23 DONE

24 DONE

25 DONE

26 DONE

27

28

29

30

31 DONE // S: to jak będę miała jeszcze czas to usunę te za bardzo szczegółowe rzeczy ;), edit: chyba nie znajdę na to czasu :/

32 DONE

33 DONE

34

35 DONE

36

37 DONE

38 DONE

39 DONE

40 DONE

41 DONE

42 DONE A: Początkowo opis co to jest i jak działa sieć, właściwości i zastosowania poniżej.

43 DONE

44 Trochę wrzuciłem (zarządzanie konfiguracjami), chyba przydałoby się jakieś poszerzenie na temat zarządzania zmian.

45

46 DONE

**OPRACOWANIE ZAGADNIEŃ SI**

# 23. Algorytmy ewolucyjne w systemach informacyjnych

Działanie algorytmu ewolucyjnego można opisać następująco: algorytm ewolucyjny rozpoczyna proces przeszukiwania od utworzenia populacji potencjalnych rozwiązań nazywanych osobnikami, które są reprezentowane przez chromosomy zawierające genetyczną informację o osobnikach. W każdym ewolucyjnym kroku, nazywanym generacją, chromosomy są dekodowane i ocenianie zgodnie z pewnym z góry przyjętym kryterium jakości nazywanym przystosowaniem (funkcją przystosowania może być na przykład funkcja celu), a następnie przeprowadzana jest selekcja w celu eliminacji osobników ocenionych jako najgorsze. Osobniki wykazujące wysokie przystosowanie podlegają mutacji oraz rekombinacji przeprowadzanej przy pomocy operatora krzyżowania. Sama selekcja nie wprowadza żadnego nowego osobnika do populacji, tj. nie znajduje nowych punktów w przestrzeni poszukiwań, natomiast takie punkty wprowadzane są przez krzyżowanie i mutację. Dzięki krzyżowaniu ewolucyjny proces może się przesuwać w kierunku obiecujących obszarów w przestrzeni poszukiwań. Mutacja zapobiega zbieżności do lokalnego optimum. W wyniku działania operatora krzyżowania i mutacji tworzone są nowe rozwiązania, z których następnie budowana jest populacja następnej generacji. Warunkiem zakończenia algorytmu może być na przykład pewna określona liczba generacji albo osiągnięcie zadawalającego poziomu przystosowania Niech P(t) oznacza populację w generacji t.

Ogólny schemat działania algorytmuewolucyjnego jest następujący:

1. t = 0

2. Wygeneruj i oceń początkową populację P(t)

3. Dopóki warunek stopu nie jest spełniony wykonuj:

3.1. t = t + 1

3.2. Wybierz P(t) z P(t - 1)

3.3. Zmień P(t) stosując operator krzyżowania i mutacji

3.4. Oceń P(t)

Występujące w realnym świecie problemy charakteryzują się zwykle bardzo dużą liczbą zmiennych (dyskretnych lub ciągłych), dużą złożonością przestrzeni poszukiwań (wiele ograniczeń i celów, które na dodatek mogą być ze sobą sprzeczne).

ALGORYTMY EWOLUCYJNE I ICH ZASTOSOWANIA

Problemy te mogą się zmieniać w czasie (być dynamiczne), co pociąga za sobą konieczność szybkiego otrzymywania dobrych rozwiązań. Ważnym obszarem zastosowań algorytmów ewolucyjnych jest harmonogramowanie i planowanie procesów przemysłowych. Problemy harmonogramowania, gdzie należy przydzielić zasoby do kolekcji zadań, są zwykle trudne do rozwiązania z powodu występowania różnorodnych ograniczeń oraz złożonych struktur produktów. Stosowane tutaj techniki programowania matematycznego pozwalają zwykle na uzyskanie rozwiązań tylko dla problemów o małych rozmiarach. W większości zastosowań algorytmów genetycznych do problemów harmonogramowania, cały harmonogram jest kodowany za pomocą chromosomu (chromosom przechowuje zakodowany harmonogram, a więc określa kolejność wykonywania poszczególnych czynności), co wymaga następnie stosowania odpowiednio zaprojektowanych operatorów genetycznych tak, aby tworzone w kolejnych populacjach chromosomy reprezentowały rozwiązania dopuszczalne (np. aby wykonanie tej samej czynności nie powtarzało się). Kolejnym obszarem zastosowań algorytmów ewolucyjnych jest przemysł chemiczny. Zakłady chemiczne zawierają urządzenia takie jak pompy, destylatory i reaktory chemiczne, które tworzą złożoną sieć wzajemnie powiązanych strumieni materiału, ogrzewania i informacji, przeznaczonych do przeprowadzania procesu chemicznego, podczas którego surowy materiał jest przekształcany w wymagany produkt. Optymalizacja w zakładach chemicznych polega na modyfikacji struktury parametrów operacyjnych tak, aby znaleźć globalne optimum w celu wykonania pewnego zadania chemicznego. Klasyczne podejścia do rozwiązywania problemów chemicznych nie pozwalają na otrzymanie rozwiązań w zadowalającym czasie. Techniki gradientowe zbiegają do lokalnych optimów i nie są odpowiednie dla występujących tu problemów nieróżniczkowalnych. Stąd też wynika zainteresowanie stosowaniem technik opartych na algorytmach ewolucyjnych w tym klasycznych algorytmów genetycznych. Sukces w stosowaniu algorytmów genetycznych w tak skomplikowanym środowisku wynika z faktu, że na ogół przy pomocy chromosomu kodowana jest tylko niezbędna cześć problemu, co pociąga za sobą niewielki rozmiar chromosomu, co z kolei zwiększa szansę na otrzymanie dobrego dopuszczalnego rozwiązania. Innym przykładem związanym z zastosowaniem przemysłowym może być wykorzystanie algorytmu genetycznego do kontroli fermentacji piwa. Algorytm genetyczny jest tutaj użyty do dopasowania profilu temperatury mieszaniny w ustalonym okresie czasu. Algorytmy ewolucyjne znalazły również dość szerokie zastosowanie w medycynie. Większość medycznych decyzji może być sformułowana jako poszukiwanie w pewnej odpowiedniej przestrzeni. Na przykład radiolog planując serię naświetlań poszukuje najlepszego sposobu leczenia w przestrzeni wszystkich możliwych sposobów. Przestrzenie poszukiwań w medycynie są zwykle bardzo duże i złożone. Decyzje są oparte na testach klinicznych, które dostarczają ogromnych ilości danych. W oparciu o te dane trzeba ostatecznie podjąć jedną decyzję (np. patolog musi stwierdzić czy nowotwór jest łagodny czy złośliwy na podstawie cech komórek, czyli wśród wszystkich możliwych cech komórek poszukuje się zbioru cech, który pozwala postawić właściwą diagnozę). Algorytmy ewolucyjne są stosowane w medycynie do wykonywania zadań, które mogą być podzielone na trzy grupy: (a) eksploracja danych, głównie w celu diagnozowania i prognozowania, (b) obrazowanie i przetwarzanie sygnałów, (c) planowanie i harmonogramowanie. (a) Eksploracja danych (odkrywanie wiedzy) jest procesem znajdowania wzorów, trendów i regularności poprzez „przesiewanie” dużych ilości danych. W medycznej eksploracji danych algorytmy ewolucyjne są zwykle wykorzystywane w celu znajdowania wartości parametrów ustawianych przez projektanta tak, aby przeszukiwane dane były interpretowane w sposób optymalny (np. znajdowanie wag w sieciach neuronowych). (b) Wiele danych medycznych jest wyrażanych za pomocą obrazów lub innych sygnałów. Algorytmy ewolucyjne znajdują tu zastosowanie do poprawiania działania algorytmów przetwarzających sygnały (np. filtrów lub kompresorów) przez znalezienie ich optymalnych parametrów. Mogą też zostać wykorzystane do bezpośredniego wyprowadzenia użytecznej informacji z dostarczonych danych. Algorytmy ewolucyjne szczególnie dobrze nadają się do rozwiązywania problemów planowania i harmonogramowania (c). Przykładem może tu być problem harmonogramowania dla pacjenta, poddawanego różnym medycznym procedurom i potrzebującego konsultacji różnych specjalistów, w celu optymalizacji zarówno czasu oczekiwania pacjenta jak również wykorzystania aparatury.

# 24. Jakość oprogramowania i jakość danych w systemach informacyjnych

Definicje jakości:

• stopień uwolnienia wyrobu od wad i błędów.

• stopień, w jakim określony wyrób zaspokaja potrzeby danego nabywcy (jakość rynkowa)

• stopień zgodności wyrobu z modelem, wzorcem lub wymaganiami (jakość zgodności)

• stopień, w jakim określony wyrób znajduje u konsumenta pierwszeństwo przed innym wyrobem w wyniku ich porównania (jakość preferencji)

• dający się wyodrębnić zespół cech istotny dla danego wyrobu, np. wielkość, wygląd, niezawodność

• stopień jednorodności i niezawodności wyrobu przy możliwie niskich kosztach i maksymalnym dopasowaniu do wymagań rynku.

• ogół cech i właściwości wyrobu lub usługi, które decydują o zdolności wyrobu lub usługi do zaspokajania stwierdzonych i przewidywanych potrzeb [ISO 8402]

• zdolność wyrobu, systemu lub procesu o określonym zestawie właściwości, do spełnienia wymagań klienta i wymagań stron zainteresowanych [ISO 9000:2000]

Najczęściej wymieniane cechy jakości produktu to:

• funkcjonalność

– użyteczność: stopień realizacji oczekiwanych funkcji

– praktyczność: komfort użytkowania, łatwość obsługi i konserwacji

– niezawodność: zdolność do pracy bezusterkowej

– trwałość: okres zachowania cech użytkowych

– bezpieczeństwo użytkowania

• satysfakcjonowanie

– ekskluzywność: prestiż nabywcy związany z posiadaniem produktu lub jego marką

– estetyczność: pozytywne odczucia osobiste

– prezentacja: forma zaoferowania (warunki sprzedaży: otoczenie, obsługa, sposób dostawy, opakowanie, certyfikaty, referencje itp.)

• koszt nabycia

Koło Deminga:

Zdaniem Deminga tylko proste metody i zrozumiałe narzędzia pozwalają na osiągnięcie efektów. Im metoda jest bardziej złożona, tym mniejsze są szanse na uzyskanie pozytywnych wyników jej stosowania. Deming zaproponował przejrzystą i powszechnie zrozumiałą zasadę doskonalenia jakości, nazwaną od nazwiska jego twórcy „kołem Deminga” lub modelem P-D-C-A (planuj – wykonaj – sprawdź – koryguj).

• Jeśli na etapie sprawdzania okaże się, że występują rozbieżności między zamierzeniami i efektami, to wówczas należy uruchomić działania korygujące, prowadzące do ich likwidacji.

• Często, pierwsze działania korygujące mogą być niewystarczające, dlatego też po powtórnym sprawdzeniu musimy przystąpić do kolejnych działań korygujących. Postępujemy w ten sposób tak długo, aż uzyskamy satysfakcjonujące wyniki. Dopiero wówczas przechodzimy do następnego etapu określanego mianem - działaj.

• Zorganizowanie ciągłej poprawy jakości w oparciu o zasadę „koła Deminga” przyniosło sukcesy firmom japońskim, amerykańskim oraz europejskim i stało się podstawą doskonalenia wszelkich działań realizowanych w ramach nowego podejścia do systemów zarządzania jakością, odzwierciedlonego w znowelizowanych normach jakościowych z roku 2000, uwzględniających podejście procesowe.

Zapewnienie jakości oprogramowania:

„Zapewnienie jakości są to wszystkie zaplanowane i systematyczne działania, niezbędne do uzyskania i utrzymania odpowiedniego stopnia wiarygodności, że system spełnia ustalone wymagania techniczne”

• ZJO oznacza sprawdzanie:

• czy plany są zdefiniowane zgodnie ze standardami;

• czy procedury są wykonywane zgodnie z planami;

• czy produkty są implementowane zgodnie z planami.

• Kompletne sprawdzenie jest zwykle niemożliwe. Projekty bardziej odpowiedzialne powinny być dokładniej sprawdzane odnośnie jakości.

• Najbardziej istotnym kryterium przy zapewnianiu jakości jest ryzyko. Najczęstszymi czynnikami ryzyka utraty jakości są:

• nowość projektu,

• złożoność projektu,

• niedostateczne wyszkolenie personelu,

• zbyt małe doświadczenie personelu,

• niesformalizowane (tworzone i zarządzane ad hoc) procedury

• niska dojrzałość organizacyjna wytwórcy.

• Dla zmniejszenia ryzyka personel ZJO powinien być zaangażowany w projekt programistyczny jak najwcześniej.

• Powinien on sprawdzać wymagania użytkownika, plany, procedury i dokumenty na zgodność ze standardami i przyjętymi procedurami postępowania.

• Wynika to z faktu, że dodatkowe koszty związane z problemem lub błędem są tym większe, im później zostanie on zidentyfikowany.

Zadania zapewnienia jakości:

Firma

– ciągła pielęgnacja procesu wytwarzania

– definiowanie standardów

– nadzór i zatwierdzanie procesu wytwarzania

Projekt

– dostosowywanie standardów

– przeglądy projektu

– testowanie i udział w inspekcjach

– ocena planów wytwarzania i jakościowych

– audyt systemu zarządzania konfiguracją

– udział w komitecie sterującym projektu

Tworzenie technologii

– tworzenie standardu

– wdrażanie standardu

Kontrola jakości

– ocena produktu

– ocena procesu

– zatwierdzanie jakości

Analiza działalności firmy

– zbieranie danych

– analiza danych

Administrowanie siecią komputerową

Zarządzanie personelem

Normy:

• Norma ISO 9001:2000 opiera się o 8 głównych zasad zarządzania jakością, które powinny być wykorzystane przez zarząd organizacji do poprawy jej funkcjonowania.

• Odpowiadają one praktycznym wymogom szybko zmieniającego się rynku, proponując proces ciągłego doskonalenia, a także kładąc szczególny nacisk na potrzeby i wymagana klienta w celu pozyskania jego satysfakcji z dostarczonego produktu.

1. Koncentracja na kliencie
2. Przywództwo
3. Zaangażowanie ludzi
4. Podejście procesowe
5. Podejście systemowe do zarządzania
6. Ciągłe doskonalenie
7. Podejmowanie decyzji na podstawie faktów
8. Wzajemnie korzystne powiązania z dostawcami

Jakość danych:

• Dane są wysokiej jakości jeżeli nadają się do użycia zgodnie z przeznaczeniem w zakresie działania, podejmowania decyzji i planowania. Dane nadają się do użycia zgodnie z przeznaczeniem, jeżeli nie zawierają defektów i posiadają pożądane cechy.

Cechy danych dobrej jakości:

• dostępność – możliwość wykorzystania; fakt występowania danych w postaci łatwej do przetworzenia, do których użytkownik ma dostęp oraz które są gotowe do wykorzystania nie później, niż z maksymalnie tolerowanym opóźnieniem,

• zrozumiałość – możliwość łatwej interpretacji; czytelna charakterystyka danych w celu ich prawidłowego użycia,

• spójność i poprawność – prawidłowość; przystawanie do rzeczywistych faktów, brak błędów i wzajemnych sprzeczności,

• kompletność – występowanie wszystkich elementów, które w rzeczywistym świecie odpowiadają występującemu zakresowi faktów,

• użyteczność – zgodność z oczekiwaniami; dopasowanie do potrzeb użytkownika i/lub realizowanego procesu biznesowego

Wang i Strong opracowali 15 wymiarów jakości danych z perspektywy użytkownika danych.

• Podzielili je na cztery kategorie:

• wewnętrzną,

• dostępu,

• kontekstu

• reprezentacji.

• Taki podział został przyjęty i zastosowany w wielu firmach i agendach rządowych a jego przydatność potwierdzona została w licznych badaniach.

• Zmierzenie jakości danych pod kątem wszystkich 15 wymiarów jest jednak przydatne jedynie w teorii.

Z praktycznego punktu widzenia nie ma sensu używać wszystkich wymiarów do oceny danych, a jedynie tych, które mają dla nas faktyczne jakieś znaczenie

|  |  |
| --- | --- |
| Kategoria | Wymiar |
| Wewnętrzna | dokładność, obiektywność, wiarygodność, reputacja |
| Dostępu | dostępność, bezpieczeństwo dostępu |
| Kontekstu | relewancja, wartość dodana, aktualność, kompletność, ilość danych |
| Reprezentacji | interpretowalność, łatwość zrozumienia, zwięzłość, spójna reprezentacja |

• Dokładność – zakres w jakim dane są poprawne i odpowiadają rzeczywistości

• Obiektywność – zakres w jakim dane są bezstronne i pozbawione tendencyjności

• Wiarygodność – zakres w jakim dane postrzegane są jako prawdziwe i poprawne

• Reputacja – zakres w jakim dane posiadają wysokie uznanie pod względem źródła lub zawartości

• Dostępność – zakres w jakim dane są dostępne lub łatwe do uzyskania

• Bezpieczeństwo dostępu – zakres w jakim dostęp do danych został ograniczony aby zapewnić ich bezpieczeństwo

• Relewantność – zakres w jakim informacje zawarte w danych nadają się do stawianych im zadań

• Wartość dodana – zakres w jakim wykorzystanie danych przyniesie wymierne korzyści

• Aktualność – zakres w jakim dane są aktualne ze względu na potrzeby stawianych im celów

• Kompletność – zakres w jakim dane zawierają wszystkie wymagane informacje, zarówno ilościowo jak i jakościowo, potrzebne do stawianych im celów

• Ilość danych – zakres w jakim ilość danych wpływa na utrudnienie wykonania operacji na nich

• Interpretowalność – zakres w jakim dane są zapisane w odpowiednim języku, przy użyciu odpowiedniej symboliki i z zachowaniem odpowiednich jednostek

• Łatwość zrozumienia – zakres w jakim typowy użytkownik jest w stanie zrozumieć informacje zawarte w danych

• Zwięzłość – zakres w jakim dane nie zawierają nadmiarowych i zbędnych informacji oraz nie zajmują w sposób nieuzasadniony dużo miejsca

• Spójna reprezentacja – zakres w jakim dane przestawiane są w jednolity sposób

Przyczyny problemów w danych, można podzielić na:

• spowodowane czynnikiem ludzkim

• błędy użytkowników,

• wady aplikacji

• wielokrotne migracje i integracje danych,

• powstałe wskutek czynników zewnętrznych.

• awarie sprzętowe

• samoistne starzenie się danych

# 25. Metoda COCOMO szacowania kosztów projektów informatycznych

Pierwsza wersja powstała w 1981 i nosiła nazwę COCOMO-81. Aktualna wersja to COCOMO2. Autor metody doszedł do wniosku, że w celu oszacowania kosztów osobowych tj. liczby osobmiesięcy( liczba miesięcy potrzebnych do realizacji projektu przez jedną osobę), konieczne jest podanie w przybliżeniu, z ilu linijek kodu będzie się składać gotowy produkt. Dalej to oszacowanie trzeba uściślić w kontekście określonych rodzajów projektów. Aby metoda była skuteczna, rozmiar oprogramowania powinien być szacowany przynajmniej na kilkanaście tysięcy linii kodu.

Model ten jest modelem trójpoziomowym, który pozwala na szacowanie kosztów z wrastającą dokładnością wraz z postępem cyklu inżynierii oprogramowania. Bazuje na rozmiarze oprogramowania.

* **Poziom prototypowania (application composition model)**

Faza prototypowania, poprzedzająca systematyczne modelowanie i projektowanie systemu. Występuje prototypowanie, wykorzystanie istniejących komponentów, podstawą oszacowań są punkty obiektowe. Jest przeznaczony dla projektów wykorzystujących nowoczesne narzędzia do tworzenie (generowania) interfejsów użytkownika (czyli na przykład dla prototypów)

* **Poziom wczesnego projektowania (early-design model)**

Odpowiada wczesnym fazom modelowania i projektowania systemu. Jest to etap projektowania architektury, najbliższy oryginalnego COCOMO, podstawą oszacowań są punkty funkcyjne przeliczane na linie kodu.

* **Poziom post-architektoniczny (post-architecture model)**

Odpowiada późnym fazom cyklu rozwoju systemu. Dla etapu rozwoju i utrzymywania, najbardziej szczegółowy model COCOMO II, podstawą oszacowań jest rozmiar kodu .

Model COCOMO stosuje proste wzory matematyczne z szeregiem parametrów. W modelu tym koszt wyraża się jako funkcję miary kodu (miarą może być liczba linii lub liczba tzw. punktów funkcyjnych). COCOMO pozwala także szacować pracochłonność wyrażoną w w osobo-miesiącach i czas realizacji projektu. Podstawowym wzorem dla oszacowania kosztów jest: **nakład[osobomiesiące]=A\*Kb**, gdzie K to rozmiar kodu źródłowego mierzony w tysiącach linii. A i b, to stałe, które są zależne od klasy przedsięwzięcia (klasy: 1. przedsięwzięcie ograniczone (łatwe) – wykonywane przez stosunkowo mały zespół przy pomocy dobrze znanych metod, 2. przedsięwzięcie pół-oderwanych (pośrednie) – członkowie zespołu różnią się stopniem zaawansowania, niektóre aspekty dziedziny nie są dobrze znane, 3. przedsięwzięcie osadzonych (trudne)– realizacja systemu o bardzo złożonych wymaganiach).

Wady COCOMO:

* Liczba linii kodu znana jest dokładnie dopiero wtedy, gdy system jest napisany. Szacunki są zwykle obarczone bardzo poważnym błędem.
* Określenie ‘linii kodu źródłowego’ inaczej wygląda dla każdego języka programowania

Zalety:

* Pozostawia miejsce na subiektywne oceny
* Utrzymany na podstawie doświadczenia
* Łatwy i szybki do uzyskania wyników
* Ma dobrą dokumentację

# 26. Metody automatycznej identyfikacji

Automatyczna identyfikacja polega na samoczynnym zebraniu danych o obiekcie, a następnie wprowadzeniu ich do systemu komputerowego bez ingerencji człowieka. Istnieje wiele metod automatycznej identyfikacji, jednak najbardziej znanymi są

· RFID

· Optyczne rozpoznawanie znaków (ang. OCR)

· Technologie biometryczne: identyfikacja głosowa, daktyloskopia

· Technologie kart inteligentnych: karty pamięciowe, karty mikroprocesorowe

· Kody kreskowe

**RFID** Określenie identyfikacja radiowa, "RFID", ang. Radio Frequency Identification, jest używane w odniesieniu do różnych technik komunikacji radiowej między urządzeniem służącym tylko do odczytu danych lub do odczytu i zapisu danych, nazywanym czytnikiem a urządzeniem przechowującym unikalne dane, nazywanym transponderem. System RFID składa się z systemu komputerowego (z bazą danych) i dwóch rodzajów urządzeń identyfikacji radiowej: czytników i transponderów. System komputerowy obsługuje aplikacja, która steruje urządzeniami RFID.

