

Zakres egzaminu dyplomowego

Specjalność - Systemy Informacyjne (SI)

Odpowiedzi na pytania wspólne zapożyczone z materiałów udostępnionych na forum przez Fenomena.
Dzięki!

- [1. Modelowanie a metamodelowanie.](#)
- [2. Własności i zakres zastosowań języków UML i LOTOS.](#)
- [3. Problemy transformacji i spójności modeli.](#)
- [4. Walidacja i weryfikacja modeli.](#)
- [5. Różnice między wyszukiwaniem informacji a wyszukiwaniem danych.](#)
- [6. Działanie systemu informacyjnego w sieci komputerowej.](#)
- [7. Technologie multimedialne stosowane w systemach informacyjnych.](#)
- [8. Efektywność systemów informacyjnych.](#)
- [9. Zadania projektowania sieci komputerowej.](#)
- [10. Klasyfikacja ruchu teleinformatycznego.](#)
- [11. Zarządzanie zasobami sieci komputerowej.](#)
- [12. Metody naprawiania błędów w systemach teleinformatycznych.](#)
- [13. Koncepcje dostarczania jakości usług w sieciach teleinformatycznych.](#)
- [14. Pojęcie systemu decyzyjnego oraz komputerowego systemu wspomaganie decyzji.](#)
- [15. Czynności techniki systemów.](#)
- [16. Problemy decyzyjne dla kompleksu operacji.](#)
- [17. Podstawowe problemy, metody i algorytmy optymalizacji dyskretniej.](#)
- [18. Podstawowe metody „obliczeń miękkich \(inteligentnych\)”.](#)
- [19. Podejmowanie decyzji w warunkach niepewności.](#)
- [20. Metody i algorytmy rozpoznawania.](#)
- [21. Postulaty metodologii nauk.](#)
- [22. Współczesne metody naukometrii.](#)
- [23. Algorytmy ewolucyjne w systemach informacyjnych.](#)
- [24. Jakość oprogramowania i jakość danych w systemach informacyjnych.](#)
- [25. Metoda COCOMO szacowania kosztów projektów informatycznych](#)
- [26. Metody automatycznej identyfikacji.](#)
- [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.](#)
- [32. Narzędzia typu CASE.](#)
- [33. Podejścia do zarządzania zespołami realizującymi projekty informatyczne.](#)

- [34. Podpis elektroniczny](#)
- [35. Specyficzne własności struktur baz danych w systemach informacyjnych.](#)
- [36. Standardy zapewnienia jakości oprogramowania.](#)
- [37. Strategie realizacji systemu informacyjnego.](#)
- [38. Systemy biometryczne.](#)
- [39. Systemy pośredniczące w wymianie danych.](#)
- [40. Techniki modelowania danych i przepływu danych.](#)
- [41. Tłumaczenie komputerowe tekstów w językach naturalnych.](#)
- [42. Własności sieci neuronowych.](#)
- [43. Zarządzanie ryzykiem w projekcie informatycznym.](#)
- [44. Zarządzanie zmianami i konfiguracjami oprogramowania.](#)
- [45. Zasady tworzenia harmonogramów realizacji systemu.](#)
- [46. Zastosowanie sztucznej inteligencji w systemach informacyjnych.](#)

1. Modelowanie a metamodelowanie.

Model

- Abstrakcyjna (uproszczona) wizja pewnego rzeczywistego lub wyobrażanego bytu
- Zależy od przyjętej perspektywy modelowania (wybrane istotne własności, zależy od celu modelowania)
- Abstrakcja systemu lub jego części
 - Role modelu:
 - Reprezentuje to co istnieje (dziedzinowe, biznesowe)
 - Reprezentuje to co ma powstać (projektowy)

MODEL	METAMODEL
wyrażenie, zestaw diagramów zapisanych w danym języku	model definiujący język, w którym jest wyrażony model
jest instancją metamodelu	
składa się z elementów modelowania	składa się z metaelementów modelowania

