owali,
  - 3DES – 3 x DES – zdecydowanie wzmacnia siłę algorytmu,
  - RC4 – szyfr strumieniowy, 40- albo 128-bitowe klucze, zastosowanie: WEP, WPA, SSH,
  - AES – blokowy, klucze 128, 192, 256, obecnie najbardziej popularny (wygrywa stosunkiem bezpieczeństwo/szybkość), jego użytek jest bezpłatny, łatwy w implementacji na różnych platformach sprzętowych, zastosowanie: WPA2,
  - BlowFish – popularny (ssh, pgp) algorytm, również bezpłatny, dobrze przebadany kryptoanalitycznie, można go efektywnie implementować, podczas pracy mało pamięciożerny
- **asymetryczne** – do szyfrowania i deszyfrowania używa się dwóch różnych kluczy: jeden z kluczy jest jawny (znany wszystkim), a drugi prywany (znany tylko jednej osobie lub grupie osób), w zależności od zastosowań jawny może być albo klucz służący do szyfrowania albo deszyfrowania. Cechy algorytmów asymetrycznych wykorzystuje się przy cyfrowym podpisywaniu, wszędzie tam, gdzie wymaga się pewności co do tożsamości i niezaprzeczalności nadawcy (odbiorcy). Przykłady:
  - RSA – do szyfrowania wykorzystuje się funkcje matematyczne (arytmetykę modulo), kluczami są pary liczb, siła algorytmu tkwi w tym, że jedną z liczb w parze jest iloczyn dwóch bardzo dużych liczb pierwszych, a nie istnieje, przynajmniej na razie, żadna efektywna metoda, rozdziału liczb na czynniki pierwsze. Dopiero znajomość rozkładu drugiej liczby z pary pozwoliłaby na deszyfrowanie znając klucz służący do szyfrowania,
  - LUC – podobnie jak RSA, ale wykorzystuje tzw. ciągu Lucasa, a nie arytmetykę modulo, ponadto nie jest ograniczony umowami patentowymi (jak RSA),
  - El-Gamal – wykorzystuje logarytmy dyskretne (nie ma efektywnej metody, by je liczyć), najpopularniejszy obok RSA

Algorytmy asymetryczne są wolniejsze od symetrycznych

Wykorzystanie:

- **Podpisu cyfrowy** umożliwia potwierdzenie autentyczności dokumentu, weryfikację sygnatariusza, zapewnienie niezmienności treści od momentu podpisania oraz uniemożliwienia wyparcie się faktu podpisania dokumentu. Podpis cyfrowy wykonywany jest przy pomocy klucza prywatnego, a weryfikowany za pomocą klucza publicznego. Oznacza to, że tylko jedna osoba może dokonać podpisu, właściciel klucza prywatnego, natomiast weryfikacji dokumentu może dokonać każdy, wystarczy, że z ogólnodostępnego katalogu pobierze klucz publiczny. Sytuacja jest odwrotna niż w przypadku szyfrowania. Podczas szyfrowania najpierw używamy klucza publicznego, a później klucza prywatnego. Gdy dokonujemy podpisu cyfrowego najpierw używamy klucza prywatnego później publicznego.
- Standard X.509 opisuje budowę **certyfikatu klucza publicznego**. Może on zostać użyty, do przetransportowania różnych informacji związanych zarówno z procedurą szyfrowania jak i z weryfikacją podpisów. Duża część tej informacji jest opcjonalna, a zawartość obowiązkowych pól może się również zmieniać. Certyfikat klucza publicznego jest ochroniony przez podpis cyfrowy wydawcy. Użytkownicy certyfikatów wiedzą, że jego zawartość nie została sfałszowana od momentu, gdy została podpisana, jeśli tylko podpis może zostać zweryfikowany. Certyfikaty zawierają grupę wspólnych pól, ponadto mogą zawierać opcjonalną grupę rozszerzeń. Jest dziesięć elementarnych pól: sześć obowiązkowych i cztery opcjonalne. Obowiązkowe pola to: numer seryjny, algorytm podpisu, nazwa wydawcy certyfikatu, okres ważności certyfikatu, klucz publiczny i nazwa podmiotu. Podmiot jest osobą, bądź organizacją, która jest w posiadaniu odpowiadającego klucza prywatnego. Opcjonalne pola to: numer wersji, dwa unikalne identyfikatory i rozszerzenie.
- **Protokół SSL** (ang. Secure Sockets Layer), opracowany przez firmę Nescape, stał się standardem komunikacji w Internecie. Został wprowadzony w celu zapewnienia prywatności komunikacji pomiędzy dwoma komputerami (serwerem i klientem) oraz dostarczenia autentyfikacji serwera i opcjonalnie klienta. Swoje bezpieczeństwo opiera na bezpieczeństwie dostarczonym przez certyfikaty klucza publicznego. SSL posługuje się protokołem komunikacyjnym, (np. TCP) przy wysyłaniu i odbieraniu wiadomości. Zaletą tego protokołu jest fakt, że jest on niezależny od aplikacji. Ustalenie bezpiecznego kanału transmisji może nastąpić przed przesłaniem pierwszej porcji danych przez inne protokoły (np. FTP, TELNET, HTTP). Wszystkie dane przesyłane za pomocą niezabezpieczonych protokołów, dzięki SSL, będą poufne.