Transponder jest przeznaczony do umocowania do obiektu, który ma być identyfikowany. Może przechowywać i po uaktywnieniu wysyłać tylko swój indywidualny niepowtarzalny numer identyfikacyjny (ID), albo także inne dane odnoszące się do samego transpondera (np. kod wytwórcy) lub dane odnoszące się do obiektu, z którym transponder jest skojarzony. W niektórych systemach te dodatkowe dane mogą być wielokrotnie modyfikowane. Czytnik jest urządzeniem, które komunikuje się z transponderami znajdującymi się w zasięgu jego komunikacji radiowej. Czytnik nadzoruje protokół transmisji, odczytuje informacje z transponderów, może polecać transponderom przechowywanie informacji.

Działanie systemów RFID polega na komunikacji w zakresie częstotliwości radiowych pomiędzy transponderem i czytnikiem.

Transponder

Transponder (znacznik, tag) RFID jest radiowym urządzeniem nadawczym lub nadawczo-odbiorczym, wysyłającym sygnał zawierający kodowane dane identyfikacyjne tylko w odpowiedzi na pobudzenie sygnałem radiowym o określonej częstotliwości. Każdy transponder jest zbudowany z układu scalonego i anteny umieszczonych na odpowiednim podłożu lub w odpowiedniej obudowie. Układ scalony transpondera zawiera procesor, pamięć i nadajnik radiowy. W obudowie transpondera mogą znajdować się również dodatkowe elementy, takie jak np. czujnik temperatury i bateria stanowiąca źródło zasilania. Transponder może przechowywać różne dane i wskutek pobudzenia przesyłać je, np. w najprostszym przypadku tylko wpisany na stałe numer seryjny.

Transpondery mogą być również klasyfikowane jako:

– bierne (passive), tj. takie, które uzyskują energię tylko z pola elektromagnetycznego wytwarzanego przez czytnik;

– aktywne (active), tj. takie, których nadajnik RF jest zasilany z baterii;

– półaktywne (semi-active), tj. takie, w których wbudowana bateria służy do ciągłego zasilania czujnika (np. temperatury) zintegrowanego z transponderem, ale nie jest wykorzystywana do zasilania nadajnika RF.

Klasyfikacja:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Źródło energii zasilania transpondera |  |
|  |  | Bateria (aktywny) | Pole RF czytnika (bierny) |
| Sposób komunikacji | Bierny (przesyłanie danych przez modulację pola RF) | Transpondery nazywane również półaktywnymi | Prawie wszystkie transpondery powszechnie stosowane |
|  | Aktywny (własny nadajnik) | Transpondery o dużym zasięgu | Nie istnieją |

Czytnik

Czytnik RFID jest radiowym urządzeniem nadawczo-odbiorczym. Jego nadajnik za pośrednictwem anteny emituje energię wykorzystywaną do uaktywniania transponderów, a w niektórych systemach również sygnały poleceń sterujących transponderami i/lub modyfikujących dane zapisane w pamięci transponderów. Jego odbiornik demoduluje i dekoduje dane nadawane przez transponder.

Czytnik odbierający sygnał transpondera dekoduje dane i może je przesyłać do komputera systemowego za pośrednictwem łącza kablowego (tylko czytniki stacjonarne) lub radiowego.

Podział:

-cz stacjonarne – w celu tworzenia stref odczytu np. w bramie magazynu w celu inwentaryzacji przyjmowanych i wydawanych produktów, są stale w trybie czuwania, aby wykryć każdy transponder, który znajdzie się w ich strefie. Czytnik stacjonarny wymaga dołączenia przewodów zasilania i połączenia z siecią komputerową. Jego anteny powinny być chronione przed wpływami środowiska (kurz, zawilgocenie, wibracje) i uszkodzeniami mechanicznymi.

-cz ruchome – zainstalowane np. na wózku widłowym. Czytniki tego rodzaju są zasilane z baterii pojazdu. Ich anteny są szczególnie narażone na uszkodzenia mechaniczne i środowiskowe

-cz noszone – zasilane z wbudowanej baterii i zwykle są zintegrowane z anteną. Często noszony czytnik RFID jest skojarzony z czytnikiem kodów kreskowych. Połączenie z komputerem systemu identyfikacji jest realizowane drogą radiową np. w radiowej sieci lokalnej (WLAN). W tym przypadku czytnik jest zabierany do miejsca, gdzie znajdują się oznakowane obiekty, jest uaktywniany tylko na czas pojedynczego odczytu. Zatem użytkownik czytnika noszonego w pełni kontroluje gdzie, kiedy i jak czytnik będzie wykorzystany. Tego rodzaju czytniki są narażone na uszkodzenia wskutek upadku

Wady i zalety

+Odczytywanie danych na odległość, nawet gdy transponder nie jest widoczny znacznie usprawnia identyfikację obiektów oznaczonych RFID

+Transpondery przechowują o wiele więcej informacji niż kody kreskowe i etykiety zajmując przy tym mniej miejsca

+Transpondery RFID są bardziej odporne na uszkodzenia

+Technologia RFID znajduje zastosowanie w niemalże każdej dziedzinie życia, pozwalając użytkownikom oszczędzić czas i pieniądze

+Znaczniki RFID mogą być odczytane znacznie szybciej niż kody kreskowe, a do tego nie jest istotne ich położenie względem czytnika

+Znaczniki są trudniejsze do zniszczenia lub usunięcia

-Brak jednolitego standardu protokołu RFID

-Dane mogą zostać przechwycone przez niepowołaną osobę, jeśli są to dane osobowe przechowywane np. w E-paszportach grozi to poważnymi konsekwencjami. To główny argument przeciwników RFID

-Nie zapewnia 100% pewności odczytu

**OPTYCZNE ROZPOZNAWANIE ZNAKÓW** OCR (ang. Optical Character Recognition) – zestaw technik lub oprogramowanie służące do rozpoznawania znaków i całych tekstów w pliku graficznym o postaci rastrowej. Zadaniem OCR jest zwykle rozpoznanie tekstu w zeskanowanym dokumencie (np. papierowym formularzu lub stronie książki).

Niegdyś termin oznaczał samo rozpoznawanie ciągów znaków, głównie drukowanych, które są łatwiejsze do rozpoznania, dziś również pisma odręcznego oraz cech formatowania, jak krój pisma, stopień pisma, interlinia (techniki służące do tego typu zaawansowanego rozpoznawania nazywane są terminem ICR), a nawet układów tabelarycznych, np. formularzy.

Techniki OCR wykorzystywane są m.in. przy digitalizacji zasobów bibliotek, a także jako ułatwienie przy odczytywaniu danych z formularzy wypełnianych pismem odręcznym. W obu przypadkach oprogramowanie OCR nie jest tak skuteczne jak człowiek, zatem w przypadkach wątpliwych (trudności z odczytaniem fragmentu) oraz w celu uniknięcia błędów nieodzowna jest weryfikacja wyniku OCR przez człowieka.

Nowoczesną metodą współpracy oprogramowania OCR z ludźmi jest technika reCAPTCHA. Nie wymaga ona zatrudniania osób specjalnie do weryfikacji OCR, lecz wykorzystuje rozproszoną aktywność milionów użytkowników Internetu, którzy wchodząc na strony internetowe zabezpieczone przez CAPTCHA rozpoznają fragmenty tekstu wymagające weryfikacji.

Zasada działania

Rozpoznawanie pisma jest możliwe dzięki zastosowaniu metod z dziedziny rozpoznawania wzorców zaliczanej do sztucznej inteligencji. Oprogramowanie OCR wykorzystuje różne metody segmentacji obrazu, np. progowanie, aby wyodrębnić poszczególne znaki z obrazu, które następnie są najczęściej osobno klasyfikowane jako poszczególne litery. Zwykle w tym procesie wykorzystywane są sieci neuronowe. Zazwyczaj, by wyeliminować pomyłki, program sprawdza całość rozpoznanego tekstu lub poszczególne wyrazy pod kątem poprawności ortograficznej i gramatycznej danego języka.

**TECHNOLOGIE BIOMETRYCZNE** Opierają się na metodach identyfikacji osób na podstawie indywidualnych cech anatomicznych lub behawioralnych takich jak: linie papilarne, tęczówka i siatkówka oka, kształt twarzy, głos oraz charakter pisma. Rozwiązania wykorzystujące technologie biometryczne są wykorzystywane głównie w systemach wymagających dużego bezpieczeństwa.

Biometryczne cechy fizyczne:

**Tęczówka oka** uznawana obecnie za najdoskonalszy identyfikator człowieka. Zostaje ona ukształtowana już pod koniec 15 tygodnia życia płodowego i pozostaje dalej w niezmienionej postaci (pomijając uszkodzenia mechaniczne i poważne choroby). Jej dodatkową zaletą jest fakt, że jest chroniona przez inne narządy przed ewentualnymi uszkodzeniami. Potwierdzeniem posiadania przez nią wielu indywidualnych cech jest fakt, iż do tej pory nie znaleziono dwóch identycznych tęczówek – pozwala ona na rozróżnienie nawet bliźniąt jednojajowych. Systemy biometryczne dokonujące identyfikacji na podstawie tęczówki oka porównują badaną tęczówkę z zapamiętanym wcześniej wzorcem. Zdjęcie tęczówki wykonywane jest w wysokiej rozdzielczości i na jego podstawie określane są cechy charakterystyczne każdej tęczówki. Okulary i soczewki kontaktowe, a także przypadkowe mrugnięcia czy ruchy głowy nie przeszkadzają podczas przeprowadzania identyfikacji.

**Linie papilarne**, znajdujące się na palcach (maleńkie bruzdy, których zupełnie przypadkowy układ jest wynikiem marszczenia się skóry w czasie rozwoju płodowego), mają zastosowanie w roli identyfikatora człowieka już od dawnych lat, jednak szersze zastosowanie znalazły dopiero w dobie informatyzacji. Jak do tej pory nie znaleziono dwóch identycznych wzorców linii papilarnych (nawet u bliźniaków jednojajowych), co uzasadnia wykorzystanie ich w roli identyfikatora. Największą wadą jest to, że linie papilarne nie mają osłony przed ewentualnym ich uszkodzeniem. Drobne rany zostają zregenerowane, ale poważniejsze mogą spowodować powstanie trwałych zmian we wzorze. Do odczytania służą czytniki linii papilarnych, do których przykłada się palec. System biometryczny porównuje sprawdzane linie z zapamiętanym wcześniej wzorcem. Weryfikacja następuje w samym czytniku lub na serwerze z bazą danych zapamiętanych wzorców.

**Geometria twarzy**. Twarz każdego człowieka ma cechy charakterystyczne, które mogą być wykorzystane do identyfikacji. Elementem pobierającym próbki jest aparat fotograficzny lub kamera, a z uzyskanego w ten sposób obrazu tworzony jest wzór matematyczny opisujący geometrię badanej twarzy. Dąży się do tego, aby w procesie tworzenia wzorca wybrać cechy niezmienne (np. wielkość oczu) tak, aby móc go wykorzystywać w zmieniających się warunkach (np. zmiana uczesania).

**Rozkład temperatury na twarzy.** Dopiero kilka lat temu odkryto, że u każdego z nas układ temperatur na twarzy jest niepowtarzalny. Można go dojrzeć dzięki specjalnej kamerze termowizyjnej wychwytującej zmiany temperatury w dalekiej podczerwieni. W przeciwieństwie do skanowania tęczówki nie jest to metoda inwazyjna. Można też identyfikować osobę w całkowitych ciemnościach. Są też jednak wady – temperatura twarzy zależy częściowo od temperatury na zewnątrz, trudniej też ocenić tożsamość osoby w okularach, bo odbijają promieniowanie. Jednak Amerykanie, którzy prowadzą badania w tym zakresie, identyfikują osoby już z dokładnością 99%.

**Geometria dłoni**. Dłoń ludzka ma wiele indywidualnych cech. Podczas pomiaru umieszcza się ją w specjalnym czytniku, który wykonuje trójwymiarowe zdjęcie, rejestrując długość, szerokość, grubość czterech palców (bez kciuka) oraz wielkość obszarów pomiędzy kostkami. Łącznie dokonywanych jest ponad 90 pomiarów różnych cech charakterystycznych. Wynik tych pomiarów jest przechowywany w pamięci urządzenia w formie 9 bajtowego wzorca (co w teorii daje ponad 1024 możliwych kombinacji). Identyfikator ten jest praktycznie unikatowy dla każdego człowieka. Co więcej nowoczesne systemy biometryczne wykorzystujące geometrię dłoni uwzględniają także jej zmiany spowodowane upływem czasu (np. starzenie).

Inne cechy fizyczne: DNA, zapach, kształt ust, kształt uszu, barwa głosu, siatkówka oka (dno oka), kształt linii zgięcia wnętrza dłoni, układ naczyń krwionośnych na dłoni lub przegubie ręki.

Biometryczne cechy behawioralne:

**Weryfikacja głosu** polega na nagraniu przez sprawdzaną osobę próbki jej głosu, wyszczególnieniu w niej cech charakterystycznych oraz porównaniu ich z wcześniej wprowadzonym wzorcem. Na opis charakterystycznych cech głosu składa się m.in. brzmienie oraz szybkość wymawiania wyrazów czy sposób stawiania akcentu. Identyfikacja z wykorzystaniem głosu jest najczęściej uzupełniana innymi metodami identyfikacji (np. badaniem geometrii twarzy).

**Sposób wykonywania odręcznego podpisu**. Biometria podpisu odręcznego tym różni się od innych biometrii, że nie polega na mierzeniu określonych cech wrodzonych, a pewnego wyuczonego zachowania. Podczas badania podpisu wykorzystuje się nie tylko charakterystykę wizualną (dwuwymiarowy obraz), ale także sposób, w jaki podpis został złożony, tj. dynamikę ruchu pióra. Podpisy są pobierane przy pomocy tabletu graficznego. Każdy pobrany podpis jest reprezentowany przez sekwencje wektorów, z których każdy zawiera współrzędne końcówki pióra, nacisk oraz kąty: azymut i nachylenie. Liczba wymiaru wektora zależy od liczby cech, jakie można pobrać za pomocą konkretnego tabletu.

Inne cechy behawioralne: sposób chodzenia, sposób pisania na klawiaturze, sposób reakcji mózgu – fala P300 – na pewne znane informacje-bodźce (fala P300 jest reakcją mózgu powstającą w czasie 300 msek od bodźca; jest ona charakterystyczna dla każdego człowieka).

**KARTY INTELIGENTNE** to małe plastikowe karty z mikroukładem, umożliwiającym przechowywanie różnego rodzaju informacji (klucze prywatne, hasła itp.). Karty inteligentne wykorzystywane są w systemach PKI, w bankowości oraz w innych instytucjach i firmach IT.

Główną zaletą kart inteligentnych jest ochrona przechowywanych danych przed niepowołanym dostępem oraz modyfikacją. Wykorzystanie kart inteligentnych pozwala budować systemy łatwe w użyciu, bezpieczne i tanie. Dlatego rynek ten sięga setek milionów rocznie wyprodukowanych kart i ciągle rośnie. Największa wadą kart jest ich podatność na uszkodzenia. Dotyczy to głównie układu scalonego, a także styków. Układ scalony można uszkodzić wyginając kartę, a styki ulegają wytarciu, zabrudzeniu lub zardzewieniu.

Wyróżniamy:

-Karty tłoczone

-Karty magnetyczne

-Karty elektroniczne (inteligentne, smart card)

--Pamięciowe, mikroprocesorowe

--Stykowe, bezstykowe (contactless), hybrydowe

Obszary zastosowań:

-Transakcje płatnicze

-Programy lojalnościowe

-Zapis danych biometrycznych (elektroniczny dowód osobisty)

-Infrastruktura klucza publicznego (podpis cyfrowy)

-System GSM

-System dostępu

-Systemy transportu publicznego (elektroniczny bilet)

-Telefonia komórkowa (karta SIM)

Ze względu na budowę układu scalonego możemy wyróżnić dwa typy kart inteligentnych:

K pamięciowe – są to karty elektroniczne posiadające pewien obszar pamięci służący do przechowywania danych. Zapis i odczyt odnosi się do struktury fizycznej karty. Przy czym sam sposób komunikacji, zapisu/odczytu z/do pamięci nie jest ustandaryzowany. Wyposażone są zazwyczaj w pamięć typu EEPROM bądź ROM. Czytnik kart pamięciowych odczytuje zapisane dane, traktując kartę jako maszynę stanów. Dodatkowo można zastosować metody kryptologiczne w celu zwiększenia bezpieczeństwa danych przechowywanych w pamięci. Największą zaletą tego rodzaju kart jest ich stosunkowo niska cena. Przykłady: k telefoniczna.

K procesorowe – wyposażone są w system operacyjny. Nazywane są też kartami mikroprocesorowymi. Zaopatrzone są w mikroprocesor podłączony do pamięci typu ROM, RAM i EEPROM. Pamięć ROM zawiera system operacyjny wgrywany podczas produkcji karty. Jest on identyczny dla wszystkich kart danego typu i nie może podlegać zmianom. Pamięć EEPROM wypełniana jest kodem oraz danymi aplikacji. Zapis danych w tym obszarze kontrolowany jest przez system operacyjny. Pamięć RAM pełni natomiast rolę pamięci roboczej. W pamięci pojedynczej kart mikroprocesorowej można umieścić dowolną aplikację (ograniczeniem jest tylko pojemność), przez co zastosowania tego typu kart są ograniczone tylko przez twórczą inwencję projektantów. Przykłady: SIM, VITAY

**KODY KRESKOWE** Kod kreskowy jest kombinacją jasnych i ciemnych kresek, ułożonych równolegle do siebie, o zróżnicowanych szerokościach, odzwierciedlającą ciąg ściśle określonych znaków. Występują kody jednowymiarowe – liniowe, dwuwymiarowe – piętrowe i matrycowe oraz kody hybrydowe. Symbolem kodu kreskowego nazywa się kompletny kod kreskowy składający się z jasnego marginesu poprzedzającego kod, znaku startu, znaków odwzorowujących dane, znaku stopu i ponownie jasnego marginesu. W skład symbolu wchodzą również znaki pisma, które zostały odzwierciedlone tym kodem kreskowym, umieszczone bezpośrednio nad lub pod kodem.

Zastosowanie m.in.

-W magazynach – do przyspieszenia przygotowywania remanentu i innych operacji magazynowych, np. przyjmowania i wydawania przesyłek, sortowania, konfekcjonowania, zarządzania magazynem oraz spisem z natury

-W zakładach produkcyjnych – do rejestrowania przepływu surowców i półproduktów w procesie produkcyjnym

-W firmach wysyłkowych - do oznaczania wysłanych paczek

-W firmach kurierskich – do oznaczania przesyłek

-W dużych zakładach – do identyfikowania środków trwałych i szybkiego sporządzania spisów, do sporządzania kart klienta, kart rabatowych, kart identyfikacyjnych

-W przedsiębiorstwach spedycyjnych, logistycznych, kurierskich i na poczcie – śledzenie przesyłek

-W sklepach – odczytywanie cen przesuwając produkty z kodami nad czytnikiem

-Znakowanie rachunków, listów, kopert itp.

Podział kodów:

· Linearne - symboliki tego rodzaju składają się z pionowych linii różnej szerokości, rozdzielonych białymi przestrzeniami. Przykłady: Code 128, Code 39, EAN-13, EAN-8, ITF, UPC-A, UPC-E

· Dwuwymiarowe: mają one dużo większą pojemność danych, jeden kod dwuwymiarowy może pomieścić do 3,5 tys. znaków, gdy kod liniowy do 50 znaków, dzielimy na *macierzowe* – wyglądają jak macierze punktów, przykłady: Aztec, Data Martix, MaxiCode, QRCode, *wielowierszowe liniowe (piętrowe)* – linearne kody kreskowe z bardzo krótkimi paskami ułożone jeden na drugim, przykłady: Codablock F, Code 49, PDF417, SuperCode.

· Trójwymiarowe (wypukłe) - jest to tak naprawdę kod liniowy, tylko zamiast czarnych i białych linii wykorzystuje wgłębienia i wypukłości

**KODY KRESKOWE A RFID**

Technologia RFID ma wiele zalet, których brakuje kodom kreskowym, między innymi:

-Możliwość zmiany danych zawartych w znaczniku,

-Znacznik nie musi być widoczny w trakcie odczytu,

-Większy zasięg odczytu (zależnie od rodzaju znacznika),

-Możliwość zapisania większej ilości danych,

-Możliwość równoczesnego odczytu wielu znaczników,

-Większa odporność na zabrudzenie i uszkodzenia,

-Możliwość innych zastosowań poza przechowywaniem danych.

Mimo tak wielu zalet znaczników RFID, kody kreskowe prawdopodobnie nie wyjdą nigdy z użycia, z takich przyczyn jak:

-Niższy koszt jednostkowy,

-Zbliżona a nawet wyższa dokładność odczytu,

-Neutralny wpływ materiału, na którym znajduje się kod kreskowy na jego odczyt,

-Brak międzynarodowych ograniczeń związanych z częstotliwościami pracy urządzeń RFID,

-Brak zastrzeżeń ze strony obrońców prywatności,

-Dojrzała i powszechnie używana technologia.