Modelowanie

- Proces budowy modeli

- Przykładowe języki modelowania (meta-modele, oparte na MOF):
 - UML – do modelowania różnego rodzaju systemów. Aby rozszerzyć UML lub dostosować go do bardziej specjalistycznej dziedziny problemu wprowadza się profile dla UML.
 - SysML (System Modeling Language) – rozszerzenie jednego profilu UML (UML Profile for Schedulability, Performance and Time Specification). Ma część wspólną z UML + rozszerza ją o dodatkowe elementy
 - SPEM (Software Process Engineering Metamodel) – do modelowania procesów wytwarzania oprogramowania
 - CWM (Common warehouse Metamodel)
 - BPMN (Business Process Modeling Notation) – do modelowania procesów biznesowych
 - ISO/IEC 24744 - meta-model używany do definiowania nowych metodyk
- P.S. Zakładam, że każdy umie coś powiedzieć o UML i modelowaniu więc się nie rozpisuje na ten temat :P

Metamodelowanie

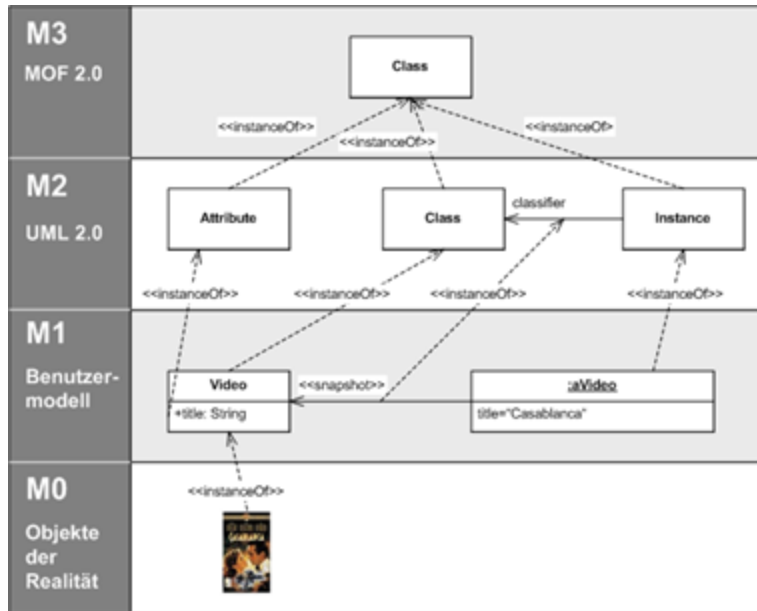
- Proces budowy meta-modeli
- oznacza konstruowanie zbioru "koncepcji" (obiektów, terminów, itp.) w zakresie pewnej dziedziny. Uznając model za abstrakcję pewnego zjawiska ze świata rzeczywistego, to meta-model jest abstrakcją ukazującą właściwości owego modelu.
- Do definiowania meta-modeli można wykorzystać np. standard MOF (Meta Object Facility) – który reprezentuje poziom meta-metamodelu.
- Meta-modele dla języków modelowania (meta-meta modele):
 - MOF (OMG)
 - ECore (Eclipse)
 - KM3 (grupa ATLAS/INIRA)
 - Kermeta (IRISA)
 - Microsoft ma rozwiązanie własne
- Powiązane tematy: MDA (temat 25. IO), DSL (temat 26 IO), transformacja modeli (temat 3), walidacja i weryfikacja modeli (temat 4) – Jak ktoś będzie miał czas to można płynnie przejść do tych tematów ;)

MOF – Meta Object Facility

- Standard OMG dla MDE (Model-driven engineering)
- architektura meta-modelowania, dostarcza środków do definiowania struktury lub atrakcyjnej składni dla języka lub danych.
- Składa się z 4 warstw. Na najwyższym poziomie (warstwa M3) znajduje się meta-meta model np. MOF 2.0. Meta-meta model warstwy M3 jest językiem do definiowania meta-modeli (warstwa M2) np. UML 2.0, SBVR. Meta-modele warstwy M2 służą do opisu elementów warstwy M1, czyli modeli np. modele UML. Ostatnią warstwą jest warstwa M0 – warstwa danych, która reprezentuje obiekty rzeczywiste. Architektura MOF została zobrazowana na rysunku 1. Każdy element w danej warstwie jest w relacji z elementem

znajdującym się w warstwie wyższej.