18 S8 – Metody projektowania gier komputerowych

Produkcja gier zwyczajowo nie odpowiada tradycyjnym cyklom życia programu takim jak model kaskadowy. Jedną z używanych metod podczas produkcji jest programowanie zwinne. Metoda ta jest skuteczna, ponieważ większość projektów nie zaczyna się od jednoznacznych wymogów. Proces produkcji dzieli się na **preprodukcję**, **produkcję właściwą** i **etap konserwacji**. W etapach tych wyróżnić można także **kamienie milowe**.

Etapy projektowania gry komputerowej:

- **Preprodukcja** jest początkowym etapem, w którym twórcy skupiają się wokół projektowania elementów rozgrywki i tworzenia dokumentów. Jednym z głównych celów tej fazy jest stworzenie jednoznacznej i łatwej do zrozumienia dokumentacji, która zawiera wszystkie wytyczne projektu i harmonogram prac. Podczas preprodukcji powstają prototypy, które często służą jako proof-of-concept albo możliwość przetestowania gry.

  Czasem projekt gry dzieli się na wiele dokumentów, powstających w podanej kolejności:

  - **Game Concept Document:** Podstawa do stworzenia wizji – dane na temat gatunku, docelowej grupy odbiorców, świata gry. W tym dokumencie także jest zawarty dział "Wykrywanie wymagańńa podstawie którego pozyskamy dane, informacje potrzebne do stworzenia wizji.
  - **Vision Document:** Rozwinięcie dokumentu koncepcji gry. Tutaj już znajdą się pierwsze opisy przewidywanych używanych technologii, dokładniejsze informacje o świecie gry, ważne zależności mechaniki, a nawet pierwsze statystyki na temat postaci, czy misji.

  Następne powstają zazwyczaj jednocześnie:

  - **Art Design Document:** Koncepcje, instrukcje dla grafików, muzyków i innych artystów. Także kolorystyka tworzonych obiektów, ich ogólny klimat.
  - **Project Timeline Document:** Terminy, daty i kamienie milowe, których zadaniem jest regulowanie wykonywania zadań według ściśle określonego czasu.
  - **Testing Document:** Wytyczne dla zespołu testerów, jakie elementy gry są najbardziej narażone na błędy.

  Osobną grupą są dokumenty techniczne, które opisują dane elementy – sposób ich implementacji. Przykładowo przed zaprogramowaniem systemu map (wczytywanie – wyświetlanie) jest tworzony dokument opisujący jak należy ten element zaimplementować, czasem także stara się on zwrócić uwagę na newralgiczne punkty tworzonego kodu, w których programista może stworzyć wiele błędów.