# 27. Metody szacowania wielkości projektów informatycznych

# 28. Mobilność w systemach informacyjnych

# 29. Modelowanie funkcji systemu informacyjnego - techniki modelowania, hierarchie funkcji, korzyści z modelowania funkcji i tworzenia hierarchii funkcji

# 30. Modelowanie struktur wymiany danych

# 31. Narzędzia integracji systemów informacyjnych

EDI, CORBA, DCOM, RPC, RMI, Web Service, ESB

**EDI (Electronic Data Interchange)** – ustandaryzowana metoda wymiany danych pomiędzy komputerami bądź sieciami komputerowymi, z wykorzystaniem standardowych, zaakceptowanych formatów komunikatów. EDI to sposób wymiany informacji pomiędzy kontrahentami, charakteryzujący się przesyłaniem dokumentów wyłącznie w formie elektronicznej („bezpapierowej”), wymianą dokumentów spełniających uzgodnione wcześniej standardy (definicja pól dokumentu i ich zawartości), wymianą danych bezpośrednio pomiędzy systemami informatycznymi partnerów handlowych; dane będące przedmiotem elektronicznej wymiany są odpowiednikami tradycyjnych dokumentów handlowych takich jak: faktura, zamówienie, awizo dostawy, harmonogram produkcji itp. Kluczowym aspektem EDI jest standaryzacja przesyłanych informacji oraz bezpieczeństwo komunikacji.

Celem EDI jest wyeliminowanie wielokrotnego wprowadzania danych oraz przyspieszenie i zwiększenie dokładności przepływu informacji dzięki połączeniu odpowiednich aplikacji komputerowych między firmami uczestniczącymi w wymianie. Użycie EDI pozwala poprawić czasową dostępność informacji logistycznej, poszerzyć i uściślić dane, a także zmniejszyć pracochłonność procesu. Aby w pełni wykorzystać zalety EDI, uczestnicy kanału logistycznego powinni się komunikować za pośrednictwem komputera. Innymi słowy, efektywne wdrożenie EDI wymaga bezpośredniej komunikacji między systemami komputerowymi, zarówno nabywców jak i sprzedawców produktu.

EDI nie określa sposobu przesyłania komunikatów – mogą one być przesyłane przez dowolne medium, którym posługują się obie strony transmisji. Może to być transmisja modemowa, poprzez FTP, HTTP, AS1, AS2.

Wady i zalety:

+Eliminujemy błędy klawiatury,

+Automatycznie wielokrotnie wykorzystuje wprowadzane dane,

+Pozwala na szybką transmisję danych,

+Szybki czas wypełniania typowych formularzy,

+Eliminuje dokumentację papierową,

+Usprawnia księgowość,

+Pozwala przesunąć personel do innych zadań,

+Obniża koszty operacji administracyjnych

+systemy EDI nie wymagają skomplikowanego sprzętu, korzystają z sieci transmisji danych

-Poziom skomplikowania komunikatów EDI

-Mnogość standardów oraz wersji katalogów i komunikatów

-Problemy z utrzymywaniem i aktualizacją standardów

-Wysokie koszty, trudny i długotrwały proces wdrażania

-Brak pełnego standardu wymiany informacji

-Często zbyt kosztowne dla małych i średnich przedsiębiorstw

**CORBA (Common Object Request Broker Architecture)** – technologia zapewniająca komunikację pomiędzy obiektami pracującymi w heterogenicznych (różnorodnych) systemach komputerowych. Obiekty pełniące dowolne funkcje mogą być zaimplementowane w różnych językach programowania, na dowolnej platformie sprzętowej, pod kontrolą różnych systemów operacyjnych.

Opis obiektów, a właściwie interfejsów do nich, znajduje się w pliku IDL (ang. Interface Definition Language), który jest kompilowany na kod zajmujący się przekazywaniem metod (w przypadku implementacji technologii CORBA w niektórych językach interpretowanych, plik IDL jest interpretowany w czasie wykonania).

Obiekty mają swoje adresy IOR (ang. Interoperable Object Reference). Są to kilkusetznakowe adresy kodujące wiele informacji o obiekcie, m.in. adres komputera, adres programu na komputerze, informacje o kolejności zapisu bajtów (czy jest to big endian, czy little endian), numer obiektu, typ obiektu, itd.

Adresy IOR mogą dotyczyć także niskopoziomowych protokołów transmisji danych – zwykle GIOP (ang. General Inter-ORB Protocol) lub IIOP (ang. Internet Inter-ORB Protocol).

Standardy tworzone są według zasady "najpierw standard, potem implementacje" (czyli tak samo jak w W3C i zupełnie inaczej niż w IETF).

Jedną z wad architektury CORBA jest brak standardowego i szeroko zaimplementowanego mechanizmu bezpieczeństwa.

Popularne implementacje standardu CORBA to np. ORBit, OMNIOrb.

Wady i zalety:

+Architektura zapewnia bardzo wygodny sposób zapewniania współpracy różnorodnych systemów

+Daje możliwość zamaskowania systemów odziedziczonych w pełni wykorzystując ich możliwości

+Zapewnia dostęp do obiektów niezależnie od platformy implementacji obiektu dzięki wykorzystaniu języka opisu styków IDL zamiast języka programowania, co jest podstawowym wymaganiem przy współpracy systemów dostarczonych przez różnych dostawców

+Otwarty standard

+Obsługa wielu platform

+Obsługa wielu języków

+Skalowalność

-Duże koszty uruchomienia

-Słaba wydajność, mało efektywne działania w czasie rzeczywistym

-Zapory (?)

-Brak standardowego i szeroko zaimplementowanego mechanizmu bezpieczeństwa

**DCOM (Distributed Component Object Model)** – interfejs programistyczny realizujący rozproszony obiektowy model składników. Jest opatentowaną technologią firmy Microsoft służącą do budowania składników programowych i zapewniania komunikacji między nimi w małej sieci komputerowej. Skonstruowany został również zestaw bazowy takich składników.

Rozwinięty z COM jako odpowiedź na CORBA, stał się później częścią COM+. Został przez Microsoft uznany za przestarzały na rzecz platformy .NET Framework.

Dodanie litery "D" do COM spowodowane było szerokim wykorzystaniem technologii DCE/RPC (Distributed Computing Environment / Remote Procedure Call), a dokładniej udoskonalonej przez Microsoft wersji, znanej jako MSRPC.

*COM - Standard definiowania i tworzenia interfejsów programistycznych na poziomie binarnym dla komponentów oprogramowania wprowadzony przez firmę Microsoft wraz z bibliotekami zapewniającymi podstawowe ramy i usługi dla współdziałania komponentów COM i aplikacji. Nowe środowisko programistyczne Visual Studio oparte na frameworku .NET umożliwia komunikację ze starszymi aplikacjami z interfejsem typu COM. Na bazie standardu COM została zdefiniowana znaczna część niskopoziomowego API dla produktów firmy Microsoft m.in. dla DirectX, SQL Server, MS Access, MSHTML, MSXML.*

Stosownie do tego określenia, DCOM miał rozwiązać problemy:

-Serializacja argumentów i wartości zwracanych wywoływanych metod, oraz reprezentację ich podczas połączenia.

-Rozproszone zbieranie nieużytków - zapewnienie, że połączenie nawiązane przez program klienta zostanie zwolnione, gdy na przykład proces klienta ulegnie awarii lub połączenie sieciowe zostanie przerwane.

Jednym z kluczowych czynników w rozwiązywaniu tych problemów jest zastosowanie DCE / RPC jako mechanizmu RPC wykorzystanego w DCOM. DCE / RPC ma ściśle określone zasady dotyczące serializacji i kto jest odpowiedzialny za zwolnienie pamięci.

Wady i zalety:

+Wsparcie

+HYPED MARKET( ?)

-Wsteczna kompatybilność

-Interoperacyjność (zdolność sieci telekomunikacyjnych do efektywnej współpracy w celu zapewnienia wzajemnego dostępu użytkowników do usług świadczonych w tych sieciach) (?)

**RPC (Remote Procedure Call)** – zdalne wywołanie procedury, to protokół zdalnego wywoływania procedur, stworzony przez firmę Sun i swego czasu dość popularny na Uniksach, obsługiwany w bibliotekach języka Java. Współcześnie wypierany przez bardziej rozbudowane protokoły takie jak CORBA, XML-RPC, czy JSON-RPC.

RPC zdefiniowany jest w RFC 1057. RPC używa do kodowania danych formatu XDR (eXternal Data Representation) zdefiniowanego w RFC 1832.

Protokoły tego typu (jak RPC, CORBA, DCOM, czy XML-RPC) mają na celu ułatwienie komunikacji pomiędzy komputerami. Na typowy scenariusz użytkowania składają się:

-Serwer (czyli program oferujący usługi, np. drukowania) przez cały czas nasłuchuje na wybranym porcie, czy ktoś się z nim nie łączy.

-Klient (czyli program który potrzebuje jakiejś usługi od serwera na innym komputerze) nawiązuje z nim łączność poprzez sieć komputerową.

Klient wysyła swoje dane we wcześniej ustalonym przez programistów klienta i serwera formacie. Serwer realizuje usługę i odsyła potwierdzenie lub kod błędu. Powyższe protokoły same zapewniają cały powyższy mechanizm, ukrywając go przed klientem. Może on nawet nie wiedzieć, że łączy się z innym komputerem - z punktu widzenia programisty zdalne wywołanie procedury serwera wygląda jak wywołanie dowolnej innej procedury z programu klienta.

Wady i zalety:

+prosty interfejs

+popularny język programowania (proceduralny)

+obsługuje modułowe i hierarchiczne modele systemów rozproszonych

+Łatwość pisania przenośnych programów rozproszonych / sieciowych. Implementacja komunikacji client–server sprowadza się do prostego wywoływania odpowiednich funkcji.

-Problemy z przekazywaniem argumentów do funkcji (np. przekazywanie wskaźników), procedury nie mogą korzystać ze zmiennych globalnych

-Różne systemy, języki, kompilatory, architektury sprzętowe wymuszają określony format wbudowanych typów danych (zakres liczb, kolejność bajtów), a także wprowadzają różnorakie ograniczenia na rzutowanie jednego typu na drugi itp.

-Trudno zrealizować zdalne funkcje o zmiennej liczbie argumentów, oraz takie w których zmieniają się typy tych argumentów (np. standardowa funkcja printf).

-Odpowiednie zabezpieczenie systemu obsługującego RPC staje się wręcz krytyczne.

-Trudno kwestią jest zagwarantowanie wywołania procedury na serwerze, sprawdzenie czy procedura została rzeczywiście wykonana oraz czy została wykonana tylko raz.

-Technologia RPC generuje spore narzuty.

**RMI (Remote Method Invocation)** – zdalne wywołanie metod, to mechanizm umożliwiający zdalne wywołanie metod obiektów. Obiekty te mogą znajdować się w innych maszynach wirtualnych Javy, które mogą znajdować się na innych komputerach.

RMI dostarcza zestaw narzędzi umożliwiających pisanie programów rozproszonych przy użyciu takiej samej syntaktyki i semantyki jak dla programów pracujących na jednym komputerze (jednej maszynie javy, JVM).

RMI oddziela definicję zachowania (interfejs) od jego implementacji i pozwala na uruchamianie implementacji na oddzielnej maszynie javy. Odpowiada to programowaniu typu klient-serwer, gdzie klient wykorzystuje zdefiniowany interfejs, a serwer dostarcza jego implementację. RMI dba o szczegóły dotyczące protokołu, przekazywania parametrów zwracania wyników.

Interfejs javy nie zawiera wykonywalnego kodu, więc RMI używa dwóch klas implementujących ten sam interfejs. Pierwsza klasa implementuje działanie (*service implementation*) i jest uruchamiana na serwerze. Druga klasa funkcjonuje jako pośrednik (*service proxy*) dla wywołań klienta. Program klienta wywołuje metody w klasie pośredniczącej, a ta przekazuje je do zdalnego serwera oraz pobiera zawracane wyniki.

Implementacja RMI oparta jest na trzech warstwach abstrakcji. Pierwsza to warstwa szkieletów i trzonów (*stubs and skeletons*). Warstwa ta jest bezpośrednio używana przez programistę. Przyjmuje ona wywołania metod skierowane do interfejsu i przekazuje do zdalnych wywołań RMI. Kolejną warstwą jest warstwa referencji (*remote reference layer*). Warstwa ta odpowiada za interpretację i zarządzanie referencjami do obiektów przekazywanych między klientem, a serwerem. Dokonuje automatycznej serializalizacji oraz deserializacji obiektów lub przekazuje zdalne referencje do nich. Warstwa transportowa (*transport layer*) bazuje na protokole TCP/IP i dostarcza podstawowego połączenia między systemami. Może być zastąpiona implementacją bazującą np. na protokole UDP bez zmiany wyższych warstw.

Warstwa szkieletów i trzonów

Szkielet jest klasą pomocniczą generowaną przez RMI, która potrafi komunikować się z trzonem poprzez łącze RMI. Szkielet jest klasą działającą na serwerze. Odbiera parametry z połączenia RMI, przekazuje je do wywoływanej klasy serwera. Otrzymany wynik przekazuje z powrotem do klienta poprzez łącze RMI. Od wersji 2 JDK klasy szkieletów nie są wymagane. Trzon łączy się bezpośrednio z obiektami na zdalnym serwerze. Trzon jest klasą generowaną przez RMI, służy jako pośrednik między programem klienta, a RMI. Przekazuje parametry wywołania do łącza RMI, odbiera wynik i zwraca do programu klienta.

Warstwa zdalnych referencji

Warstwa to dostarcza obiekt klasy RemoteRef, który reprezentuje łącze do zdalnej implementacji. Trzon wykorzystuje metodę invoke() do przekazywania wywołań do serwera.

W wersji JDK 1.1 znajduje się tylko jedna metoda zdalnego wywoływania: unicast. Zanim klient wywoła zdalną metodę, zdalna usługa musi zostać utworzona na serwerze i wyeksportowana do mechanizmu RMI.

JDK 1.2 dostarcza nowa metoda zdalnego wywoływania: za pomocą aktywowania. RMI samo zajmuje się tworzeniem zdalnych obiektów oraz odtwarzaniem ich stanu z dysku.

Możliwe jest dodawanie nowych rodzajów wywołań, np. skierowanych do wielu redundantnych serwerów usług.

Warstwa transportowa

Tworzy połączenia między wirtualnymi maszynami javy. Wszystkie połączenia są oparte na strumieniach i używają TCP/IP, nawet jeśli obie JVM są uruchomione na tej samej maszynie.

RMI używa protokołu JRMP (Java Remote Method Protocol) opartego na bazie TCP/IP. Jest to strumieniowy protokół występujący w dwóch wersjach. Pierwsza wersja użyta w JDK 1.1 wymaga klas szkieletów na serwerze. Druga wersja pojawiła się wraz z JDK 2, zastała zoptymalizowana oraz nie wymaga klas szkieletów. Istnieją implementacje RMI, które nie używają JRMP (np. BEA Weblogic).

Wady i zalety:

+Prosty w implementacji i używaniu, która prowadzi do bardziej wytrzymałych, elastycznych aplikacji i lepszego ich utrzymania

+Zapewnia niezależność programów od procesora (wystarczy działająca maszyna Java VM).

+Troszczy się o szczegóły przesyłania obiektów, ich serializację i deserializację

-Wsparcie tylko dla programów napisanych w javie, zarówno klient jak i serwer muszą być aplikacjami lub apletami javy i nie można komunikować się z programami napisanymi w żadnym innym języku programowania za pomocą tego mechanizmu

-Nie może używać kodu spoza zakresu Javy

-Zarówno klient jak i serwer potrzebują dostępu do najświeższych identycznych definicji klas obiektów

**WEBSERWIS** To metoda komunikacji dwóch elektronicznych urządzeń poprzez WWW. WS jest systemem utworzonym w celu wspierania interoperacyjnej interakcji komputer-komputer poprzez sieć. Posiada interfejs opisany w formacie przetwarzanym maszynowo (WSDL). Inne systemy współdziałają z webserwisem w sposób opisany w jego opisie (WSDL) przy pomocy wiadomości SOAP, przeważnie przekazywanej przy użyciu protokołu http. Webserwisy zwykle nie tworzą samodzielnych aplikacji, a jedynie stanowią interfejs komunikacyjny pomiędzy klientami z sieci a systemami biznesowymi.

Wyjątkową siłą webservices jest wykorzystanie istniejących i szeroko stosowanych technologii tj. protokołu HTTP i języka XML. HTTP jest jednym z najbardziej rozpowszechnionych protokołów w sieci WEB, co umożliwia natychmiastowe wykorzystanie tej platformy do przesyłania komunikatów. XML dostarcza metajęzyk za pomocą, którego porozumiewają się klienci z usługami oraz poszczególne komponenty. Idąc dalej, przy budowie i wykorzystaniu webservices nie ma znaczenia w jakiej technologii jest napisana usługa, na jakim systemie operacyjnym pracuje. Pełną interoperatywność zapewnia protokół SOAP, który ma aspiracje stać się następcą technik typu CORBA, RMI czy DCOM.

Dzięki wykorzystywaniu nowego protokołu SOAP usługa opracowana w technologii Web Services ma możliwość współpracy (wymiany komunikatów) z dowolną inną usługą. Powinny zniknąć problemy występujące przy konwersjach realizowanych pomiędzy protokołami architektur DCOM i CORBA. Web Services mogą być pisane w dowolnych językach programowania, więc twórcy aplikacji będą mogli pisać usługi bez zmiany środowiska tworzenia aplikacji, w którym wcześniej pracowali. Protokół SOAP jest obecnie wspierany przez wszystkich podstawowych dostawców systemów oraz oprogramowania narzędziowego.

**SOAP – Simple Object Access Protocol (Prosty Protokół Dostępu do Obiektów).**

*Czym jest SOAP:*

· Jest protokołem, a zatem mechanizmem transportu informacji;

· Przenoszone informacje są uporządkowane (posiadają strukturę i typy);

· Dane zapisane są w języku rozszerzalnych znacznikÛw (XML);

· W tym protokole można przenosić wszelkiego rodzaju dane (w razie potrzeby są one kodowane do postaci wyrażalnej w XML);

· SOAP posiada bardzo szeroki zakres zastosowań: od prostych zastosowań komunikacyjnych po zdalne wywoływanie procedur (RPC).

*Czym nie jest SOAP:*

· Nie jest tym samym, co XML, chociaż jest na nim oparty;

· Nie ma związku z protokołami SMTP czy HTTP, chociaż komunikaty SOAP często są przesyłane z wykorzystaniem mechanizmów typowych dla tych protokołów (np. transmitowane przez TCP/IP na portach 25 lub 80, często spotyka się również enkapsulowanie wewnątrz komunikatów SMTP lub HTTP);

· Nie jest związany z żadnym konkretnym modelem programowania czy jakąś specyfiką implementacji - przeciwnie: jest uniwersalnym sposobem komunikacji w środowiskach rozproszonych oraz heterogenicznych

Model komunikacji

Komunikacja za pomocą SOAP jest zasadniczo komunikacją jednokierunkową: od nadawcy do odbiorcy. Łatwo sobie jednak wyobrazić inne scenariusze bazujące na tym podstawowym: np. schemat wywołanie ñ odpowiedź (request - response) lub dialog dwóch stron. Wybrana do przenoszenia komunikatów warstwa transportowa może ułatwiać budowanie takich bardziej rozbudowanych scenariuszy, np. przy łączeniu za pomocą TCP/IP po przesłaniu komunikatu nadawca z odbiorcą zamieniają się rolami, a odpowiedź jest transmitowana tym samym połączeniem, którym poszło wywołanie.

Struktura komunikatu SOAP

Komunikat SOAP jest dokumentem XML, który składa się z trzech elementów:

1. Koperty SOAP (SOAP envelope) – obowiązkowo: jest głównym, najbardziej zewnętrznym elementem dokumnetu XML, który reprezentuje komunikat. Nazwa tego elementu musi brzmieć „Envelope”, ten element musi być obecny w komunikacie SOAP, może zawierać deklaracje przestrzeni nazw oraz dodatkowe atrybuty.

2. Nagłówka SOAP (SOAP header) – niekoniecznie: służy do przekazywania informacji o tym kto i w jaki sposób powinien przetwarzać dane zawarte w ciele komunikatu. Zawartość nagłówka może nieść także informację dla ogniw pośredniczących w transporcie komunikatu (np. serwerów proxy), może być także przez nie zmieniana. Nazwa tego elementu musi brzmieć „Header”, jeśli ten element występuje w komunikacie SOAP, musi być pierwszym bezpośrednim potomkiem elementu „Envelope”.

3. Ciała SOAP (SOAP body) – obowiązkowo: w tym elemencie przenoszona jest właściwa informacja. Nazwa elementu musi brzmieć „Body”, element ten musi wystąpić w komunikacie i musi być bezpośrednim potomkiem elementu „Envelope”, jeśli w komunikacie występuje nagłówek, to ciało musi następować bezpośrednio po nim, w przeciwnym wypadku musi być pierwszym potomkiem elementu „Envelope”, ten element może zawierać serię wpisów, z których każdy jest jego bezpośrednim potomkiem.

Kodowanie danych w komunikatach SOAP

Kodowanie danych w SOAP wykorzystuje hierarchię typów danych, która jest generalizacją podobnych hierarchii spotykanych we współczesnych językach programowania czy bazach danych. Najbardziej podstawowy podział typów to podział na typy proste (skalarne) oraz na typy złożone, które są połączeniem pewnej liczby części, z których każda posiada swój typ. XML dopuszcza bardzo elastyczne kodowanie danych, SOAP w tej dziedzinie jest bardziej ograniczony (ma ściślejsze reguły kodowania).

Transmisja komunikatów SOAP protokołem HTTP

Bardzo często do transportu komunikatów SOAP używany jest protokół HTTP. Przyczyn jest kilka:

-jest to protokół bardzo elastyczny;

-jest zarazem pewny - posiada mechanizmy raportowania błędów;

-jest niezwykle rozpowszechniony (co wiąże się z bogactwem narzędzi);

-umożliwia przekazywanie komunikatów SOAP nawet przez zapory (firewall) analizujące zawartość transmitowanego strumienia danych (ponieważ zazwyczaj zapory takie przepuszczają transmisję HTTP).

Ze względu na charakter protokołu HTTP najbardziej naturalne jego wykorzystanie zachodzi podczas scenariusza transmisji wywołanie - odpowiedź (request - response). Podczas transmisji komunikatów SOAP wewnątrz wywołań/odpowiedzi HTTP należy używać typu „text/xml”. Pomimo iż SOAP można powiązać z różnymi typami wywołań HTTP, najczęściej jednak jest używane wywołanie POST.