- Odgrywa taką samą rolę w definiowaniu meta-modeli jaką EBNF odgrywa przy definiowaniu gramatyki języków. MOF jest więc językiem DSL (DSL – opracowano w pyt. 26 dla IO) dla definiowania meta modeli, tak jak EBNF jest językiem DSL dla definiowania gramatyk.
- Może być używany do definiowania obiektowych meta-modeli jak np. UML, jak również meta-modeli nieobektowych np. Sieci Petriego, meta-model usług webowych.
- Standardem wspierającym MOF jest XML, który definiuje XML'owy format dla reprezentacji modeli warstwy M3, M3 i M1.



Rys.1 Architektura MOF

2. Własności i zakres zastosowań języków UML i LOTOS.

Własności UML:

- jest językiem półformalnym:
 - zdefiniowana formalnie składnia bezkontekstowa (za pomocą podzbioru UML)
 - zdefiniowana formalnie składnia kontekstowa (za pomocą OCL)
 - zdefiniowana nieformalnie (intencjonalnie) semantyka (w języku naturalnym)
- Graficzny język modelowania (specyfikowania, projektowania, dokumentowania)
- Jest standardem powszechnie akceptowanym, wykorzystywany do odniesień dla pojęć obiektowości
- Reprezentuje model w postaci zbioru diagramów, pełniących określone role

- Koncentruje się na danych i operacjach ich przetwarzania
- Reprezentuje zazwyczaj zbiory bytów, a nie pojedyncze byty

Zastosowanie UML:

Głównym przeznaczeniem UML jest budowa systemów informatycznych. Służy do modelowania dziedziny problemu - w przypadku stosowania go do analizy oraz do modelowania rzeczywistości, która ma dopiero powstać - tworzy się w nim głównie modele systemów informatycznych.

Własności LOTOS:

- Jest językiem interaktywnym, opisującym interakcje systemu z otoczeniem za pomocą komunikacji poprzez bramki (porty)
- Analityka nie interesuje wnętrze systemu (czarna skrzynka), a jedynie zmiany jego stanów
- Umożliwia modelowanie procesów w systemach zagnieżdżonych, gdzie jeden system zawiera w sobie kilka pomniejszych procesów
- Powstało kilka wersji języka: bazowa (bez uwzględnienia przesyłanych danych), pełna (uwzględnienie przesyłanych danych) oraz rozszerzona (upływ czasu)

Zastosowanie LOTOS:

Wykorzystany w tworzeniu systemów informatycznych, jako narzędzie pozwalające modelować systemy z wykorzystaniem algebry procesów – wyrażenie wymagań funkcjonalnych systemu w postaci sekwencji akcji.

3. Problemy transformacji i spójności modeli.

Transformacja modeli to opisanie modelowanego bytu (systemu) za pomocą innego modelu. Jej celem może być przekształcenie modelu do postaci mniej lub bardziej abstrakcyjnej, ograniczenie stopnia szczegółowości opisu (formalnego i nieformalnego) lub opisanie procesu / systemu / bytu z innej perspektywy celem wyszczególnienia pewnych procesów i problemów niewidocznych w opisie z wykorzystaniem innego modelu. Główny problem, jaki rodzi transformacja to możliwość utraty danych opisujących byt, która może być powodem zniekształcenia opisu (lub nawet jego zafałszowania). Transformacja może też prowadzić do niekontrolowanego rozrostu opisu / modelu, co również może być działaniem niepożądanym.

Oceną, czy model po transformacji odpowiada modelowi sprzed transformacji jest ocena ich

spójności, czyli na ile różne modele opisują ten sam system / byt.

Przykładem transformacji modeli może być przekształcenie diagramu UML na proces LOTOS lub opis z użyciem sieci Petriego.

4. Walidacja i weryfikacja modeli.

Walidacja jest to proces wyznaczania stopnia, w jakim model jest wiernym odzwierciedleniem rzeczywistego systemu z przyjętego punktu widzenia. Ma na celu określenie, czy symulacja daje wiarygodne wyniki, w założonym stopniu zgodne z odpowiedziami rzeczywistego systemu na takie same dane wejściowe. O ile dzięki weryfikacji projektant uzyskuje informacje o zgodności systemu symulacyjnego z jego założeniami, o tyle walidacja weryfikuje zgodność jego wizji z realnym światem. Obie te fazy wzajemnie się uzupełniają i jako takie czasami przedstawiane są wspólnie jako faza oceny adekwatności modelu.