- **Produkcja właściwa** jest główną częścią tworzenia gry, podczas której powstaje kod źródłowy, grafiki i oprawa dźwiękowa. Testowanie gier komputerowych rozpoczyna się, gdy tylko pierwszy kod źródłowy zostanie napisany i wzrasta w trakcie procesu produkcji. Opinie testerów mogą mieć wpływ na ostateczne decyzje dotyczące wykluczenia czy integracji elementów gry w jej ostatecznej wersji. Wprowadzenie wcześniej niezaangażowanych testerów ze świeżą perspektywą może pomóc w identyfikacji nowych błędów. Beta testy mogą obejmować ochotników, na przykład jeśli gra zawiera tryb wieloosobowy. Dźwięki występujące w grze mogą

być podzielone na trzy typy: efekty dźwiękowe, muzykę i dialogi postaci. Muzyka może być stworzona przy użyciu programów lub nagrana na żywo.

W trakcie wczesnej fazy produkcji gra wchodzi w wersję alfa. Jest to moment, kiedy główne elementy rozgrywki zostały zaimplementowane. Zawartość gry na tym etapie może zostać ponownie przejrzana po wcześniejszym testowaniu. W wersji beta wszystkie funkcje zostały zaimplementowane, a twórcy skupiają się na usuwaniu znalezionych błędów. Testy wersji beta odbywają się na dwa do trzech miesięcy przed premierą. Gdy gra jest wysłana do tłoczni w celu masowej produkcji, zostaje oznaczona wtedy jako wersja „złota".

- **Postprodukcja** - po zakończeniu prac nad projektem rozpoczyna się etap konserwacji gry.

19 S9 – Zasady projektowania bezpiecznych systemów i sieci komputerowych

Podstawą bezpieczeństwa systemu informatycznego jest dobrze opracowany projekt, wdrożony z użyciem właściwie dobranych technologii renomowanych producentÛw, zarządzany przez wykwalifikowaną kadrę informatyczną. Projektowane zabezpieczenia powinny być oparte w znacznej mierze na wynikach specyfikacji wymagań bezpieczeństwa, a także ogólnej teorii zabezpieczeń (m.in. wymagane jest dokonanie weryfikacji odporności systemu na strategię włamań Island Hopping Attack). Tworzenie zabezpieczeń systemu informatycznego powinno odbywać się w przemyślany, wcześniej szczegółowo zaplanowany sposób zgodnie ze sprawdzoną metodyką.

Podczas projektowania bezpiecznych sieci i systemów należy pamiętać, by używać **jedynie wymaganego** sprzętu i oprogramowanie. Każda nadmiarowa rzecz może powodować istotne luki w bezpieczeństwie systemu. Podczas nadawania **uprawnień** trzeba dawać ich **tylko tyle, ile jest faktycznie potrzebne**. Sieć powinna być zaprojektowana w ten sposób, aby maksymalnie **ograniczyć możliwość podłączania niepotrzebnych i obcych urządzeń**. Istotnym elementem jest także uwzględnienie podczas projektowania mechanizmów monitorujących i sterujących siecią – oczywiście muszę one spełniać wymienione powyżej warunki. Bezpieczny system komputerowy również musi spełniać podobne warunki. Procesy muszą działać zawsze możliwie **z najniższymi możliwymi uprawnieniami. Udostępniane** powinny być **jedynie naprawdę wymagana elementy systemu** (np. baza danych postawiona na serwerze WWW wcale nie musi być bezpośrednio dostępna dla klientów z zewnątrz – wystarczy, że ma do niej dostęp usługa HTTP). Ponadto mechanizmy ochronne muszę być **możliwie proste, bez cudowania** i zaimplementowane w możliwie **niskich warstwach systemu**. Ważnym – chyba często pomijanym – aspektem jest taka budowa systemu bezpieczeństwa, **aby nie drażnił on i nie zniechęcał użytkowników.**

20 S10 – Protokoły routingu

Służą do wymiany informacji, pomiędzy routerami, dotyczącymi tras pomiędzy sieciami komputerowymi. Tradycyjne trasowanie jest bardzo proste, bo polega na wykorzystaniu tylko informacji o następnym "przeskoku"(ang. hop). W tym przypadku router tylko kieruje pakiet do następnego routera, bez uwzględnienia na przykład zbyt wielkiego obciążenia czy awarii na dalszej części trasy.