Wywoływanie procedur (RPC)

Jednym z celów projektowych SOAP było umożliwienie transportowania za jego pomocą wywołań i odpowiedzi RPC. Choć nie jest to jedyne rozwiązanie, jednak naturalne staje się zastosowanie jako protokołu transportującego HTTP, gdzie wywołanie HTTP staje się wywołaniem zdalnej procedury, natomiast odpowiedź HTTP reprezentuje odpowiedź RPC. Reprezentacja RPC opisana w specyfikacji protokołu SOAP zakłada, że dane niezbędne do wywołania procedury (URI obiektu docelowego, nazwa metody, opcjonalna sygnatura metody i parametry metody) przekazywane są jako struktura (złożony typ danych) wewnątrz ciała komunikatu SOAP. Oczywiście nie jest to jedyna możliwość.

Wady i zalety

+niezwykła elastyczność protokołu, który pozwala przenosić właściwie dowolne informacje;

+możliwość definiowania struktury i semantyki przenoszonych informacji;

+możliwość łączenia z różnymi protokołami transportowymi (np. HTTP);

+możliwość realizacji różnych scenariuszy komunikacji;

+akceptowalność protokołu przez właściwie wszystkie systemy komputerowe i środowiska systemowe;

+niezawodność protokołu dzięki ścisłemu zdefiniowaniu sytuacji wystąpienia błędu oraz zachowania aplikacji w takich okolicznościach.

-duży narzut samego języka XML (rozmiar komunikatu jest znacząco większy niż sumaryczny rozmiar danych w nim zawartych);

-jest jeszcze dość młodym protokołem, podlega rozwojowi i modyfikacjom, chociaż jest już dość dobrze i ściśle zdefiniowany.

**WSDL (Web Services Description Language)** – prosty język pozwalający opisać usługę sieciową - jej sposób wywołania, parametry i formaty wyników, lecz bez żadnych informacji biznesowych czy parametrów dostępności. Dzięki WSDL mamy informacje jakie metody udostępnia Web Service. Usługę można odpytać, jakie funkcje udostępnia. WSDL umożliwia twórcom usługi Web opisanie w zrozumiałym formacie co potrafi usługa, gdzie się znajduje i jak ją wywołać. Język WSDL jest standardowym formatem opisu sieciowych usług XML. Język WSDL bazuje na standardzie XML. Jego zadaniem jest tworzenie opisu funkcji usług Web Services oraz sposobu ich wywoływania. Język WSDL jest przydatny przy automatyzacji komunikacji pomiędzy usługami Web Services, umożliwiając współdziałanie usług.

WSDL nie wprowadza nowego języka definicji typu. Rozpoznaje jedynie typy w opisywanym formacie wiadomości i wspiera specyfikację XML Schema. WSDL definiuje również wspólny binding mechanism. Jest on używany do dołączania właściwego protokołu lub formatu danych lub struktury do abstrakcyjnej wiadomości, operacji czy punktu końcowego.

Wady i zalety stosowania WSDL

+standardowy format opisu usług

+możliwość wykorzystania w wielu różnych technologiach i systemach komputerowych

+zrozumiały interfejs wykorzystywany przy definiowaniu usług

+łatwy w użyciu

+automatyzacja komunikacji między usługami

+szybsze udostępnianie i aktualizacja oprogramowania

-problemy z bezpieczeństwem (to dodatkowy protokół obsługi, który trzeba zabezpieczyć)

-problemy z autoryzacją

-niezbyt duża wydajność

-jest jeszcze dość młodym protokołem, podlega rozwojowi i zmianom

**UDDI (Universal Description Discovery and Integration)** – głównym zadaniem protokołu jest stworzenie niezależnego od platformy, otwartego rozwiązania do odnajdywania usług sieciowych. UDDI to globalny rejestr usług udostępniający klientom mechanizmy dynamicznego wyszukiwania innych usług Web. UDDI stanowi interfejs umożliwiający dynamiczne połączenie się z usługą udostępnianą przez innego usługodawcę. Projekt UDDI często określany jako DNS dla biznesu. UDDI jest bowiem usługą katalogową przechowującą opisy i lokalizacje Web Services zarejestrowanych przez dostawców. Wyszukiwanie usług jest podobne do działania wyszukiwarek internetowych.

UDDI posiadają dwa rodzaje klientów: usługodawców - publikujących swoje usługi, klientów - pragnących skorzystać z tych usług.

Specyfikacja UDDI opisuje również sposób wymiany danych pomiędzy serwerami, które mogą tworzyć sieć węzłów. Rejestry UDDI zawierają: informacje o webservices na bazie nazwy usługodawcy, jego adresu, kategorii biznesowej czy informacji technicznej itp., operacje dotyczące usługi Web tj. rejestracji, wyszukiwania i korzystania z usługi szczegóły udostępniane przez niskopoziomowe API

Wady i zalety

UDDI jest rozległą inicjatywą umożliwiającą systemom biznesowym odkrywanie siebie nawzajem oraz definiowanie sposobów interakcji poprzez Internet. Standard UDDI definiuje również sposoby replikacji pomiędzy repozytoriami. Obsługa UDDI powinna być integralną częścią systemów biznesowych, dzięki czemu będą one potrafiły łatwo, w sposób dynamiczny i przede wszystkim szybko, znaleźć inne systemy i współdziałać z nimi.

**PODSUMOWANIE SOAP:**

Protokół SOAP umożliwia wywoływanie i wykonywanie usług Web Services, publikowanych za pomocą WSDL i wyszukiwanych za pośrednictwem rejestru UDDI. Wykorzystywanie protokołu SOAP do wywoływania i wykonywania aplikacji poprzez Internet i korporacyjne zapory ogniowe umożliwia przedsiębiorstwom bezpieczną integrację własnych procesów biznesowych z procesami partnerów i dostawców. Korzystanie z usług przebiega w następujący sposób:

• dostarczyciele usług - rejestrują się u pośredników (UDDI).

• klient - pyta pośrednika, kto mu da daną usługę. (SOAP).

• pośrednik (broker) - lokalizuje mu tę usługę i podaje opis jej użycia (WSDL).

• klient podłącza się do dostarczyciela i korzysta z usługi (SOAP)

**Bezpieczeństwo w WS**

Problemy z bezpieczeństwem jest to największa przeszkoda w komercyjnym wykorzystaniu usług sieciowych. Usługi sieciowe wprowadzają bowiem trzy lub cztery dodatkowe warstwy pośrednie, które wymagają odpowiednich zabezpieczeń. Więcej warstw oznacza więcej możliwości dla hakerów.

Jako rozwiązanie tego problemu proponuje się stworzenie pojedynczego standardu. Najbardziej istotnym elementem bezpieczeństwa jest identyfikacja użytkownika, bowiem udostępniając usługi w Internecie, zezwalamy na korzystanie z nich w dowolny sposób. Aby ograniczyć dostęp do konkretnych usług, wystarczy zastosować kod (np. w postaci parametru wywołania usługi) umożliwiający dostęp tylko autoryzowanemu użytkownikowi. Bezpieczeństwo transmisji zapewnia SSL użyty w niższych warstwach protokołu. W przypadku powszechnie dostępnych usług, gdy nie ma możliwości przekazania kodu dostępu, konieczne jest zastosowanie innych metod uwierzytelniających.

**ESB (Enterprise Service Buses)** – Architektura ESB zakłada model centralnej szyny danych jako model połączeń między serwisami. Do szyny ESB podłączane są wszystkie serwisy związane z systemem i wszelkie wiadomości przesyłane od i do serwisów przekazywane są za jej pośrednictwem. Zaletą tego rozwiązania w stosunku do rozwiązań alternatywnych (np. bezpośrednie połączenia między serwisami) jest mała ilość połączeń między usługami.

ESB zakłada jeden, spójny model dla danych wewnątrz szyny. Nie wynika z tego jednak, że zewnętrzne serwisy integrujące się z nią też muszą mieć taki sam model danych. Wymaganie takie byłoby trudne do spełnienia i często uniemożliwiałoby integrację dwóch serwisów o różnych modelach danych. Architektura ESB rozwiązuje ten problem za pomocą adapterów, które umieszczane są na styku pomiędzy szyną a serwisami i służą do konwersji danych pomiędzy poszczególnymi modelami danych: Serwis1----Adapter-Szyna-Adapter----Serwis2

Zastosowanie wzorca ESB do integracji aplikacji daje m.in. następujące korzyści:

+Uniezależnienie od lokalizacji usług. Strona korzystająca z danej usługi nie musi znać jej konkretnej lokalizacji, w szczególności każdorazowo usługa może być dostarczana przez inny podmiot,

+Uniezależnienie od protokołu interakcji. Usługi potraﬁące komunikować się za pomocą jednego protokołu (np. HTTP/SOAP) mogą współpracować z serwisem używającym zupełnie innego protokołu (np. RMI),

+Uniezależnienie od zgodności interfejsów. Wywołujący nie jest zobligowany do dostosowania parametrów wywołania do wymagań serwisu, z którego korzysta, ale odpowiednia konwersja wywołania może zostać dokonana w ramach szyny ESB.

Zastosowanie ESB nie ma (a przynajmniej nie powinno mieć) wpływu na sposób budowy komunikujących się z jego pomocą serwisów. Z tego powodu w szczególności znajduje zastosowanie tam, gdzie nie ma możliwości ingerencji w kod źródłowy aplikacji (np. w przypadku systemów zastanych, które są kosztowne w modyﬁkacji lub systemów o nieznanym kodzie źródłowym), która jednak musi zostać przystosowana do współpracy z innym modułem. Wprowadzenie ESB jest decyzją na poziomie integracji poszczególnych części systemu rozproszonego (ang. deployment), a nie ich projektowania. Sam proces dostosowania szczegółów integracji usługobiorców i usługodawców, polegający na zmianie ﬁzycznej reprezentacji wywołania ”w locie” (ang. in-ﬂight), zwany jest mediacją wiadomości

Mediacja wiadomości

Szyna ESB nie jest tylko warstwą transportową dla wiadomości wysyłanych między serwisami. Oprócz zwykłego łączenia usług, posiada jeszcze następujące funkcje:

1. Mapowanie żądań usług z konkretnego protokołu i adresu na inny protokół/adres,

2. Transformacja danych na inny format,

3. Zarządzanie wieloma modelami transakcji i bezpieczeństwa i łączy różne modele integrowanych serwisów,

4. Agregacja żądań do serwisów,

5. Obsługa protokołów sieciowych między różnymi platformami z zachowaniem jakości usług (QoS).

Oprócz wyżej wymienionych własności, szyna ESB musi posiadać wiele sposobów integracji zewnętrznych aplikacji, które często posiadają własny, unikalny sposób dostępu, własne protokoły, modele bezpieczeństwa itd. Szyna powinna dostarczać w tym celu różne rodzaje adapterów, obsługujące wymagane protokoły komunikacji i formaty danych. Adaptery takie ukrywają detale związane z wykorzystaniem wymaganych rodzajów zasobów udostępniając interfejsy umożliwiające korzystanie z zasobów w jednolity sposób w ramach ESB. Stopień skomplikowania nie powinien być widoczny dla użytkownika serwisów, szczegóły implementacji powinny być zasłonięte za dobrze zdeﬁniowanym interfejsem adapteru. W ten sposób, przy wykorzystaniu szyny ESB, można skupić się na integracji serwisów na poziomie logicznym, pomijając szczegóły niskopoziomowego połączenia.

Wzorce mediacyjne Wyróżnia się kilka wzorców, według których dokonywana jest transformacja komunikatów przesyłanych poprzez ESB:

· Wzbogacenie (ang. enrich) - wzbogacenie zawartości wiadomości poprzez dołączenie dodatkowych parametrów pochodzących z mediacji lub np. z bazy danych,

· Przełączenie protokołu (ang. protocol switch) - zmiana protokołu komunikacji pomiędzy usługodawcą a usługobiorcą, może mieć miejsce w dowolnym punkcie na drodze pomiędzy współpracującymi stronami,

· Transformacja - zmiana formatu wiadomości z postaci rozumianej przez usługobiorcę do formatu akceptowanego przez usługodawcę; może mieć postać enkapsulacji, dekapsulacji, szyfrowania itp.

· Przekierowanie (ang. routing) - Wybór usługodawcy na podstawie określonych kryteriów. Mogą być one rozpatrywane na podstawie zawartości komunikatu, jego kontekstu, ale także na podstawie cech dostępnych usługodawców,

· Dystrybucja (ang. distribute) - rozsyła wiadomość do wielu zainteresowanych stron, rejestrujących się zwykle na zasadzie subskrypcji,

· Monitoring - umożliwia szeroko pojętą obserwację przepływających wiadomości w celach logowania, rozwiązywania problemów, pomiaru wykorzystania zasobów, w celach billingowych i innych,

· Korelacja - łączy komunikaty lub strumienie zdarzeń; zawiera reguły identyﬁkacji i rozpoznawania wzorców w komunikatach.

Powyższe wzorce można zestawiać w dowolne konﬁguracje tworząc złożone łańcuchy dokonujące pożądanej mediacji.

# 32. Narzędzia typu CASE

Narzędzia CASE (czyli Computer Aided Software Engineering lub Computer Aided System Engineering) to systemy komputerowe, przeznaczone do wspomagania rutynowych czynności procesu tworzenia oprogramowania. Dzięki nim projekty tworzy się dokładniej, a praca nad diagramami, sprawdzanie ich poprawności oraz śledzenie wykonanych testów jest prostsze i szybsze.

**Podział narzędzi CASE**

Systemy CASE można podzielić według faz cyklu życia systemu na: Upper-CASE i Lower-CASE, a także według zakresu zastosowań na pakiety narzędziowe oraz pakiety zintegrowane.

**Upper-CASE**

* wspomaganie wczesnych faz prac nad oprogramowaniem,
* w szczególności fazy analizy (potrzeby analityków i projektantów),
* narzędzia te nie są związane z konkretnym środowiskiem implementacyjnym.

**Lower-CASE**

* wspomaganie faz projektowania i implementacji (potrzeby programistów)
* narzędzia te są zazwyczaj związane z konkretnym środowiskiem implementacji (np. Java)
* notacje graficzne są bardziej naturalnym sposobem prezentacji dużych programów niż tradycyjny zapis tekstowy
* dzięki nim nie jest konieczne zapisywanie w całości kodu programu ręcznie

Istnieje także inny podział:

* **narzędzia -** wspierające tylko jedną funkcję,
* **pakiety zintegrowane -** podchodzące do projektowania kompleksowo.

**Funkcje**

Od systemów CASE wymagamy bardzo wiele. Wspomaganie w każdej fazie cyklu projektu jest inne i wymaga różnych funkcjonalności. Można jednak wyróżnić kilka standardowych modułów, których istnienie świadczy o zaawansowaniu danego systemu i spełnieniu wymagań użytkownika.

* **Business process engineering tools (BPMN) –** modelowanie procesów biznesowych.
* **Planowanie, Harmonogramowanie projektu**
* **Definiowanie wymagań na system**
* **Szacowanie kosztów**
* **Narzędzia oceny ryzyka projektu**
* **Słowniki danych (repozytoria)** – bazy wszelkich danych o tworzonym systemie wraz z narzędziami edytującymi, zarządzającymi i wyszukującymi te dane.
* **Edytor Notacji Graficznych** – program graficzny, umożliwiający tworzenie i edycję diagramów (np. UML) dla faz określania wymagań systemu, analizy i projektowania. Powinien też umożliwiać powiązania między symbolami w modelu a innymi, zdekomponowanymi modelami, oraz wydruk tych diagramów.
* **Moduł Kontroli Poprawności** – narzędzie do wykrywania i poprawiania błędów w diagramach i repozytoriach. Bardzo często działa w czasie rzeczywistym, co znacząco wpływa na komfort pracy.
* **Moduł Kontroli Jakości** – narzędzie do oceny pewnych ustalonych miar jakości projektu – np. stopnia złożoności lub powiązań składowych modelu (metryki kodu).
* **Generator testów jednostkowych**
* **Generator Raportów** – narzędzie tworzące dowolny raport na podstawie danych z repozytorium.
* **Generator Kodu** – narzędzie transformujące projekt na szkielet kodu w wybranym języku programowania. Usprawnia pracę programistów, pozwala na zautomatyzowanie pewnych fragmentów kodu, a także na uzupełnienie kodu o dodatkowe informacje ze słownika danych.
* **Generator Bazy danych** – narzędzie transformujące projekt na szkielet kodu w języku SQL lub automatycznie tworzące strukturę bazy na wybranym serwerze bazodanowym.
* **Generator Dokumentacji Technicznej** – generator ustandaryzowanych dokumentów, zawierających specyfikację, opisy faz projektu, diagramy oraz wybrane raporty.
* **Moduł Projektowania Interfejsu Użytkownika** – narzędzie do projektowania menu, okien dialogowych oraz innych elementów interfejsu użytkownika.
* **Moduł refaktoryzacji kodu**
* **Moduł Inżynierii Odwrotnej** – narzędzie pozwalające odtworzyć słownika danych oraz diagramów, na podstawie kodu źródłowego lub struktury bazy danych.
* **Moduł Importu/Eksportu Danych** – narzędzie służące do wymiany danych z innymi CASE'ami czy też innymi programami.
* **Moduł Zarządzania Pracą Grupową** – narzędzie umożliwiające współpracę grupy osób podczas pracy nad projektem (np. SVN)

Gdy chcemy zacząć pracę nad projektem, możemy to zrobić tworząc dowolny model od podstaw, ale możemy też za pomocą inżynierii odwrotnej stworzyć model, opierając się o istniejące struktury bazy danych, kod źródłowy z klasami, czy też struktury w XMLu. Gdy już będziemy mieli modele i będziemy równolegle pracować nad kilkoma etapami, może się okazać, że potrzebujemy wprowadzić zmiany w kilku modelach. Dobry system CASE potrafi powiązać zmiany w tych modelach z koniecznymi zmianami w innych modelach oraz dokonać automatycznie odpowiednich korekt.

**Co brać pod uwagę przy wyborze narzędzi:**

* Ergonomia pracy. Diagramy graficzne są jednym z podstawowych narzędzi pracy w fazach analizy i projektowania. Powinny one pozwalać analitykom i projektantom skupić się na pracy, a nie na “zmaganiach” z edytorem.
* Możliwość kontrolowania ilości informacji prezentowanej w sposób graficzny.
* Jakość i możliwość formatowania wydruków.
* Wykrywanie na bieżąco konstrukcji niepoprawnych.
* Zapewnienie spójności informacji umieszczonych na różnych diagramach.

**Popularne narzędzia**

* **Eclipse** – darmowe, otwarte środowisko programistyczne dla Javy, które za pomocą platformy modelowania Eclipse (Eclipse Modeling Framework) może posłużyć do budowania oprogramowania, wykorzystując także UML. EMF posiada także generator kodu.
* **NetBeans** – otwarty projekt zawierający wiele narzędzi wspomagających tworzenie oprogramowania. Dodatkowo "Enterprise Pack" umożliwia modelowanie UML oraz użycie schematów XML.
* **StarUML** – Zapewnia forward i reverse engineering kodu w Javie, C# i C++.
* **Borland Together** – rodzina programów integrujących środowisko IDE Javy z narzędziami do UMLa. Posiada m.in. funkcje modelowania danych, szablony kodu, generator dokumentacji, czy też moduł weryfikacji kodu.
* **Enterprise Architect** generowanie kodu: ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, EJB, XML Schema, Ada, VHDL, Verilog, WSDL, BPEL, Corba IDL ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, XML Schema, WSDL (dla wielu istnieje także wsparcie reverse engineeringu) Integracja z: Eclipse, Visual Studio, TcSE , Modelowanie: UML 2.4.1, SysML, BPMN, SoaML, SOMF, WSDL, XSD, ArchiMate. Frameworks: UPDM, Zachman, TOGAF. Forward and Reverse Engineering for code and Database. Model Driven Integrated Developement (Edit/Build/Debug) for Java, .Net, PHP & GNU compilers. Simulates Activity, State Machine, Sequence and BPMN diagrams.
* **Visual Paradigm** - Generowanie kodu: Java, C#, C++, PHP, Ada, Action Script Reverse engineering: Java, C# (binary), C++, PHP Integracja z: Eclipse, NetBeans and IntelliJ Full UML 2.1, SysML, ERD, BPMN, data modeling,business modeling and reverse engineering of source code and dtabases

# 33. Podejścia do zarządzania zespołami realizującymi projekty informatyczne

**KILKA UWAG O PRACY ZESPOŁOWEJ**

Większość profesjonalnego oprogramowania jest tworzona przez zespoły składające się od dwustu do kilkuset osób (Sommervile). Nie ma możliwości, aby wszyscy członkowie tak wielkiego zespołu pracowali razem na jednym problemem. Duże zespoły dzieli się na kilka grup. Każda grupa odpowiada za budowę jakiegoś podsystemu. Przyjmuje się zasadę, że grupy nie powinny liczyć więcej aniżeli ośmiu członków. Stworzenie małych grup umożliwia ograniczenie problemów komunikacyjnych. Cała grupa może się spotkać przy jednym stole lub zebrać w swoich pokojach, nie są potrzebne skomplikowane struktury komunikacyjne. Kiedyś celem zarządzania było kontrolowanie siły roboczej, wskazywanie co trzeba zrobić i pilnowanie, aby było to zrobione. Obecnie uznaje się, że takie podejście nie daje przewagi konkurencyjnej, a raczej odwrotnie. Obecnie pracownicy są dobrze wykształceni i oczekują, że będą włączani w przedsięwzięcie. A zatem, gdy kierownictwo nadal musi koordynować pracę ludzi i grup, to pomału odchodzi się od podejścia opartego na władzy i ścisłej kontroli. Nowoczesne podejście polega na wsparciu siły roboczej, umożliwianiu wykonania pracy oraz dbaniu o rozwój poprzez szkolenie i zachęcanie do samorozwoju. Utworzenie grupy, która będzie wydajnie pracować, jest krytycznym zadaniem menadżera. W grupie powinna panować równowaga umiejętności technicznych, doświadczenia i osobowości. W dobrej grupie panuje duch zespołu, tzn. jej członkowie są zmotywowani sukcesem grupy na równi z realizacją własnych celów. Menadżerowie powinni zachęcać do jawnych czynności „budowania zespołu”, aby wypracować poczucie lojalności grupowej.