5. Różnice między wyszukiwaniem informacji a wyszukiwaniem danych.

Zostawiam to, co niżej, ale na początek fragment z mojej pracy mgr (na podstawie (Baeza-Yates i Ribeiro-Neto 1999, 1-2) oraz (van Rijsbergen 1979, 1-2)) //IceMan

“Wyszukiwanie danych, będące pojęciem związanym przykładowo z bazami danych, ma na celu uzyskanie kompletnej odpowiedzi na zapytanie o pełnej specyfikacji względem strukturalizowanego zbioru danych. Inaczej sprawa ma się w przypadku wyszukiwania informacji, które skupia się na pozyskaniu z niestukturalizowanego (można nawet powiedzieć: chaotycznego) zbioru danych maksymalnie istotnych dokumentów/mediów przy zapytaniu sprecyzowanym niekoniecznie dokładnie i niekoniecznie właściwie.”

Czyli najprościej - wyszukiwanie danych mamy w SQLu, wyszukiwanie informacji mamy w Google.

... i niżej już inna interpretacja pytania, IMO niewłaściwa (dzwoni, ale nie w tym kościele, poniższa tematyka była przy piramidzie, gdzie były dane, informacje, wiedza, mądrość, nijak to ma się do samego wyszukiwania) //IceMan

Różnice między wyszukiwaniem informacji, a wyszukiwaniem danych wynikają bezpośrednio z różnic między informacjami, a danymi.

Dane (ang. data) — to litery, słowa, teksty, liczby, znaki, symbole, kombinacje liter, liczb, symboli i

znaków. Dane przetwarza się po to by otrzymać informację, wiadomość. Informacja jest produktem finalnym przetwarzania danych.

Istotną cechą danych jest brak uporządkowania; jest to zbiór nieuporządkowany. Ale jednocześnie dana jest wiarygodna i pewna (przy odpowiednim przechowywaniu). Dane w każdej chwili mogą być weryfikowane pod względem poprawności i aktualizowane. Dane pełnią także rolę nośników przepływu informacji.

Informacja jest zbiorem uporządkowanym wg określonego kryterium i poddanym interpretacji. Cechą każdej informacji (jako wyniku interpretacji) jest jej niepewność i ograniczona w czasie trwałość (wiarygodność) [4]. W popularnym pojmowaniu informacji zwracamy uwagę na ilość informacji (dużo, mało). Z informacją wiążemy także pojęcie jakości, na którą składa się wierność przekazu, wiarygodność, szybkość i sposób archiwizowania.

Z powyższych rozważań wynika, że dane i informacje różnią się własnościami. Okazuje się, że z "morza danych" możemy uzyskać niewiele informacji i odwrotnie — z niewielkiej ilości danych można otrzymać znaczące informacje. Tak więc ilość informacji nie jest zależna do ilości danych.

6. Działanie systemu informacyjnego w sieci komputerowej.

Rozwój technologii telekomunikacyjnych i komputerowych w ostatniej dekadzie, zwłaszcza niezwykle ekspansja WWW – stworzyły nowe możliwości rozpowszechniania i wymiany informacji, a tym samym przyczyniły się do powstania „nowoczesnych technik informacyjnych”. Sam System WWW, po wprowadzeniu standardów i metod pozwalających na opis i organizację multimedialnych danych, zawartych na stronach WWW i opracowaniu systemów wspomagających procesy wyszukiwawcze - ma szansę stać się globalnym systemem informacyjnym, efektywnie wykorzystywanym przez użytkowników końcowych do wyszukiwania informacji o wysokiej trafności.