Mimo że dynamiczne trasowanie jest bardzo skomplikowane, to właśnie dzięki niemu Internet jest tak elastyczny i rozwinął się o ponad 8 rzędów wielkości w ciągu ostatnich 40 lat.

Protokoły trasowania robią dwie proste rzeczy:

- mówią światu, kim są sąsiedzi
- mówią sąsiadom, jak wygląda świat

Metryka trasowania jest wartością używaną przez algorytmy trasowania do określenia, która trasa jest lepsza. Brane są pod uwagę: szerokość pasma, opóźnienie, liczba przeskoków, koszt ścieżki, obciążenie, MTU, niezawodność, koszt komunikacji. Tylko najlepsze trasy przechowywane są w tablicach trasowania, podczas gdy inne mogą być przechowywane w bazach danych. Jeśli router korzysta z mechanizmów równoważenia obciążenia (ang. *load balancing*), w tablicy trasowania może wystąpić kilka najlepszych tras. Router będzie je wykorzystywał równolegle, rozpraszając obciążenie równomiernie pomiędzy trasami.

Protokoły sieci rozległych:

RIP – router wysyła do sąsiednich routerów informacje dotyczące jego tablicy routingu. Router otrzymujący taką informacje aktualizuje swoją tablicę, zapisując jednocześniej odległość (w sensie liczby skoków), jaka dzieli go od danej sieci. Wybierana są ścieżki o najmniejszym koszcie. Używa algorytmu Forda-Bellmana. W wersji RIPv1 używa trasowania klasowego, w wersji RIPv2 – trasowania bezklasowego. Protokół wewnętrzny, wektora odległości.

OSPF – Shortest Path First, wybierana jest najkrótsza ścieżka do miejsca przeznaczenia. Routery wysyłają do siebie pakiety informacyjne, dzięki temu każdy z nich ma bogatą informacje o topologii sieci. Następnie używany jest algorytm Dijkstry do znalezienia najbardziej korzystej (najkrótszej) ścieżki. O wiele bardziej skalowalny i dokładny niż RIP. Protokół wewnętrzny, stanu łącza.

IS-IS – praktycznie to samo, co OSPF, ale: OSPF działa na (z ang. on top of IP) warstwie drugiej modelu ISO/OSI, natomiast IS-IS jest protokołem warstwy trzeciej. Wynika z tego, że OSPF jest uzależniony w działaniu od protokołu IP, natomiast IS-IS ma serdecznie gdzieś, jaki protokół warstwy sieciowej jest aktualnie używany do normalnego ruchu w sieci,

IGRP – protokół wewnętrzny, wektora odległości. Podobnie jak RIP, ale w RIP-ie jako metryki używa się liczby skoków do miejsca przeznaczenia, natomiast tutaj metryką jest szerokość pasma, jego obciążenie, opóźnienie i niezawodność. Obsługuje średnie sieci. Protokół własnościowy CISCO. Jego następcą jest **EIGRP**, który używa już trasowania bezklasowego.

BGP – protokół zewnętrzny, używany do wymiany informacji pomiędzy różnymi systemami autonomicznymi. Nie używa tradycyjnych metryk, ale zestawu atrybutów. Aktualna wersja jest podstawą działania Internetu. Protokół działa przy użyciu protokołu TCP, przez co cechuje go duża niezawodność. Routery BGP wymieniają informacje z innymi routerami BGP o ich osiągalności w sieci Internet, informacje te zawierają listę systemów, przez jakie muszą przejść dane, aby dotrzeć do adresata