**KILKA UWAG O MOTYWOWANIU**

Kierownik musi motywować zespół jako całość oraz motywować osobno każdą osobę. Motywacja zespołu ma swoje źródło w osobistym zaangażowaniu kierownika, w sposobie przydziału i podziału pracy, wyraźnej wizji celu i sposobu jego osiągnięcia. Kierownik daje przykład przez własne zaangażowanie i zachowanie oraz tworzy klimat postępu i akceptacji zmiany. Motywację osobistą osiąga się poprzez własny stosunek i „niepisaną umowę”, czego dana osoba i kierownik wzajemnie od siebie oczekują. Kluczowym elementem motywacji jest projektowanie pracy poszczególnych osób; praca powinna zawierać odpowiednią ilość wyzwań i różnorodności oraz zmierzać do znaczącego wyniku. Każdy potrzebuje uzgodnionych celów, które są zrozumiałe i splatają się z osobistym i zawodowym rozwojem kariery, wynikającym z ambitnej pracy, profesjonalnych standardów, właściwych relacji i szkolenia.

Pożądane cechy członków zespołu:

* podatność na oddziaływanie kierownictwa projektu,
* umiejętność pracy zespołowej,
* wysokiej klasy umiejętności techniczne,
* silna orientacja na rozwiązywanie problemów,
* silne nastawienie na osiąganie rezultatów,
* wysoka samoocena,
* konstruktywna krytyka

W tabeli poniżej opisane zostały czynniki, które rozważa się przy doborze personelu.

**Tabela 1 Czynniki doboru personelu**

|  |  |
| --- | --- |
| **Czynnik** | **Opis** |
| Doświadczenie w dziedzinie zastosowania | Jeżeli projekt ma zakończyć się sukcesem, to jego twórcy muszą rozumieć dziedzinę zastosowania |
| Doświadczenie w pracy z platformą | Może być istotne, jeżeli prace obejmują programowanie na niskim poziomie. W przeciwnym wypadku zwykle nie jest krytycznym atrybutem |
| Doświadczenie w pracy z językiem programowania | Zwykle jest istotne jedynie w wypadku krótkotrwałych projektów, przy których nie ma czasu na naukę nowego języka |
| Zdobyte doświadczenie | Może być podstawową wskazówką co do tego, co kandydat powinien umieć, i o jego zdolności do uczenia się. Ten czynnik jest mniej istotny, gdy programiści zdobywają coraz więcej doświadczenia w licznych projektach |
| Zdolności komunikacyjne | Są ważne, ponieważ członkowie zespołu muszą porozumiewać się pisemnie i ustnie z innymi współpracownikami, menedżerami i klientami |
| Zdolność do przystosowywania się | Tę zdolność można ocenić przyglądając się różnym rodzajom doświadczeń zdobytych przez kandydata. To ważny atrybut, bo z niego wynika zdolność do uczenia się. |
| Nastawienie | Członkowie zespołu powinni być pozytywnie nastawieni do wykonywanej pracy i chcieć zdobywać nowe umiejętności. To bardzo ważny atrybut, ale zwykle bardzo trudny do oceny. |
| Osobowość | To bardzo ważny atrybut, ale zwykle bardzo trudny do oceny. Kandydaci muszą być racjonalnie zgodni z innymi członkami zespołu. Żaden konkretny typ osobowości nie jest mniej lub bardziej odpowiedni do inżynierii oprogramowania |

**Podejścia w zarządzaniu zespołami**

* **ZESPÓŁ LIDERA-PROGRAMISTY**

Wg Bakera, Arona i Brooksa najskuteczniejsze wykorzystanie dobrych programistów osiąga się przez zbudowanie zespołu wokół jednego wysoko wykwalifikowanego lidera-programisty.

Głowni członkowie zespołu lidera-programisty:

* **Lider - programista** – bierze całkowitą odpowiedzialność za zaprojektowanie, zaprogramowanie, przetestowanie i instalację systemu.
* **Doświadczony programista zapasowy** – wspiera lidera-programistę, bierze odpowiedzialność za zatwierdzanie oprogramowania.
* **Dokumentator** – przejmuje wszystkie funkcje urzędnicze projektu, takie jak zarządzanie konfiguracjami, redagowanie dokumentów, opracowywanie dokumentacji.
* Zależnie od rozmiarów i rodzaju tworzonego oprogramowania, mogą być potrzebni inni specjaliści do czasowej lub stałej pracy w zespole. Mogą to być administratorzy, specjaliści od systemów operacyjnych i języków, testerzy itp.

Uzasadnienie: najlepsi programiści są do 25 razy bardziej wydajni od najgorszych. Pomysł ma już 30 lat, a ciągle jest skutecznym sposobem organizacji małych grup tworzących oprogramowanie.

Problemy:

* Liczba utalentowanych projektantów i programistów jest niewielka. jeżeli oni popełnią błędy, to nikt nie będzie kwestionował ich decyzji.
* Lider-programista bierze całą odpowiedzialność i może przypisywać sobie wszystkie zasługi w wypadku sukcesu. Członkowie grupy mogą być niezadowoleni, jeżeli ich rola w przedsięwzięciu nie jest doceniana. Ich potrzeba szacunku nie będzie zaspokojona.
* Przedsięwzięcie będzie zagrożone, gdy lider-programista zachoruje lub odejdzie z firmy. Zarząd firmy może nie chcieć zaakceptować takiego ryzyka.
* Struktury organizacyjne mogą nie być zdolne do przyjęcia takiej grupy. Wielkie firmy mają starannie zdefiniowaną strukturę.
* **PROGRAMOWANIE EKSTREMALNE**

Programowanie Ekstremalne (XP) to lekka, zwinna (*ang. agile*) metodyka tworzenia oprogramowania

**Podstawowe zasady XP:**

* Najważniejsza komunikacja ustna.
* Jedyne artefakty: kod + testy
* IEEE/ANSI standard 830/1993? **Zbędny !!!**

*(IEEE Recommended Practice for Software Requirements Specifications)*

* Inspekcje Fagana**? Zbędne !!!**
* Punkty funkcyjne? **Zbędne !!!**
* Żadnych nadgodzin!

**12 praktyk XP wg Kenta Becka:**

1. **Planowanie**. Tworzenie oprogramowania w XP odbywa się przyrostowo przez wdrażanie kolejnych wydań produktu. Planowanie wydania odbywa się przed rozpoczęciem każdej nowej iteracji. Podstawowym celem jest oszacowanie każdej pojedynczej historii użytkownika, powstałej w wyniku gry planistycznej z klientem. Do szacowania używa się jednostek zwanych idealnymi osobo-tygodniami. Idealny osobo-tydzień to tydzień pracy wyłącznie nad programem, bez dodatkowych zajęć, ale wliczający czas testowania programu.
2. **Małe wydania**. Małe kroki to częste łączenie kodu napisanego przez programistów. Osiąga się je przez podział zadania na małe historie użytkownika. Dzięki temu pojedynczy fragment kodu może być łatwo i szybko wykonany, przetestowany i złączony z reszta systemu. Małe wydania to częste akceptacje powstałego systemu przez klienta. Dzięki ciągłym testom i łączeniu zawsze istnieje sprawnie działająca wersja, a klient nie musi długo czekać na kolejną.
3. **Metafora systemu**. Każdy zespół programistyczny musi kontaktować się z klientem. Aby możliwe było wzajemne porozumienie potrzebne jest opracowanie wspólnego słownika używanych pojęć. Aby uniknąć problemów z komunikacja oraz ze słownikiem XP stosuje metaforę systemu. Jest to nic innego jak analogiczne do projektowanego oprogramowania pojęcie, które jest w sposób oczywisty zrozumiałe na klienta i dla programisty
4. **Prosty projekt**. XP zakłada, ze wymagania klienta, rynku i sytuacja w branzy ciągle się zmieniają. Nie ma wiec sensu planować rozwiązań, o których nie wiadomo, czy zostaną wykorzystane w przyszłości. Celem XP jest jak najszybsze i najprostsze osiągnięcie satysfakcji klienta przez dostarczenie oprogramowania, spełniającego postawione wymagania.
5. **Ciągłe testowanie**. Ciągłe testowanie to podstawowe działanie podczas pisania programu w metodzie XP. Programista jeszcze przed napisaniem danej procedury tworzy kod, który ma ją testować. W ten sposób wcześniej musi pomyśleć o wszystkich rzeczach, które mogą ’pójść źle’. Dzięki temu podczas pisania właściwego kodu procedury zabezpieczy ja przed tymi możliwościami. Pisanie procedury testowej nie powinno jednak trwać zbyt długo i nie powinna być ona zbyt rozbudowana. Zespół dąży do automatyzowania procedur testowych, które są uruchamiane po każdorazowym łączeniu kodu oraz po przerabianiu.
6. **Przerabianie**. Przerabianie (ang. refactoring) jest konieczne zaraz po przetestowaniu działającej procedury. Przerabianie to według autora sformułowania „poprawianie projektu istniejącego kodu”. Należy pamiętać, że przerabianie nie może zmieniać zewnętrznego zachowania programu.
7. **Programowanie w parach**. Jednym z bardziej krytykowanych aspektów XP. Jednakże diametralnie zmienia ono sposób pisania kodu. Podczas gdy jedna osoba (trzymająca klawiaturę) pisze kod, druga na bieżąco go sprawdza, sugeruje możliwe rozwiązania, może służyć pomocą i zwraca uwagę na błędy syntaktyczne. Tak powstały kod jest nie tylko lepszy ale i łatwiej oraz szybciej się kompiluje. Według Kenta [6] pary powinny się miedzy sobą mieszać. Również programiści wewnątrz pary powinni co jakiś czas zamieniać się miejscami.
8. **Standard kodowania**. XP narzuca wszystkim programistom wspólny standard kodowania i dokumentowania. Standard taki powinien być ustalony i zaakceptowany przez cała grupę. Standard powinien jednoznacznie określać wygląd kodu, ale nie powinien być zbyt długi i szczegółowy. Polecane są opracowania jednostronicowe. Standard dokumentowania zakłada, ze samych komentarzy w kodzie jest jak najmniej. Klasy powinny być tak zaprojektowane by przeznaczenie poszczególnych metod było jasne, a samo działanie oczywiste.
9. **Wspólna odpowiedzialność.** Dzięki standardom kodowania każdy programista czuje się jak ’u siebie’ w każdym fragmencie systemu, nawet jeśli go nie pisał. Zbiorowa praca nad kodem, to jednak nie tylko wspólne pisanie go, ale i wspólna odpowiedzialność za niego. Jeśli trzeba cos zmodyfikować nie ma problemu, bo poprawki może zrobić każdy. XP preferuje umieszczenie całej grupy programistów w jednym pomieszczeniu, co ma pomagać w komunikacji i rozwijaniu życia grupy. Zostawia jednak dla każdego jego prywatną przestrzeń. Do pracy w parach powinny być wyznaczone oddzielne komputery.
10. **Ciągłe łączenie**. Ciągłe łączenie to integracja programu tak często, jak to tylko możliwe. Programista po wykonaniu każdego nowego fragmentu programu łączy go z systemem. Najczęściej stosuje się jedna maszynę, na której w danej chwili może pracować jedna osoba zajmująca się łączeniem kodu. Ciągłe łączenie jest ułatwione w XP dzięki prostym projektom, ciągłym testom i wspólnej odpowiedzialności za kod.
11. **40-godzinny tydzień pracy**. Swego rodzaju symbolem, znakiem rozpoznawczym XP, stało się wymaganie 40-to godzinnego tygodnia pracy. Zespoły programistów powinny być przyzwyczajone do stałej wydajności i stałego obciążenia.
12. **Ciągły kontakt z klientem**. Aby zadowolić wymagania klienta należy bezwzględnie podążać za jego życzeniami. Co jednak zrobić, gdy klient zapomniał nas o czymś poinformować lub mamy problem który wymaga przekonsultowania? Potrzebny jest kontakt z klientem. Często jest on jednak nieosiągalny, co doprowadza do opóźnień w realizacji projektu. XP zakłada ciągłą możliwość konsultacji z klientem ’na żywo’. W praktyce oznacza to codzienna obecność klienta w zespole programistów. Niektórzy twierdzą, że klient nie jest poważny, jeśli nie może poświęcić wystarczająco dużo czasu dla nowego systemu.

**Zespół w XP:**

* W Extreme Programming zespół jest wspólnotą ludzi, którzy razem dążą do celu. Są nie tylko odpowiedzialni za projekt ale i troszczą się o niego. Darzą się wzajemnie szacunkiem i maja silne poczucie więzi.
* W zespole XP poza programistami są też inne osoby, odpowiedzialne za zarządzanie, oraz pomagające rozwiązywać szczególnie trudne problemy. Są to:
  + trener,
  + zarządca,
  + tester,
  + konsultant,
  + szef
  + klient.

Wszystkie one mają bardzo ściśle określone kompetencje i nie powinny wzajemnie sobie wchodzić w drogę. Wszyscy są odpowiedzialni za proces powstawania aplikacji, ale gdy zespół zdobędzie pewne doświadczenie nie wszyscy będą potrzebni.

* **METODOLOGIA SCRUM**
* **Scrum** to metodyka prowadzenia projektów. Zaliczana do tzw. metodyk lekkich, (zwinnych, ang. Agile Project Management), zgodnych z The Agile Manifesto. Najczęściej wykorzystywana jest w projektach informatycznych.
* Nazwa "scrum", czyli po polsku "młyn", wywodzi się z terminu występującego w grze rugby.
* Używana jest w skomplikowanych projektach, w których nie można przewidzieć wszystkiego, co może się przydarzyć lub w przypadku przedsięwzięć o wysokim stopniu innowacyjności.

Metodyka skupia się na:

* dostarczaniu kolejnych, coraz bardziej dopracowanych wyników projektu,
* włączaniu się przyszłych użytkowników w proces wytwórczy,
* **samoorganizacji** zespołu projektowego.

Zespół i role:

* Zazwyczaj zespół Scrum składa się od 5 do 9 osób. Dobrze, gdy ma ***charakter interdyscyplinarny*** i składa się z osób reprezentujących ***różne umiejętności***. Osoby uczestniczące w zespole nie mogą uczestniczyć w innych zespołach.

Główne role w projekcie grają:

* "Mistrz Młyna" (Scrum Master),
* Właściciel Produktu (Product Owner)
* Członkowie Zespołu (The Team).

**Opis metodyki:**

* 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).**
* Po etapie przygotowawczym zespół przechodzi do realizacji przebiegu (sprinta).
* W jego trakcie Właściciel Produktu nie może ingerować w prace zespołu.
* Nie powinno się także zmieniać zakresu sprintu.
* Jako że zespół z założenia jest samoorganizującym się ciałem, nie ma mowy o odgórnym przypisywaniu zadań do poszczególnych członków zespołu, lecz samodzielnie dokonują oni wyboru realizowanych zadań, według wspólnych ustaleń, umiejętności czy innych preferencji.
* Naczelną zasadą metodyki jest przeprowadzanie codziennych (około 15 minut) **spotkań (z ang. scrum meeting),** na których omawiane są zadania zrealizowane poprzedniego dnia i problemy występujące przy ich realizacji oraz zadania do wykonania w dniu spotkania.
* Sprint kończy się *spotkaniem Sprint review*, na którym prezentowany jest wynik pracy zespołu poprzez prezentowanie produktu wykonanego podczas przebiegu.
* Powinni w nim uczestniczyć wszyscy zainteresowani projektem.
* Na spotkaniu, każdy członek zespołu może zabrać głos i wyrazić opinię o produkcie.
* Po omówieniu produktu ustalany jest termin spotkania planistycznego do następnego przebiegu.

ŹRÓDŁA:

[1] Wykład z ZPI, B. Trawiński

# 34. Podpis elektroniczny

# 35. Specyficzne własności struktur baz danych w systemach informacyjnych

Cechy BD ważne dla SI:

* Bezpieczeństwo
  + Możliwość zabezpieczenia danych, niezawodność i integralność danych, obsługa transakcji, systemy do ich zarządzania i administracji
* Wydajność
* Wielodostęp
* Otwartość
  + Elastyczny dostęp, współpraca z różnymi źródłami danych
* Możliwy rozwój
  + Skalowalność, dostęp zdalny, przenośność

Główne typy BD najczęściej stosowane w SI:

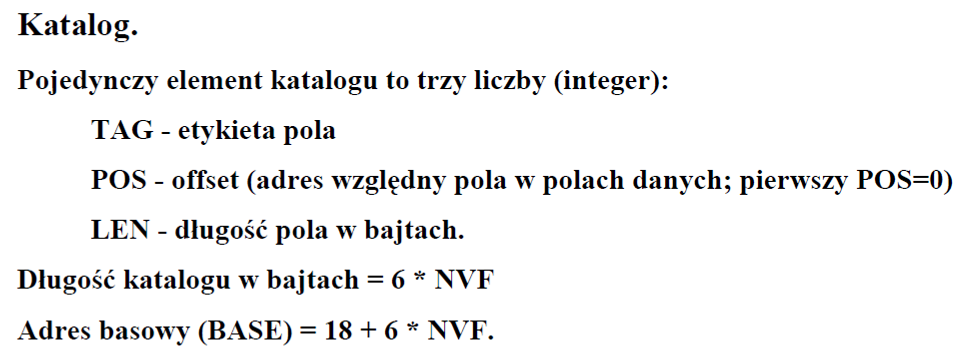
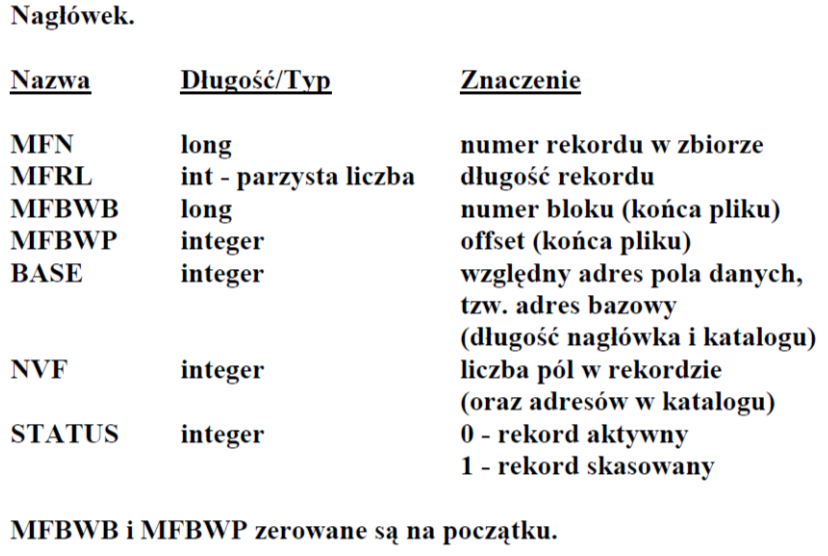
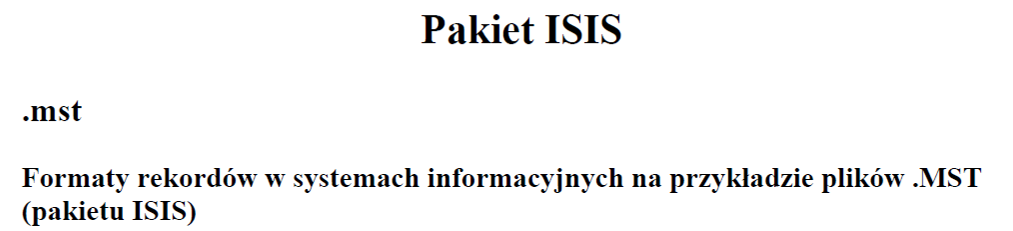
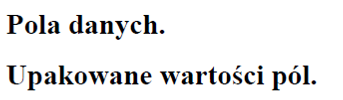
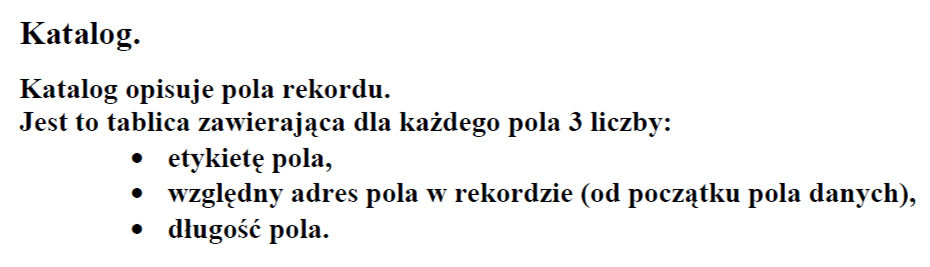
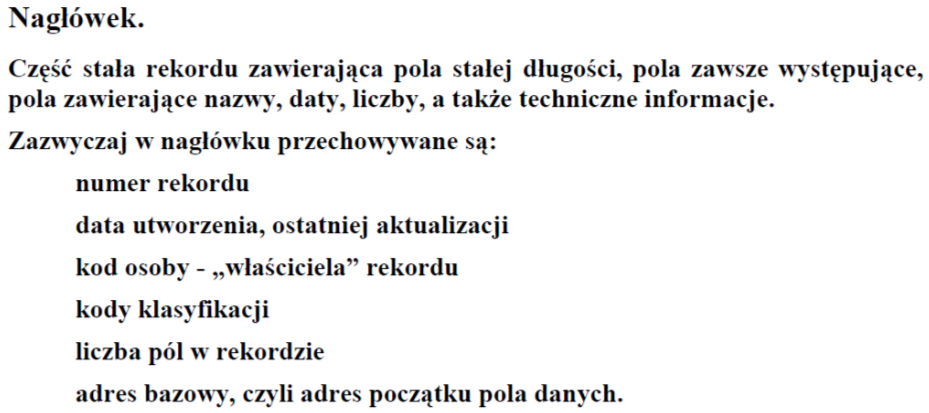
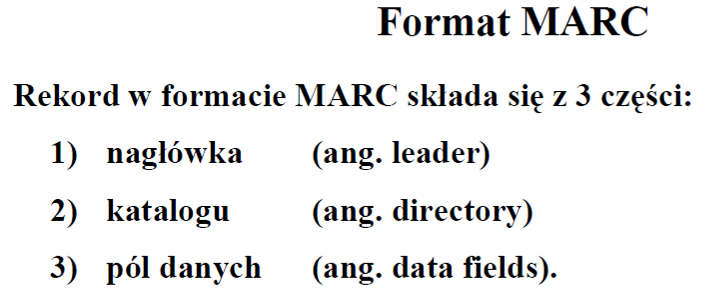
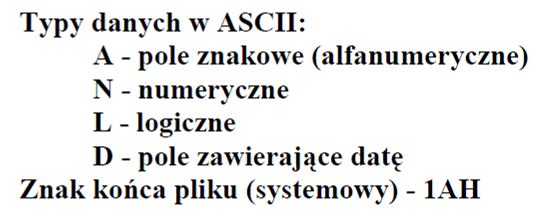
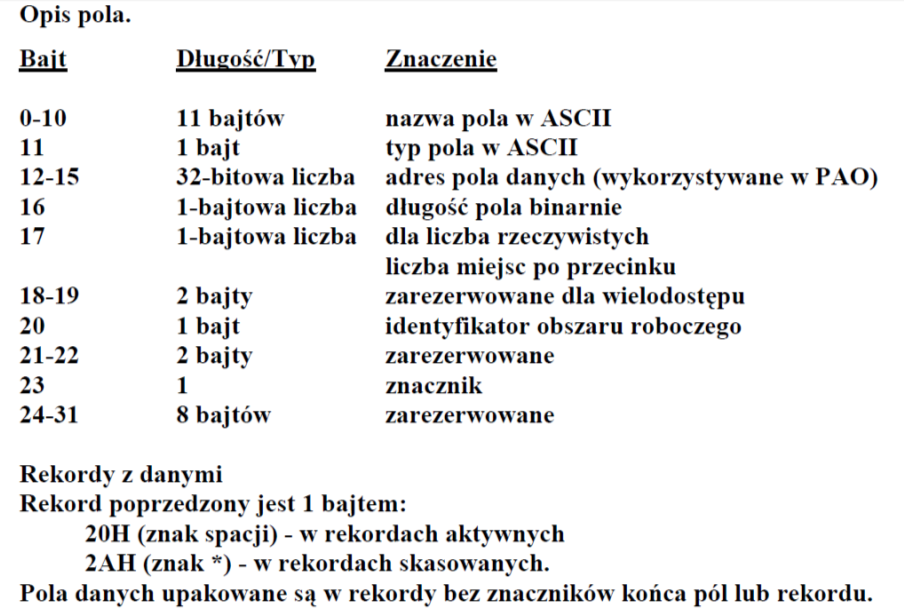
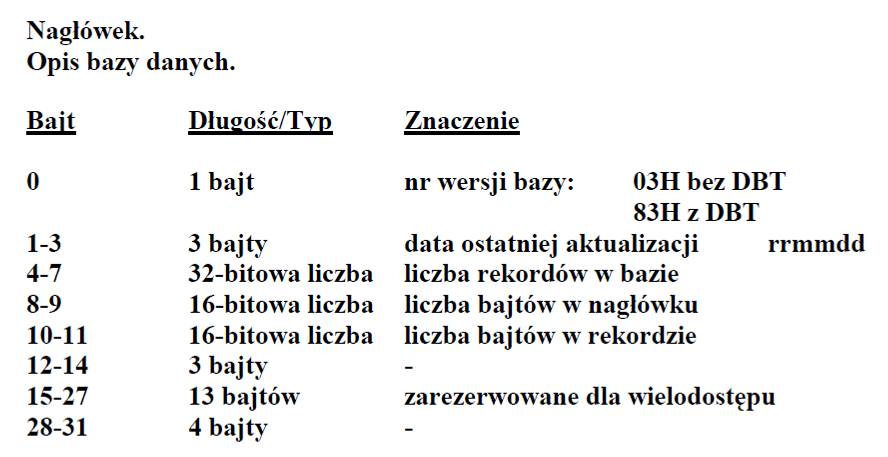
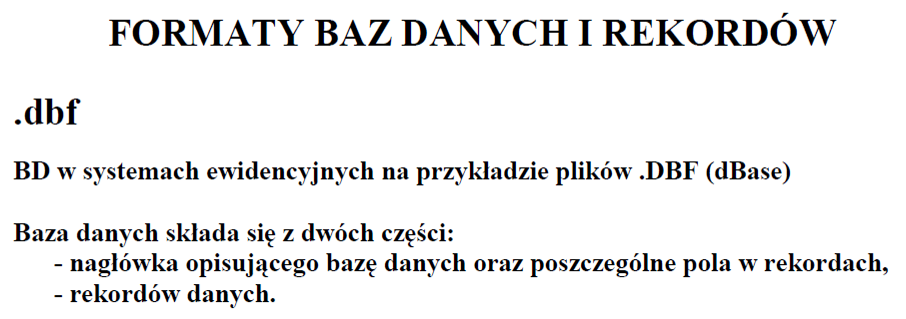
1. Systemy transakcyjne (OLTP)