Jedną z cech powstającego społeczeństwa informacyjnego jest rozwój pojęcia teleinformatyka, a więc niejako połączenia telekomunikacji i informatyki. Inaczej mówiąc są to sieciowe cyfrowe multimedia. Obecnie ważniejsze jest dotarcie do informacji niż jej gromadzenie, użytkownik ma możliwość bezpośredniego dostępu do informacji bez instytucji pośredniczących. Czas i przestrzeń dzięki połączeniom sieciowym nie odgrywają już takiej roli, jak niegdyś. Ważna jest także sprawa aktualności danych. W przypadku dokumentów drukowanych istnieje niebezpieczeństwo ich dezaktualizacji już w momencie opublikowania. Dokumenty elektroniczne w sieciach dostępne są w trybie czasu rzeczywistego. Poza tym istnieje możliwość aktualizowania danych na bieżąco.

7. Technologie multimedialne stosowane w systemach informacyjnych.

Koncepcja multimedialnych technologii w systemach informacyjnych zrodziła się na początku XXI wieku. Chęć możliwości posiadania dźwięku, obrazów oraz wideo w komputerach zrodziła ideę multimedialnych systemów informacyjnych. Złożoność multimedii determinuje ściśle określone warunki technologiczne systemów informatycznych (hardware). Wymagają one maszyn o dużej mocy obliczeniowej, aby możliwa była obsługa różnego rodzaju kodeków, multimedialnych systemów plików i odpowiadających im formatów plików. Niezbędne są dla tego typu systemów komputery wyposażone w nośniki pamięci o dużej pojemności, szybkich czasach dostępu i dużej prędkości transferu danych.

Ważnym przykładem systemu multimedialnego jest sieć www przenosząca poziom interakcji użytkowników z multimediami na bardzo wysoki poziom. Istotny rozwój sieci www następował wraz z kolejnymi erami (WEB 1.0, WEB 2.0, WEB 3.0) *[Możnaby tu pochorosiować teraz na temat każdej z er i jej wpływu na zmianę interakcji system-user, ale to chyba każdy już potrafi]* *[żeby tak bardzo nie chorosiować, a gadać konkrety, załączam krótką charakterystykę kolejnych]*

Cechy WEB 1.0:

- top-down (scentralizowany sposób tworzenia stron internetowych - stronę tworzył wyłącznie jej twórca, bez udziału społeczności internetowej)
- strony statyczne, zawierające materiały "tylko do odczytu"
- prawie brak elementów społecznościowych (co najwyżej książki gości)

Cechy WEB 2.0:

- Techniczne
 - wykorzystanie mechanizmu [wiki, blogów](#)
 - [ud](#)ostępnianie interfejsów XML, które umożliwiają innym stronom i programom korzystanie z danych Web 2.0 (przede wszystkim przez [RSS i Atom](#))
 - używanie nowych technologii, jak np. [AJAX](#) czy [Ruby on Rails](#)
 - Flash
- [Społeczne](#)[\[edytuj\]](#)
 - podejście bottom-up (zdecentralizowane): generowanie treści przez użytkowników ([user-generated content](#))
 - [tworzenie](#) się wokół serwisów rozbudowanych społeczności
 - możliwość nawiązywania kontaktów
 - wykorzystanie efektów sieciowych

- wykorzystanie kolektywnej inteligencji
- wykorzystanie otwartych licencji, jak [Creative Commons](#) czy [GNU GFDL](#)
- Wygląd
 - przejrzystość, prostota
 - pastelowe barwy
 - gradienty
 - zaokrąglenia
 - duże czcionki

WEB 3.0:

- Próba opisu przyszłości:
 - przetworzenie zawartości stron do wzorca czytanego przez różne (także nieprzeglądarkowe) aplikacje
 - wykorzystanie sztucznej inteligencji
 - wizualizacja danych w 3D

Ważne przykłady systemów multimedialnych do omówienia:

- e-learning - platformy e-learningowe etc.
- VOD - video on demand (można podać rys historyczny w postaci projektu DIVX z 1998 roku (patrz. przedmiot: Przetwarzanie Chorosioowego Wideo i obrazów))
- nośniki danych (od dyskietki po pendrive i płyty blue-ray)
- NVIDIA GeForce GRID - granie w chmurze
- Augmented reality w systemach mobilnych

8. Efektywność systemów informacyjnych.

Efektywność opisuje stopień wykorzystania zasobów sprzętowych i programowych stanowiących podstawę działania systemu informacyjnego.