* Gromadzenie danych + modyfikacja
* Zapis / Odczyt
* Są to standardowe bazy danych (można opowiadać bez końca)

1. **Systemy analityczne (OLAP)!**

* Ważne, ponieważ w systemach informacyjnych dane muszą być dostępne i użyteczne. Nie ma sensu ich gromadzić jeśli nie można ich przetwarzać. Technologia OLAP umożliwia pozwala na obróbki analityczne i zestawieniowe.
* Typowe dla hurtowni danych
  + Hurtownia danych to **tematyczna** baza danych, która **trwale** przechowuje **zintegrowane** dane opisane **wymiarem czasu**
  + TIMESTAMP – Dane mają sens jedynie kiedy jest znane ich źródło i wiek
  + Dane bieżące i historyczne
  + Zbierane dane mogą pochodzić z wielu źródeł w dużym przedziale czasowym.
* READ ONLY

|  |  |  |
| --- | --- | --- |
| **Cecha** | **OLTP** | **Hurtowania Danych** |
| Czas odpowiedzi  Aplikacji | Milisekundy - sekundy | Sekundy - godziny |
| Wykonywane operacje | DML | SELECT |
| Zakres danych w czasie | 30 – 60 dni | 2 – 10 lat |
| Typ danych | Pod aplikację | Tematyczne |
| Rozmiar | Małe - duże | Duże – wielkie |
| Wykorzystanie operacji dyskowych | małe | Wielkie |



# 36. Standardy zapewnienia jakości oprogramowania

Trawiński wykład 10.

# 37. Strategie realizacji systemu informacyjnego

**1. STRATEGIA OPARTA NA DOKUMENTACH**

(ang. document driven)

Bardzo dokładna realizacja w oparciu o model kaskadowy. Proces podzielony jest na następujące po sobie fazy, które są wykonywane kolejno, jedna po drugiej, bez powracania do faz już wykonanych. Model kaskadowy wymusza kończenie dokumentacji po każdej fazie, dokumentacja jest weryfikowana i zatwierdzana przez kierownictwo projektu. Zakończenie działań i zatwierdzenie wyników poprzedniej fazy jest niezbędnym warunkiem rozpoczęcia fazy następnej. Stosowany w projekcie o dobrze zdefiniowanych wymaganiach dla dobrze rozumianych zastosowań.

ZALETY: Model kaskadowy ułatwia planowanie, harmonogramowanie czy monitorowanie przedsięwzięcia. Jasno uporządkowanie pracy, brak powtarzania działań.

WADY: trudności przy formułowaniu wszystkich wymagań, wysokie koszty błędów popełnionych we wcześniejszych fazach, powoduje długie przerwy w kontaktach z klientem.

Fazy modelu kaskadowego:

Etap 1: określenie wymagań

Etap 2: analiza

Etap 3: Projektowanie

Etap 4: Implementacja

Etap 5: Integracja i testowanie

Etap 6: Wdrożenie

**2. PROTOTYPOWANIE (ang. prototyping)**

Prototypowanie – sposób na uniknięcie zbyt wysokich kosztów błędów popełnionych w fazie określania wymagań. Stosowany w projekcie o dobrze zdefiniowanych wymaganiach dla dobrze rozumianych zastosowań.

Prototyp – ogólny model przyszłego systemu informatycznego, który w kolejnych iteracjach jest udoskonalany aż do osiągnięcia akceptowalnego stopnia szczegółowości. Prototyp to prowizoryczna implementacja, mającca tylko pewne wybrane cechy finalnego produktu. Podstawowy cel tworzenia prototypu – weryfikacja i uściślenie wymagań.

ZALETY:

- wykrycie nieporozumień pomiędzy klientem i projektantami

- wykrycie brakujących funkcji w projekcie (w systemie)

- wykrycie trudnych funkcji

- wykrycie braków w specyfikacji wymagań

- możliwość szybkiej demonstracji pracującej wersji systemu

- możliwość szkoleń zanim zbudowany zostanie cały system.

WADĄ może być niezadowolenie klienta, który po obejrzeniu działającego prototypu musi następnie długo czekać na dostawę gotowego systemu

Fazy prototypowania:

1) Ogólne określenie wymagań

2) Budowa prototypu

3) Weryfikacja prototypu przez klienta

4) Pełne określenie wymagań

5) Realizacja pełnego systemu zgodnie z modelem kaskadowym

**3. MONTAŻ Z GOTOWYCH KOMPONENTÓW**

(ang. composition of reusable components)

Przyśpieszona realizacji systemu na bazie własnych lub kupionych komponentów, modułów, bibliotek

programistycznych. Kładzie nacisk na możliwość redukcji nakładów poprzez wykorzystanie podobieństwa tworzonego oprogramowania do wcześniej tworzonych systemów oraz wykorzystanie gotowych komponentów dostępnych na rynku.

ZALETY: wysoka niezawodność, zmniejszenie ryzyka, realizacja wg standardów, efektywne wykorzystanie nie tylko własnych doświadczeń, jednak mimo zakupów zmniejszenie kosztów.

WADY: zwykle brak odpowiednich, w pełni zgodnych z oczekiwaniami komponentów, uzależnienie się od zewnętrznych firm programistycznych i zewnętrznych standardów, dodatkowy koszt w przypadku tworzenia własnych bibliotek.

**4. REALIZACJA W OPARCIU O MODEL SPIRALNY**

(ang. spiral model)

Charakteryzuje się iteracyjnym przetwarzaniem wszystkich rodzajów działań (lub jej części) w każdej fazie. Każda kolejna iteracja zwiększa wiedzę o wszystkich aspektach tworzonego oprogramowania.

Wyróżnia się w nim cztery etapy, przez które przechodzą poszczególne fazy (czynności) w trakcie wytwarzania oprogramowania (czynnościami jest m.in. określanie wymagań, projektowanie i tworzenie kolejnych prototypów):

1) Planowanie - w kontakcie z klientem, ustalenie celów produkcji kolejnej wersji systemu

2) Analiza ryzyka - alternatyw oraz szacowanie i minimalizacja ryzyka

3) Konstrukcja - realizacja, testowanie, wdrożenie

ZALETY:

- Do dużych systemów - szybka reakcja na pojawiające się czynniki ryzyka

- Połączenie iteracji z klasycznym modelem kaskadowym

WADY:

- Trudno do niego przekonać klienta

- Konieczność umiejętności szacowania ryzyka

- Problemy, gdy źle oszacujemy ryzyko

**5. MODEL PRZYROSTOWy**

(ang. incremental programming)

Jest to swoista ewolucja modelu kaskadowego - odchodzi się w niej od idei opracowania kompletnej funkcjonalności na samym początku. Wymagania rozpoznawane są w trakcie tworzenia systemu i dodawane do sytemu. Jest to odmiana modelu spiralnego. Wybierany jest i realizowany podstawowy zestaw funkcji. Po realizacji pewnych funkcji następuje zrealizowanie i dostarczenie kolejnych funkcji.

ZALETY: bardzo szybka możliwość demonstracji pracującej i w pełni sprawnej części systemu, a w związku z tym możliwość szkoleń, a nawet częściowego wdrożenia systemu i użytkowania ograniczonej wersji systemu, częste kontakty z klientem, brak konieczności zdefiniowania z góry całości wymagań.

WADY: większe koszty realizacji, dodatkowy koszt związany z niezależną realizacją fragmentów systemu

**6. TRANSFORMACJE FORMALNE**

(ang. formal transformation)

Zakłada on, że wymagania wobec systemu specyfikowane są w pewnym formalnym .języku. Wymagania te są następnie transformowane w automatyczny sposób do pewnej postaci pośredniej bliższej kodowi w pewnym języku programowania. Postać ta podlega dalszym automatycznym transformacjom do kolejnych form coraz bliższych kodowi. Jedna z kolejnych postaci formalnych .jest już na tyle bliska kodowi, że może być w automatyczny sposób przełożona na kod w konkretnym języku programowania. Wszystkie te transformacje wykonywane są bez udziału ludzi. Poszczególne fazy tego modelu można traktować jak transformacje (nieformalne) wyników poprzedniej fazy do kolejnej postaci bliższej kodowi.

ZALETY: wysoka niezawodność tak stworzonego oprogramowania

WADY: raczej metoda teoretyczna, wąskie zastosowania, język formalnej specyfikacji jest praktycznie językiem programowania, mała efektywność, Trudność formalnego wyspecyfikowania wymagań, Brak dobrze rozwiniętych, uniwersalnych języków formalnej specyfikacji wymagań

# 38. Systemy biometryczne

Biometryka jest nauką zajmującą się identyfikowaniem i weryfikowaniem tożsamości osoby na podstawie jej cech fizjologicznych lub behawioralnych zwanych biometrykami.

Ważnym elementem biometryki są jej cechy. Pozwalają one scharakteryzować każde żywe stworzenie oraz rozróżnić je od innych przedstawicieli tego samego gatunku. Można podzielić je na dwie grupy: behawioralne oraz fizjologiczne. Elementy behawioralne to te, które dotyczą zachowania w czasie. Są to między innymi: sposób wpisywania na klawiaturze, głos, sposób chodzenia. Druga grupa to cechy fizjologiczne czyli takie cechy, które posiadają informacje o cechach fizycznych i są do zmierzenia w danej chwili. Są to np. odcisk palca, kształt dłoni czy też wygląd tęczówki.

1. Identyfikacja i weryfikacja.

Podstawowe wykorzystanie biometryki w systemach informacyjnych to weryfikacja oraz identyfikacja użytkowników. Te dwa pojęcia często są mylone ze sobą dlatego warto je przytoczyć i pokazać jaka jest różnica. Identyfikacja proces ustalający kim jest dana osoba. Mechanizm pobiera próbkę od danej osoby i wyszukuje w dużej bazie danych taką próbkę, która odpowiada pobranym.

Weryfikacja ma na celu natomiast sprawdza czy podana osoba jest tą, za którą się podaje. Tutaj pobierana jest próbka by sprawdzić czy dane wcześniej zebrane od tej osoby są takie same.Różnice między tymi procesami występują w trzech blokach procesu. Pierwszy blok to pobieranie – identyfikacja pobiera jedynie próbki, weryfikacja natomiast oprócz próbki potrzebuje również identyfikatora. Podczas działania procesu identyfikacja sprawdza całą bazę wyszukując rekordu zgodnego z elementem wzorca, natomiast w weryfikacji próbka porównywana jest tylko ze wzorcem posiadającym podany wcześniej identyfikator. Ostatnia róznica zachodzi w wyniku procesów. Identyfikacja daje wynik w postaci rozpoznania lub odrzucenia, weryfikacja akceptuje lub rozróżnia próbkę.

Opis metod indentyfikacji i weryfikacji opisane w pkt 26 **TECHNOLOGIE BIOMETRYCZNE .**

W biometryce możemy wyróżnić dwa podstawowe typy błędów:

**FAR** (ang. False Acceptance Rate) - błąd fałszywej akceptacji występuje, gdy osoba spoza bazy danych (a właściwie jej dane biometryczne) systemu zostaje rozpoznana jako jedna z bazy,

**FRR** (ang. False Rejection Rate) - błąd fałszywego odrzucenia występuje, gdy osoba zarejestrowana w bazie danych (jej dane biometryczne) systemu nie zostaje rozpoznany jest odrzucona przez system.

Większy błąd FRR świadczy o większym bezpieczeństwie systemu. W razie fiasku identyfikacji użytkownik może ponownie podać się temu procesowi. Przyczyny tego błędu są różne. Przeważnie wpływ na to mają warunki zewnętrzne: ubrudzenie czytnika, złe oświetlenie, bądź też brak szkolenia użytkowników.

Istnieje również błąd klasyfikacyjny który rozpoznaje użytkownika jako inną osobę z danej bazy.

Największe szanse na rozwój biometryka posiada tam gdzie wazny jest poziom bezpieczeństwa tj: kontrole dostępu do zakladu penitencjarnego, komputery służb specjalnych itp. Kierunek, w którym ta technologia powinna się rozwijać to standaryzacja i unifikacja metod. Ważne jest by systemy były jednolite, by np. w przypadku sposobu pisania na klawiaturze nie używały różnych cech, gdyż może to spowodować błędy. Wraz z rozwojem technologii przewidziany jest też spadek kosztów czujników, co pozwoli na rozpowszechnienie technologii na szerszym rynku.

**Klasyfikacja metod biometrycznych**

* Statyczne (fizjologiczne)
  + Metoda Bertillona
  + Odciski palców
  + Rozpoznawanie twarzy
  + Skan tęczówki
  + Skan siatkówki
  + Geometria dłoni
  + Dopasowywanie wzorców naczyń krwionośnyc
* Dynamiczne (behawioralne
  + Rozpoznawanie mowy
  + Rozpoznawanie pisma odręcznego
  + Odstępy między kliknięciami klawiszy klawiatury podczas pisania (Keystroke dynamics)

# 39. Systemy pośredniczące w wymianie danych

(wykłady Kazienki, Trawińskiego, pytanie 31)

Omawiam EDI , Web Servicy i CORBE myślę, że tyle wystarczy jeśli ktoś chce więcej to DCOM, który można by tutaj omówić został już omówione przez Sylwię w pytaniu 31)

**EDI (Electronic Data Interchange)** – ustandaryzowana metoda wymiany danych pomiędzy komputerami bądź sieciami komputerowymi, z wykorzystaniem standardowych, zaakceptowanych formatów komunikatów. EDI to sposób wymiany informacji pomiędzy kontrahentami, charakteryzujący się przesyłaniem dokumentów wyłącznie w formie elektronicznej („bezpapierowej”), wymianą dokumentów spełniających uzgodnione wcześniej standardy (definicja pól dokumentu i ich zawartości), wymianą danych bezpośrednio pomiędzy systemami informatycznymi partnerów handlowych; dane będące przedmiotem elektronicznej wymiany są odpowiednikami tradycyjnych dokumentów handlowych takich jak: faktura, zamówienie, awizo dostawy, harmonogram produkcji itp. Kluczowym aspektem EDI jest standaryzacja przesyłanych informacji oraz bezpieczeństwo komunikacji.

EDI polega na wymianie danych pomiędzy systemami informatycznymi, nie pomiędzy ludźmi. Dane te mogą być automatycznie przetwarzane przez komputer.

EDI nie określa sposobu przesyłania komunikatów – mogą one być przesyłane przez dowolne medium, którym posługują się obie strony transmisji. Może to być transmisja modemowa, poprzez FTP, HTTP, AS1, AS2.

EDI łączy możliwości informatyki i telekomunikacji. Umożliwia eliminację dokumentów papierowych zwiększając efektywność wszystkich działań związanych z handlem. Dokładniej mówiąc wykorzystanie EDI wiąże się z spadkiem kosztów własnych ponoszonych na łączności i wymianie informacji między firmami. Uzyskanie przewagi nad konkurencją, szybsza reakcja na zmiany i potrzeby. Wprowadzenie EDI w węższym zakresie nie wymaga wielu zmian w organizacji ani metodach działania firmy (jedynie przysyłania dokumentów i informacji) Redukcja błędów powstających wskutek tradycyjnych metod wymiany danych np. błędów przy wypełnianiu papierkowych dokumentów. Optymalizacja procesów np. zamówień i zleceń, wszystko odbywa się natychmiastowo bez potrzeby oczekiwania na papierowe zamówienia. Największym minusem jednak samego systemu jest jego wdrażanie, które jest raczej trudnym zadaniem, a błędy nie wykryte na początku mogą propagować w przyszłości.

**Web Servicy -** Są to usługi internetowe śwwiadczone poprzez sięc telekomunikacyjną, a w szczególności przez internet. Same Web Servicy zazwyczaj są tylko składnikiem oprogramowania, który jest niezależny od platformy oraz implementacji. Web Serivcy są opisywane przez język WSDL, język ten opsiuje sposób wywoływania usług, ich parametry, zwracane wyniki oraz udostępniane metody. Ogólnie WSDLe pozwalają twórcom usług internetowych je opisać, napisać gdzie się one znajdują oraz w jaki sposób można z nich skorzystać. WSDL bazuje na XML.

Komunikaty wysyłane przez Web Servicy są wysyłane przy użyciu protkołu SOAP, jest to protokół, mechanizm transportu informacji, służący zazwyczaj do jednokierunkowej komunikacji czyli w przypadku Web Serviców będzie to komunikacja nadwaca-klient, podobnie jak WSDL SOAP jest oparty na XML.

UDDI jest to rejestr globalny usług, które zawiera ich opis oraz lokalizację. Głównym jego zadaniem jest umożliwienie klientom dynamicznego wyszukiwania usług udostępnianych w internecie. Użytkownikami UDDI są zarówno klienci (osoby chcące skorzystać z Web Servicu) jak i usługodawcy, ponieważ usługodawca udostępnia swoje usługi a informacje przypisane do klienta zawierają szczegóły poszukiwanych przez niego usług.

Najczęsciej używanym przez usługi internetowe protokołem do przekazywania danych jest protkołu HTTP oraz wykorzystanie plików XML (są to zalecenia W3C).

Główną zaletą Web Serviców jest jak wspomniałem ich niezależność od platformy oraz implementacji, co pozwala na udostępnianie różnorodnych usług różnym klientom bez potrzeby zmiany chociażby systemu operacyjnego czy używanego języka programowania.

**CORBA** - technologia ta umożliwia komunikację pomiędzy obiektami pracującymi w różnorodnych systemach. Podobnie jak w przypadku Web Serviców obiekty te mogą być zaimplementowane w różnych językach proramowania na dowolnej platformie sprzętowej pod dowolnym systemm operacyjnym.

Również podobnie jak w przypadku Web Serviców CORBA wymusza opis obiektów (interfejsów do obiektów) w postaci plików IDL, który jest kompilowan na kod zajmujący się przekazywaniem metod.

Również jak przy Web Servicach, obiekty mają jakiś swój adres nazywany IOR, które zawierają dodatkowo wiele informacji o obiekcie m.in. adres komputera, adres programu na komputerze, numer obiektu, typ obiektu, itd.

CORBA jak i Web Servicy nie należą do najbezpieczniejszych rozwiązań, nie mniej w przypadku Web Serviców podobno coś się robi, żeby to bezpieczeństwo zwiększyć nie wiem jak w przypadku CORBY. Największym chyba minusem CORBY jest jej wydajność, która jest mało efektywna w czasie rzeczywistym oraz duże koszty jej wdrożenia. Podobnie jak DCOM CORBA najczęsciej jest opracowywana na sieci prywatnych, dlatego też z uwagi na szerszy zakres Web Servicy mają szanse zastąpić to rozwiązanie.

# 40. Techniki modelowania danych i przepływu danych

(Tutaj chyba chodzi o to co było na Chorosiu w zimowym semestrze więc diagramy ERD i DFD)

**ERD (Entity-relationship diagram)** - Diagram związków encji odzwierciedla statyczną strukturę systemu, a konkretnie związki istniejące pomiędzy poszczególnymi zbiorami danych.

Składa się z:

* typów obiektów - encji - przedstawia się ja za pomocą prostokątów. Typ obiektu reprezentuje zbiór obiektów, reczy ze świata rzeczywistego, które mogą być jednoznacznie identyfikowane oraz opisane przez jeden lub więcej atrybutów.
* związki - przedsawia się za pomocą rombów. Związki reprezentują wszelkie powiązania i asocjacje występujące pomiędzy encjami.

Encja:

* reprezentuje zbiór obiektów opisanych tymi samymi cechami (atrybuty, własności)
* konkretny obiekt świata rzeczywistego jest instancją encji
* każda encja posiada swoją unikalną nazwę oraz zbiór cech
* encje wchodzą w związki z innymi encjami.
* dowolny obiekt może być reprezentowany przez tylko jedną encję
* nazwa encji - rzeczownik w liczbie pojedynczej
* encja musi posiadać unikatowy identyfikator ( naturalni - NIP, PESEL; sztuczny - numer pozycji katalogowej, id pracownika )
* encja może być silna bądź słaba. Encja jest słaba gdy jej wystąpienie jest uwarunkowane występowaniem innych encji, tzn. zależy od kontekstu. Przykładem może być encja ‘realizacja’, która występuje tylko wtedy gdy występują encje ‘pracownik’ i ‘projekt’. Tylko encje silne mają własny identyfikator.

Związek:

* reprezentuje powiązania pomiędzy obiektami świata rzeczywistego (łączy encje)
* może zawierać krótki opis ułatwiający jej interpretację
* posiada 3 cechy:
  + stopień związku
    - unarny (1 encja)
    - binarny (2 encje)
    - ternarny (3 encje)
    - n-narny (n encji)
  + typ asocjacji
    - jeden-do-jeden ( 1:1 ) - np. jeden zespół może mieć tylko jednego kierownika, natomiast każdy zespół musi mieć kierownika.
    - jeden-do-wiele ( 1:N ) - np. Każdy pracownik pracuje w jednym dziale. Dział może zatrudniać wielu pracowników.
    - wiele-do-wiele ( M:N ) - np. Pracownik może brać udział w wielu projektach. Każdy projekt jest realizowany przez co najmniej jednego pracownika.
  + klasa przynależności
    - obowiązkowa
    - opcjonalna
* może posiadać atrybuty np. jak pracownik bierze udział w jakimś projekcie, to atrybutem może być jego wynagrodzenie

W przypadku diagramów ERD występuje jeszcze generalizacja - mówi ona o tym, że pewne encje o wspołnych cechach można uogólnić do jednej encji wyższego poziomu - encję genrealizacji. Przykładem generalizacji może być pracownik i klient, dla których taką encją będzie encja osoba.

**DFD (DataFlow Diagram)** - Diagram przepływu danych z kolei opisuje dynamiczne właściwości systemu.

Składa się z:

* procesów - proces to pojedyncza funkcja realizowana przez system. Graficznie jest reprezentowana przez kółko.
* przepływów - przepływ to związek między procesami. Graficznie jest reprezentowany przez strzałkę.
* zbiorów danych - zbiory danych to magazyny danych, które przechowują jak sama nazwa wskazuje dane. Graficznie mogą być reprezentowane w dwojaki sposób albo poprzez elipsy, albo nazwę między dwiema pionowymi kreskami.
* terminatorów - terminatorem określa się obiekt zewnętrzny, który komunikuje się z systemem, np. klient. Graficznie jest reprezentowany przez prostokąt.

Schematy te wzbogacane są również o tekstowe narzędzia modelowania:

* Specifykację danych - są to słowniki zawierające definicje struktur danych.
* Specyfikację procesów - zawiera idee oraz specyfikacje konkretnych algorytmów.

# 41. Tłumaczenie komputerowe tekstów w językach naturalnych

(Tutaj będę bazował całkowicie na starym opracowaniu, które zostało zrobione na podstawie Wikipedii i wybiorę, to co uznam za najważniejsze, bo szczerze mowiąc z zajęć w ogóle tego nie pamiętam, więc nie wiem co tu powinno się znaleźć)

**Tłumaczenie automatyczne** albo **tłumaczenie maszynowe** (ang. Machine Translation) jest dziedziną językoznawstwa komputerowego, które zajmuje się stosowaniem algorytmów tłumaczenia tekstu z jednego języka (naturalnego) na drugi. Główne metody, przez które realizowane jest automatyczne tłumaczenie:

● Systemy tłumaczenia bezpośredniego – wyrazy tekstu źródłowego zamieniane są tu wprost na tłumaczenie w oczekiwanym języku. Program zawiera odpowiadające sobie słowa i najczęściej stosowane frazy. Tłumaczenie tego typu daje akceptowalne wyniki tylko w zastosowaniu dla blisko ze sobą spokrewnionych języków. (Coś jak słownik)

● Systemy przekładu składniowego – analizują składniową stronę tekstu. Najczęściej rezultatem jest drzewo składników, do którego następnie stosuje się odpowiednie reguły transferu.

● Systemy oparte na powierzchniowym transferze semantycznym – biorą pod uwagę własności składniowe i częściowo znaczeniowe. Realizowane jest to poprzez dołączenie do drzewa struktury syntaktycznej dodatkowych informacji naprowadzających, np. atrybutów znaczeniowych.

● Systemy międzyjęzykowe – oparte są na uniwersalnym języku reprezentacji znaczenia (tzw. interlingwę), który jest niezależny od języków naturalnych, zawartych w systemie. Proces translacji składa się z dwóch etapów: tłumaczenia z języka źródłowego na interlingwę i tłumaczenia z interlingwy na język wynikowy. (skojarzeniem może być proces komunikacji w EDI, gdzie wysyła się plik z firma A, który jest zamieniany przez XSLT na uniwersalny format, a następnie zamieniany przez inny XSLT na plik w formacie firmy B)

● Tłumaczenie statystyczne – tłumaczenie w oparciu o wielkie zestawy (korpusy) tekstów przetłumaczonych przez człowieka. Dla danego zdania szukane jest jego najbardziej prawdopodobne tłumaczenie. Prawdopodobieństwo tłumaczenia obliczane jest na podstawie współwystępowania wyrazów w zebranym korpusie. Sukcesy w tym podejściu notuje portal Google, gdyż korzysta ze swoich olbrzymich korpusów stron internetowych.

● Tłumaczenie oparte na przykładach – podobnie jak tłumaczenie statystyczne opiera się na istniejących tekstach przetłumaczonych. Dla danego zdania źródłowego system szuka najbardziej podobnego przykładu w swojej bazie danych i na tej podstawie wnioskuje jego tłumaczenie.

Kwestie/problemy semantyczne związane z automatycznym tłumaczeniem:

● Trzy rodzaje sensowności:

* + Sensowność lokucyjna – związana jest ona z językiem naturalnym i jest słownikowym znaczeniem znaków. Nie zależy ona od kontekstu sytuacyjnego. Sensowność ta jest stopniowalna (np. prosimy o pojaśnianie).
  + Sensowność logiczna – nie ujawnia się empirycznie. Sensowność ta jest związana językiem logiki. Jednakże język logiki komunikowalny jest tylko poprzez język naturalny. Powstaje problem przekładu. Nie jest ona stopniowalna, ani nie zależy od kontekstu sytuacyjnego.
  + Sensowność wolicjonalno-emotywna – ujawnia się w kontekstach sytuacyjnych. Wypowiedzi mogą być niedostosowane do konwencji sytuacyjnej.

● Metaforyczność stanowi ważny problem. W języku naturalnym znajdują się metafory (nie ma ich w języku logiki). Kiedy mamy do czynienia z metaforą? Wtedy gdy zostanie złamana zasada kompozycji znaczeniowej, która mówi, że znaczenie wyrażenia całościowego jest funkcją znaczeń wyrażeń składowych. Np. zdanie: „Matematyka jest moją piętą achillesową” oznaczałoby dosłownie, nie metaforycznie, że matematyka jest dla mnie częścią nogi mitycznego herosa. Jednakże nie można pozbyć się metafor, gdyż język straciłby swą moc informacyjną.

● Synonimia. Nawet w jednym języku nastręcza ona trudności przez to, iż wyrazy mają różny zakres pojęciowy. W tłumaczeniu problem ten ulega tylko powiększeniu. Częstokroć bywa tak, że tłumaczony wyraz nie ma swego odpowiednika w drugim języku.

● Homonimy są kolejnymi problemami w automatycznym tłumaczeniu. Właściwa interpretacja jest przy nich bardzo ważna. Np. zdanie: „Podszedł do zamku” można interpretować na różne sposoby i interpretacje są zależne od kontekstu. Pojawia się tu kwestia umiejętności donoszenia się do kontekstu przez automatycznego tłumacza.

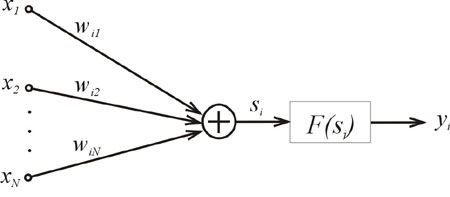
● Dużą grupę problemów stanowią różnice w składni pomiędzy językami. Np. w języku angielskim istotne jest miejsce wyrazu w zdaniu (jest to język pozycyjny), natomiast w języku polskim nie.

● Innymi problemami jest występowanie w języku rodzajników określonych i nieokreślonych lub podmiotu domyślnego. Powstaje również pytanie o możliwość stworzenia języka pośredniego w tłumaczeniu (interlingwa z systemu międzyjęzykowego), biorąc pod uwagę dotychczasowe rozważania.

# 42. Własności sieci neuronowych

Pierwowzorem sieci neuronowych jest oczywiście mózg ludzki. Mózg ma objętość 1400 cm3 i powierzchnię 2000 cm2 (kula o tej samej objętości ma zaledwie 600 cm2). Masa mózgu wynosi 1,5 kg. Są to wartości średnie, ponieważ istnieją różnice pomiędzy poszczególnymi ludźmi. Dla przykładu mózgi kobiet są z reguły mniejsze niż mężczyzn. Nie ma to jednak większego znaczenia, gdyż nie udowodniono do tej pory związku między wielkością mózgu, a jego intelektualną sprawnością.

Na podstawie zasad działania rzeczywistego neuronu stworzono wiele modeli matematycznych, w których uwzględnione zostały w większym lub mniejszym stopniu właściwości rzeczywistych komórek nerwowych. Schemat obwodowy powiązany z większością tych modeli odpowiada modelowi McCullocha-Pittsa.



**Rys. Schemat budowy pojedynczego neuronu**

Jego działanie jest następujące:

Do wejść doprowadzane są sygnały dochodzące z neuronów warstwy poprzedniej. Każdy sygnał xj mnożony jest przez odpowiadającą mu wartość liczbową zwaną wagą wij. Wpływa ona na percepcję danego sygnału wejściowego i jego udział w tworzeniu sygnału wyjściowego przez neuron. Zsumowane iloczyny sygnałów i wag stanowią argument funkcji aktywacji neuronu f(si).

Prosty perceptron jest zwykłym modelem McCullocha-Pittsa o odpowiednio przyjętej strategii uczenia. Wagi sumatora prowadzące do sygnałów wejściowych xj oznaczono poprzez wi0. Funkcja aktywacji perceptronu jest nieliniowa typu skokowego, co oznacza, że sygnał wyjściowy neuronu może przyjmować tylko dwie wartości 0 lub 1, zgodnie z zależnością



gdzie si jest sygnałem wyjściowym sumatora



We wzorze tym założono, że wektor x o długości N został rozszerzony o składową zerową x0=1, stanowiącą sygnał polaryzacji

Uczenie perceptronu należy do grupy uczenia z nauczycielem regułą perceptronową, które zostało szerzej omówione w rozdziale poświęconym metodzie uczenia perceptronu.

Cechą charakterystyczną reguły perceptronu jest wykorzystanie w uczeniu jedynie informacji o aktualnej wartości sygnału wyjściowego neuronu i wartości zadanej.

Minimalizacja różnic między odpowiedziami aktualnymi neuronu yi i wartościami zadanymi diodpowiada minimalizacji określonej funkcji błędu (funkcji celu) E, definiowanej najczęściej jako



gdzie p oznacza liczbę zadanych wzorców uczących. Minimalizacja ta w regule perceptronu odbywa się zgodnie z metodą bezgradientową optymalizacji. Efektywność metody przy dużej liczbie wzorców jest stosunkowo niewielka, a liczba cykli uczących i czasu uczenia wzrasta szybko, nie dając przy tym gwarancji minimalnej funkcji celu. Usunąć te wady można jedynie przy przyjęciu ciągłej funkcji aktywacji, przy której funkcja celu E staje się również ciągła, co umożliwia wykorzystywanie w procesie uczenia informacji o wartości gradientu.

Architektura sieci jednokierunkowej jednowarstwowej

W sieci jednokierunkowej jednowarstwowej neurony są ułożone w jednej warstwie, zasilanej z węzłów wejściowych. Połączenie węzłów wejściowych z neuronami warstwy wyjściowej jest zwykle typu każdy z każdym tzn. każdy węzeł z każdym neuronem. Przepływ sygnałów występuje w jednym kierunku, od wejścia do wyjścia.

Węzły wejściowe nie tworzą warstwy neuronów, gdyż nie zachodzi w nich żaden proces obliczeniowy. Sposób doboru wag i wybór metody uczenia decuduje tutaj o nazwie sieci. Metody uczenia są łatwe i proste w implementacji praktycznej. Sieć tego rodzaju może stanowić zarówno sieć Kohenena, jak i perceptron jednowarstwowy. Będą z tym także związane rodzaje funkcji aktywacji oraz metody uczenia.

Architektura sieci jednokierunkowej wielowarstwowej

Cechą charakterystyczną sieci jednokierunkowych wielowarstwowych jest występowanie, co najmniej jednej warstwy ukrytej neuronów, która pośredniczy w przekazywaniu sygnałów między węzłami wejściowymi, a warstwą wyjściową.

Sygnały wejściowe są podawane na pierwszą warstwę ukrytą neuronów, a te z kolei stanowią sygnały źródłowe dla kolejnej warstwy. W sieci tej występują połączenia pełne między warstwami. W szczególności w niektórych zastosowaniach pewne połączenia międzyneuronowe mogą nie wystąpić i mówi się wówczas o połączeniu częściowym, lokalnym. Zwykle połączenia dotyczą wtedy części warstwy poprzedniej, skupionej w określonym obszarze tworzącym pole recepcyjne danego neuronu. Neurony warstw ukrytych stanowią bardzo istotny element sieci, umożliwiający uwzględnienie związków między sygnałami, wynikającymi z zależności statystycznych wyższego rzędu. Uczenie perceptronu wielowarstwowego odbywa się zwykle z nauczycielem.

Sieci jednokierunkowe wielowarstwowe często wykorzystują nieliniową funkcję aktywacji typu sigmoidalnego i stanowią naturalne uogólnienie perceptronu Rosenblatta.

Architektura sieci rekurencyjnej

Sieci rekurencyjne różnią się od sieci jednokierunkowych występowaniem sprzężenia zwrotnego między warstwami wyjściowymi i wejściowymi. Można tu wyróżnić sieci jednowarstwowe mające jedynie jedną warstwę neuronów wyjściowych oraz sieci mające dodatkową warstwę ukrytą - sieci wielowarstwowe rekurencyjne.

Proces ustalania się sygnałów wyjściowych sieci rekurencyjnych jest procesem dynamicznym ze względu na występowanie jednostkowych operatorów opóźnienia. Biorąc pod uwagę nieliniowość funkcji aktywacji neuronów jest to dynamika nieliniowa, stanowiąca istotną cechę tego rodzaju sieci.

**Metody uczenia:**

• **Z nadzorem**

• **Bez nadzoru**

• **Z krytykiem**

**Własności sieci:**

Jako obiekt badań sieci neuronowe stanowią bardzo uproszczony, przez co łatwiejszy do ogarnięcia myślą lub do zamodelowania na komputerze, ale bogaty i ciekawy model rzeczywistego biologicznego systemu nerwowego.

Składają się one z połączonych ze sobą obiektów umownie zwanych neuronami. Istotną cechą sieci takich elementów jest możliwość uczenia się - tj. modyfikowania parametrów charakteryzujących poszczególne neurony w taki sposób, by zwiększyć efektywność sieci przy rozwiązywaniu zadań określonego typu.

Sieci neuronowe mogą być bardzo skuteczne jako narzędzia obliczeniowe w rozwiązywaniu takich zadań, z którymi typowe komputery i typowe programy sobie nie radzą. Po pierwsze obliczenia są w sieciach neuronowych wykonywane równolegle, w związku z czym szybkość pracy sieci neuronowych może znacznie przewyższać szybkość obliczeń sekwencyjnych. Drugą zaletą sieci jest możliwość uzyskania rozwiązania problemu z pominięciem etapu konstruowania algorytmu rozwiązania problemu.

Sieci nie trzeba programować. Istnieją metody uczenia i samouczenia pozwalające uzyskać ich celowe i skuteczne działanie nawet w sytuacji, kiedy twórca nie zna algorytmu, według którego można rozwiązać postawione zadanie.

Sieć działa zawsze jako całość i poszczególne jej elementy mają wkład w realizację wszystkich czynności, które sieć realizuje. Jedną z konsekwencji takiego działania sieci jest jej zdolność do poprawnego działania, nawet po uszkodzeniu znacznej części wchodzących w jej skład elementów.

Topologia sieci powinna wynikać z rodzaju zadania, jakie jest stawiane przed nią. Decyzje dotyczące struktury sieci nie wpływają na jej zachowanie w stopniu decydującym. Zachowanie jej w zasadniczy sposób zależy od metody jej uczenia.

Sieci neuronowe mogą całą swoją wiedzę zyskiwać wyłącznie w trakcie nauki i nie muszą mieć z góry zadanej, dopasowanej do stawianych im zadań, jakiejkolwiek precyzyjnie określonej struktury. Sieć musi jednak mieć wystarczający stopień złożoności, żeby w jej strukturze można było w toku uczenia "wykrystalizować" potrzebne połączenia i struktury. Zbyt mała sieć nie jest w stanie nauczyć się niczego, gdyż jej "potencjał intelektualny" na to nie pozwala - rzecz jednak nie w strukturze, a w liczbie elementów.

**Zastosowania**

Do dziś najczęstszym spotykanym obszarem zastosowań technicznych sieci neuronowych są zagadnienia rozpoznawania, a zwłaszcza problem rozpoznawania kontekstowego. Sieci neuronowe stosuje się także do zadań klasyfikacji oraz do analizy obrazów i ich przetwarzania. Wyróżnić można tu kompresję, segmentację, odtwarzanie oraz rozumienie obrazów. Aby sklasyfikować i rozpoznawać wzorce, sieć uczy się ich podstawowych cech takich jak: odwzorowania geometryczne, pikselowego układu wzorca, rozkładu składników głównych wzorca, składników transformacji Fouriera czy innych jego właściwości. W uczeniu podkreślane są różnice występujące w różnych wzorcach.

Kolejna dziedziną zastosowań jest wykorzystanie sieci neuronowych do klasycznych zadań przetwarzania sygnałów, takich jak konwersje, filtracje i aproksymacje oraz inne odwzorowania i transformacje. Coraz więcej prac opisuje struktury takich sieci i ich zastosowanie.

Inne, często spotykane, zastosowania sieci neuronowych dotyczą robotyki, automatyki, a także teorii sterowania i zagadnień optymalizacji, percepcji ruchu i jego planowania. W zagadnieniach identyfikacji i sterowania procesami dynamicznymi sieć neuronowa pełni zwykle kilka funkcji. Stanowi model nieliniowy tego procesu, pozwalający na wypracowanie odpowiedniego sygnału sterującego. Pełni również funkcje układu śledzącego i nadążnego, adaptując się do zmiennych warunków środowiskowych. Ważna rolę, zwłaszcza w sterowaniu robotów, odgrywa funkcja klasyfikatora wykorzystywana w podejmowaniu decyzji, co do dalszego przebiegu procesu.

Sieci neuronowe stosowane są do zbudowania pamięci asocjacyjnej, zwłaszcza duże zainteresowanie budzi klasa tak zwanych dwukierunkowych pamięci asocjacyjnych BAM. W zadaniach asocjacji sieć neuronowa pełni role pamięci skojarzeniowej. W przypadku pamięci asocjacyjnej, skojarzenie dotyczy tylko poszczególnych składników wektora wejściowego. W pamięci heteroasocjacyjnej zadaniem sieci jest skojarzenie ze sobą dwóch wektorów. Gdy na wejście będzie podany wektor odkształcony, sieć neuronowa jest w stanie odtworzyć wektor oryginalny, pozbawiony szumów.

Sieci neuronowe mają liczne i różnorodne zastosowania poza techniką np.w ekonomii. Oto ważniejsze kierunki tych zastosowań.

Sieci neuronowe są wykorzystywane do przewidywania pewnych określonych rozwiązań na podstawie danych początkowych. Przykładami mogą być wszelkiego rodzaju prognozy ekonomiczne np. przewidywania bessy i hossy na giełdzie.

Sieci mogą klasyfikować na podstawie danych bilansowych czy dane przedsiębiorstwo należy do zwyżkujących gospodarczo czy też może przeżywa stagnację lub regres. Można klasyfikować, w które branże warto inwestować lub który region może być zagrożony bezrobociem.

Dzięki zdolnościom adaptacji, sieci wykorzystuje się do procesów wnioskowania na podstawie zgromadzonych danych. Ułatwiają one managerom wykrywanie istotnych powiązań i ważnych aspektów.

Analiza danych prowadzona z wykorzystaniem sieci pozwala na ustalenie np. przyczyn niepowodzeń określonych przedsięwzięć podejmowanych w przeszłości, dzięki czemu łatwiejsze jest unikanie błędów na przyszłość.

Optymalizacja - sieci poszukują rozwiązań prowadzących do optymalnych decyzji gospodarczych. Sieć taka może rozwiązać np. klasyczne zadanie zwane problemem komiwojażera.

Mimo ogromnej liczby zastosowań sieci neuronowych, możliwości ich dalszego wykorzystania w przetwarzaniu sygnałów są jeszcze nie do końca zbadane i wydaje się, że będą one jeszcze przez wiele lat stanowić o postępie w technice informacyjnej. Prace nad dalszymi zastosowaniami trwają...

# 43. Zarządzanie ryzykiem w projekcie informatycznym

**Ryzyko** to prawdopodobieństwo wystąpienia niepożądanego zdarzenia oraz niepożądanego wpływu tego zdarzenia na cele projektu. To prawdopodobieństwo, że w danym punkcie cyklu życia oprogramowania, jego zaplanowane cele nie zostaną osiągnięte w ramach dostępnych zasobów. w projekcie informatycznym jest niepomyślnym dla przebiegu projektu zdarzeniem, prowadzącym do niepożądanego rezultatu.

Ryzyko = potencjalny problem

**Zarządzanie ryzykiem** jest to systematyczny proces identyfikowania, analizowania i reagowania na ryzyka w projekcie. Obejmuje maksymalizację prawdopodobieństwa i wpływu pozytywnych zdarzeń na cele projektu oraz minimalizację prawdopodobieństwa i wpływu negatywnych zdarzeń.

Zarządzanie ryzykiem rozumie się jako podejmowanie działań mających na celu [120];

· rozpoznanie,

· ocenę i

· sterowanie ryzykiem oraz

· kontrolę podjętych działań.

Celem zarządzania jest ograniczanie ryzyka oraz zabezpieczanie się przed jego skutkami.

**Struktura zarządzania ryzykiem**

Zarządzanie ryzykiem:

· Ocena ryzyka: proces, w którym w uporządkowany sposób prowadzi się badanie zagrożeń związanych z obiektem technicznym. Celem oceny ryzyka jest zebranie informacji niezbędnych do podjęcia decyzji o bezpieczeństwie projektu.

o Analiza ryzyka: proces, w czasie którego identyfikuje się zarówno prawdopodobieństwo wystąpienia zdarzenia niepożądanego jak i zakres związanych z tym szkód (konsekwencji)

§ Określenie zakresu

§ Identyfikacja zagrożeń

§ Analiza zagrożeń

§ Oszacowanie ryzyka

o Wyznaczanie ryzyka

§ Decyzje o dopuszczalności ryzyka

§ Analiza opcji

· Sterowanie/zmniejszanie ryzyka

o Podejmowanie decyzji: Wśród strategii postępowania z ryzykiem wyróżnia się przede wszystkim następujące podejścia do zarządzania ryzykiem: Unikanie, Przeniesienie, Minimalizacja, Akceptacja

o Zastosowanie

o Monitorowanie: ustalenie, czy dane ryzyko rzeczywiście miało miejsce i mogło wpłynąć na bieżące działania projektowe, a w przypadku sytuacji wyjątkowej: ustalenie przyczyny, zebranie doświadczeń na potrzeby bieżącego i przyszłych przedsięwzięć

Analizę ryzyka można podzielić na:

1) Jakościową: proces oceny prawdopodobieństwa wystąpienia oraz wpływu na projekt zidentyfikowanych czynników ryzyka. W procesie tym stosuje się między innymi **macierz ryzyka**, która jest jedną z prostych metod wspomagania zarządzania ryzykiem. Macierz tworzy się w procesie analizy zagrożeń. W trakcie analizy każdemu zagrożeniu przypisujemy dwa skwantyfikowane parametry: prawdopodobieństwo wystąpienia zdarzenia oraz wpływ tego zdarzenia na projekt. Pozwala to umieścić wszystkie ryzyka w tabeli. Przykładowo, konsekwencje mogą wystąpić w 5stopniowej skali: krytyczne, bardzo duże, średnie, niskie, znikome, a prawdopodobieństwo: prawie pewne, prawdopodobne, średnio, mało prawdopodobne, rzadko.

2) Ilościową: Proces oceny liczbowej (mierzalnej) wartości prawdopodobieństwa i konsekwencji czynników ryzyka dla projektu oraz ich wpływu na cele przedsięwzięcia. Przykładową metodą ilościowej analizy ryzyka może być analiza drzew decyzyjnych. Drzewa opisywane są za pomocą „kosztu” wyboru każdej z opcji, dzięki czemu możliwe jest wyznaczenie wartości „nagrody” lub „kosztu”. Otrzymane wyniki umożliwiają jednoznaczną ocenę potencjalnych decyzji.

Trzy wymiary istotnie wpływające na ryzyko:

· wielkość projektu, obejmująca m.in. przewidywany czas realizacji, koszt, liczbę działów organizacji objętych projektem

· doświadczenie technologiczne, odnoszące się zarówno do organizacji, jak i firmy tworzącej i wdrażającej system

· uniwersalności systemu (mała uniwersalność odpowiada systemom dedykowanym, zaś duża uniwersalność systemom standardowym, wykorzystywanym w wielu organizacjach)

Obszary ryzyka projektu informatycznego:

Technologia, sprzęt, koszty, ludzie, terminy, oprogramowanie.

# 44. Zarządzanie zmianami i konfiguracjami oprogramowania

Zarządzanie konfiguracjami (ang. Configuration management) jest dziedziną zarządzania skupiającą się na tworzeniu i utrzymywaniu produktu lub systemu w stanie spójnym, spełniającym założenia funkcjonalne i wydajnościowe zdefiniowane przez użytkowników. Zarządzanie konfiguracjami pozwala na spełnienie wszystkich tych założeń w całym cyklu życia produktu. Z punktu widzenia osób wykorzystujących mechanizmy zarządzania konfiguracjami – jest to narzędzie, które pozwala na łatwe i efektywne śledzenie zmian konfiguracyjnych zachodzących w zarządzanym systemie. Zmiany konfiguracyjne mogą dotyczyć zarówno warstwy sprzętowej systemu, jak i jego warstw programowych – począwszy od systemu operacyjnego z jego kolejnymi poprawkami, poprzez bazy danych i serwery aplikacyjne, konfiguracje środowiska sieciowego, skończywszy wreszcie na konfiguracji urządzeń klienckich.

Mechanizmy „zarządzania konfiguracjami” zostały po raz pierwszy zdefiniowane w roku 1950 przez Departament Obrony Stanów Zjednoczonych. W kolejnych latach zostały on zaakceptowany przez wiele modeli zarządzania projektami technologicznymi (nie tylko informatycznymi). Z najważniejszych należy wymienić: Systems Engineering, Integrated Logistics Support (ILS), Capability Maturity Model Integration (CMMI), ISO 9000, Prince2, COBIT, Information Technology Infrastructure Library (ITIL), metodyki zarządzania cyklami życia produktu i oprogramowania.

Większość z wymienionych modeli prezentuje swoje własne, specyficzne dla danego modelu, podejście do zarządzania konfiguracjami.

Tradycyjny model zarządzania konfiguracjami został zbudowany wokół czterech predykatów:

1. **Identyfikowanie konfiguracji** (ang. configuration identification) – czyli możliwość określenia, które atrybuty definiują daną konfigurację, określenie zakresu obowiązywania danej konfiguracji (np. czas lub zasięg terytorialny)

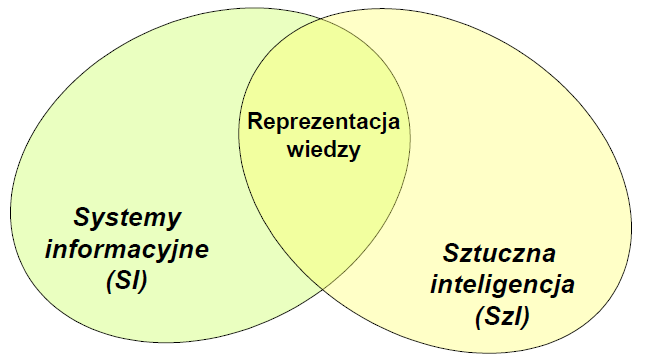
2. **Zarządzanie zmianami konfiguracyjnymi** (ang. configuration change control) – możliwość kontrolowanego, scentralizowanego zarządzania parametrami konfiguracyjnymi systemów

3. **Zarządzanie konfiguracjami historycznymi** (ang. configuration status accounting) – możliwość przeglądania, porównywania i analizowania zmian, które zachodziły w zarządzanym systemie

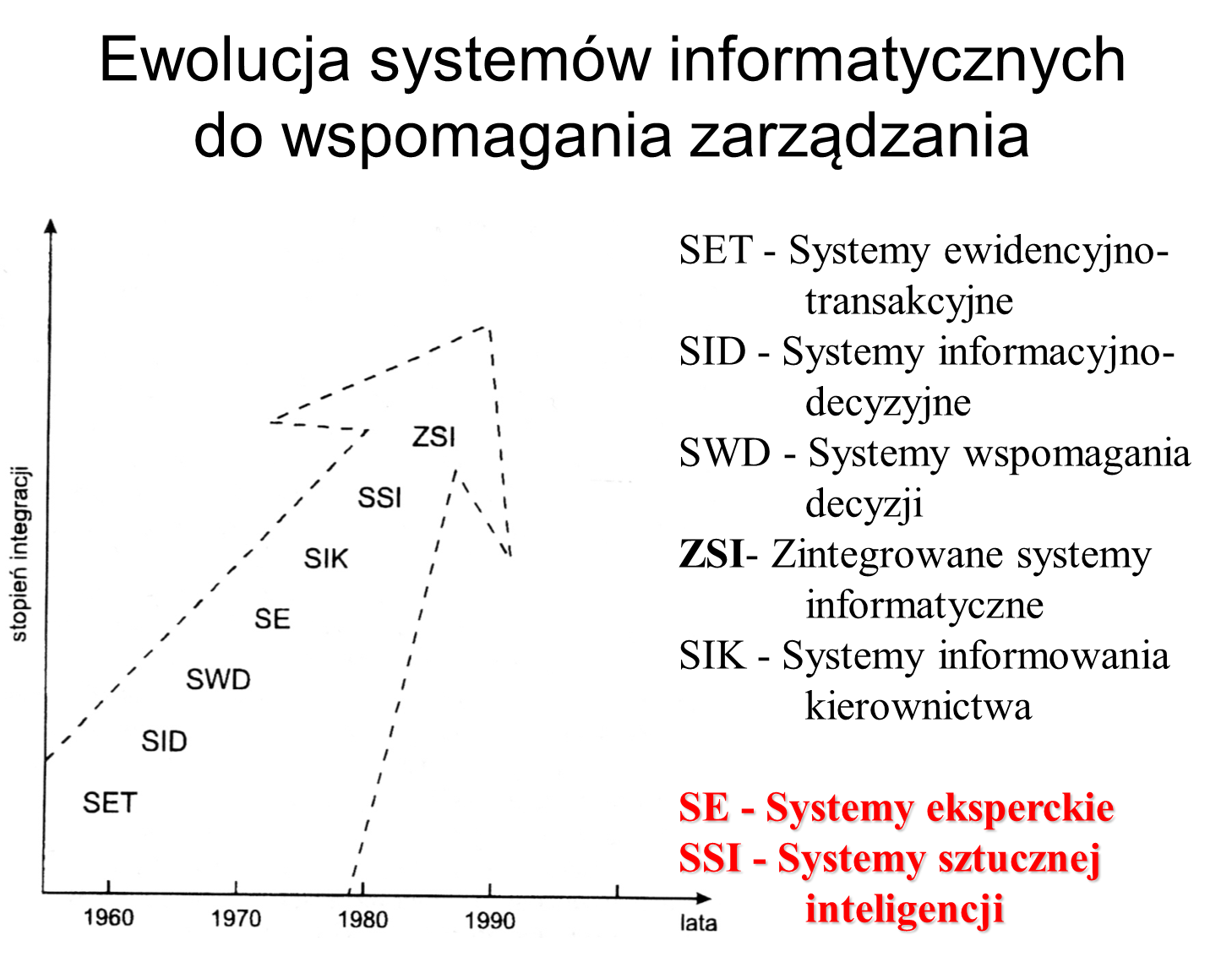
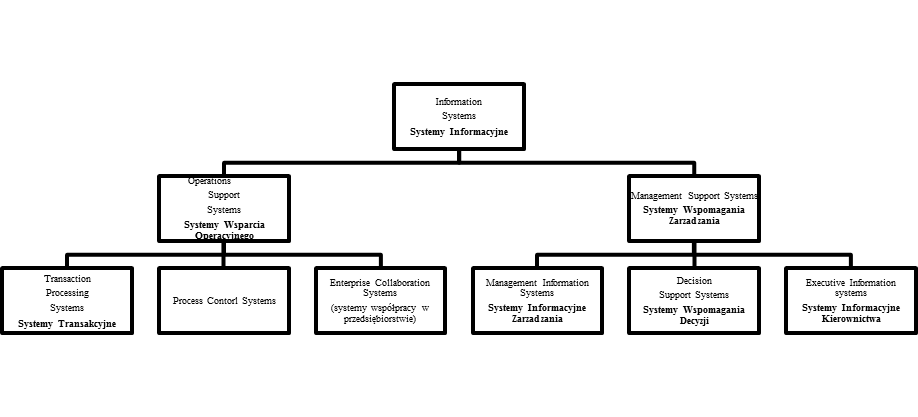
4. **Weryfikacja i przegląd konfiguracji** (ang. configuration verification and auditing) – rozbudowane mechanizmy kontrolowania zmian konfiguracyjnych oraz porównywania bieżących konfiguracji systemu z tymi, które w danym momencie uważamy za pożądane.

# 45. Zasady tworzenia harmonogramów realizacji systemu

# 46. Zastosowanie sztucznej inteligencji w systemach informacyjnych



Systemy informacyjne podzielić można na poszczególne typy:

**Tabela 1 Typy systemów informacyjnych**

Systemy informacyjno-decyzyjne – **SID** (*MIS – Management Information Systems*)

Są to systemy zapewniające firmie efektywne gromadzenie danych, organizacją ich przepływu  
 i sprawnego dostępu do danych z wykorzystaniem dużych systemów komputerowych.  
 Działają one w oparciu o bazy danych, które  
 w prosty sposób przetwarzają a wyniki prezentują  
 w postaci raportów. Przykładami są tutaj między innymi systemy: finansowo-księgowe, kadry-płace, gospodarka magazynowa.

Systemy wspomagania decyzji – **SWD**

(*DSS- Decision Support Systems*)

Są to systemy, których głównym zadaniem jest wspomaganie podejmowania decyzji strategicznych i taktycznych. W systemach tych zastosowano bazy metod, które ukierunkowane są na podejmowanie decyzji z częściowo lub słabo ustrukturalizowanymi problemami. system dostarczający informacji i wiedzy, wykorzystywany przy podejmowaniu decyzji, głównie przez kierownictwo średniego i wysokiego szczebla oraz analityków korporacyjnych. W efekcie wykorzystania systemów DSS uzyskujemy raporty i zestawienia, które dostarczane są kierownictwu w ramach systemów informowania kierownictwa (Executive Information Systems (EIS)). Dlatego też często systemy DSS określane są jako specjalizowana forma EIS.

Od końca lat 80. rozwija się też w różnych gałęziach przemysłu tzw. wysokiego ryzyka, medycynie i dla zarządzania zagrożeniami "inteligentne" DSS (IDSS) wykorzystujące technologie sztucznej inteligencji, systemy ekspertowe oraz modelowanie operacyjne i kognitywistyczne procesów decyzyjnych. Celem tych systemów jest zastępowanie lub wspomaganie złożonych, lecz już dobrze zdefiniowanych funkcji rozumowania.  
Do podstawowych obszarów wspomaganych przez te systemy zalicza się: planowanie działalności gospodarczej, inwestycje, zaopatrzenie, sprzedaż wyrobów i usług, gospodarka finansowa.

Zintegrowane systemy informatyczne – **ZSI** (*IMIS- Integrated Management Information Systems*)

W systemach tych wymagana jest realizacja kilku poziomów integracji:

§ Integracja systemu informacyjnego – czyli integracja funkcji, wyników przedsiębiorstwa, struktury organizacyjnej,

§ Integracja zastosowań - w tym integracja oprogramowania użytkowego, środków komunikacji z użytkownikami,

§ Integracja danych – rozumiana jako integracja  
 z bazą danych, słowników danych,

§ Integracja systemów – chodzi o systemy sieci, oprogramowanie komunikacyjne, oprogramowanie systemowe.

Systemy informowania kierownictwa – **SIK** (*EIS – Executive Information Systems*)

Są to systemy pozwalające skupić uwagę raczej na ogólnym, sprawnym działaniu firmy, niż na optymalizacji decyzji. Służą temu rozbudowane systemy zapytań oraz indywidualizacja przedstawionych raportów i narzędzi komunikacji z systemem.

Dostarczają informacji głównie kierownictwu najwyższego szczebla.

Systemy eksperckie – **SE** (*ES- Expert Systems*)

Określane są często jako komputerowe systemy rozwiązujące problemy z wykorzystaniem opisu (reprezentacji) wiedzy i procesu rozumowania. Systemy te generują swoje decyzje w oparciu o bazy wiedzy i mechanizmy sztucznej inteligencji. Dzięki temu mogą tworzyć różnorodne modele sytuacji decyzyjnej, uwidaczniać otrzymane rozwiązania  
 i objaśniać je. Do rozstrzygania problemu posługują się programami zawierającymi tzw. reguły heurystyczne, które odzwierciedlają wiedzę ekspertów dziedzinowych.

Systemy sztucznej inteligencji – **SSI** (*AIS – Artifical Intelligence Systems*)

Są to systemy uczące się na podstawie własnego doświadczenia. Podstawowymi narzędziami SSI są obecnie tzw. sieci neuronowe, które składają się ze sztucznych neuronów przetwarzających sygnały wejściowe w pojedynczy sygnał wyjścia. Zbiory połączonych neuronów tworzą sieć, której struktura i organizacja jest rezultatem uczenia się oraz gromadzenia doświadczeń.  
 Mogą wspomagać podejmowanie decyzji w wielu dziedzinach: usługi finansowe, marketing, analiza procesu produkcji itp.

W większości z nich zastosowanie znajdują mechanizmy sztucznej inteligencji. Takie jak:

* Systemy ekspertowe – systemy wykorzystujące bazę wiedzy (zapisaną w sposób deklaratywny) i mechanizmy wnioskowania do rozwiązywania problemów.
* Maszynowe tłumaczenie tekstów – systemy tłumaczące nie dorównują człowiekowi, robią intensywne postępy, nadają się szczególnie do tłumaczenia tekstów technicznych.
* Sieci neuronowe – stosowane z powodzeniem w wielu zastosowaniach łącznie z programowaniem "inteligentnych przeciwników" w grach komputerowych.
* Uczenie się maszyn – dział sztucznej inteligencji zajmujący się algorytmami potrafiącymi uczyć się podejmować decyzje bądź nabywać wiedzę.
* Eksploracja danych – omawia obszary, powiązanie z potrzebami informacyjnymi, pozyskiwaniem wiedzy, stosowane techniki analizy, oczekiwane rezultaty.
* Rozpoznawanie obrazów – stosowane są już programy rozpoznające osoby na podstawie zdjęcia twarzy lub rozpoznające automatycznie zadane obiekty na zdjęciach satelitarnych.
* Rozpoznawanie mowy i rozpoznawanie mówców – stosowane już powszechnie na skalę komercyjną.
* Rozpoznawanie pisma (OCR) – stosowane już masowo np. do automatycznego sortowania listów, oraz w elektronicznych notatnikach.
* W ekonomii, powszechnie stosuje się systemy automatycznie oceniające m.in. zdolność kredytową, profil najlepszych klientów, czy planujące kampanie reklamowe. Systemy te poddawane są wcześniej automatycznemu uczeniu na podstawie posiadanych danych (np. klientów banku, którzy regularnie spłacali kredyt i klientów, którzy mieli z tym problemy).

Więcej na: http://en.wikipedia.org/wiki/Applications\_of\_artificial\_intelligence

**ŹRÓDŁA:**

**[1] -** [**http://pl.wikipedia.org/wiki/System\_wspomagania\_decyzji**](http://pl.wikipedia.org/wiki/System_wspomagania_decyzji)

**[2] -** [**http://pl.wikipedia.org/wiki/Sztuczna\_inteligencja**](http://pl.wikipedia.org/wiki/Sztuczna_inteligencja)

**[3] -** [**http://www.uci.agh.edu.pl/uczelnia/tad//MBA/07-Typy%20system%f3w%20informacyjnych.ppt**](http://www.uci.agh.edu.pl/uczelnia/tad//MBA/07-Typy%20system%f3w%20informacyjnych.ppt)