Zła organizacja i niska efektywność systemu skutkują wzrostem kosztów działalności organizacji, spadkiem jakości obsługi, utrudnionym dostępem do informacji na różnych szczeblach, ich znaczną niekompletnością, wydłużonym czasem generowania.

Jedną ze strategii zwiększenia efektywności pracy zespołów projektujących systemy informacyjne jest wielokrotne używanie raz opracowanych fragmentów projektu lub modułów programowych. Ponowne użycie wcześniej opracowanego fragmentu projektu albo modułu programu ma wiele zalet, wśród których na plan pierwszy wysuwają się: obniżenie kosztów, przyspieszenie realizacji projektu oraz minimalizacja błędów

9. Zadania projektowania sieci komputerowej.

Warunki, które powinien spełniać dobry projekt sieci komputerowej:

1. Realizacja oczekiwań zleceniodawcy,
2. Fachowa dokumentacja,
3. Możliwość rekonfiguracji i rozbudowy sieci,
4. Łatwość rekonfiguracji w przypadku awarii,
5. Niezależność uszkodzeń w różnych segmentach sieci,
6. Bezpieczeństwo danych i serwerów.

Zasady:

1. Nie wolno osiągać granic możliwości sieci/sprzętu.
2. Maksymalna długość kabla dla 5 kategorii: 3m do komputera, 90m kabla poziomego, 6m kabla krosującego.
3. Zgodnie z ISO: 10 metrów kwadratowych na miejsce pracy (nie oznacza to 1 gniazdka na 10m², ale co najmniej 1 gniazdko na 10m²).
4. Każda kondygnacja musi być wyposażona w minimum 1 punkt dystrybucyjny. W przypadku pomieszczeń o powierzchni większej od 1000m² lub w przypadku, kiedy okablowanie poziome przekracza 90 m należy wprowadzić dodatkowy punkt dystrybucyjny.
5. Powierzchnia punktu dystrybucyjnego PD (SPD) w zależności od obsługiwanej powierzchni (S):
 - S 1000 m² -> SPD min. 3.0x3.4m,
 - S 800m² -> SPD min. 3.0x2.8m,
 - S 500m² -> SPD min. 3.0x2.3m.
6. Maksymalna dozwolona ścieżka sygnału obejmuje 5 segmentów kabla połączonych 4 hubami. W takim przypadku 2 z tych segmentów mogą być użyte wyłącznie jako połączenia między hubami. Jeśli się nie da inaczej, należy podzielić sieć na domeny kolizyjne (podsieci) i wprowadzić switchy.
7. Maksymalnie 1024 urządzenia na podsieć.
8. Maksymalna całkowita odległość w podsieci 500m.

10. Klasyfikacja ruchu

teleinformatycznego.

Ruchem telekomunikacyjnym nazywamy przepływ zgłoszeń, połączeń i wiadomości. W przypadku sieci pakietowych, w zależności od warstwy modelu OSI, na którym poziomie dokonujemy obserwacji ruchu, definicję tę można rozszerzyć na przepływ innych jednostek transmisji danych (pakietów – np. pakietów TCP, datagramów – np. datagramów IP, ramek – np. ramek Ethernet, komórek – np. komórek ATM (Asynchronous Transfer Mode), itd.). Każda aplikacja przesyłająca dane generuje w sieci pewien ruch. W przypadku sieci jednousługowej (np. sieci telefonii analogowej), ruch generowany przez poszczególnych użytkowników ma charakter homogeniczny.

W przypadku sieci wielousługowej (np. Internet) ruch telekomunikacyjny jest ruchem heterogenicznym. W sieci Internet składa się na niego m.in.:

- ruch generowany podczas transmisji danych masowych (np. z wykorzystaniem usługi ftp przesyłania plików) – charakteryzuje się on długim czasem trwania połączenia transportowego), związanym z przesyłaniem długich wiadomości (plików zawierających dane masowe), odstępy pomiędzy połączeniami transportowymi realizowanymi przez tego samego użytkownika są stosunkowo długie;
- ruch generowany podczas przesyłania poczty elektronicznej – charakteryzuje się on stosunkowo krótkimi wiadomościami tekstowymi (procentowy udział długich wiadomości, zawierających załączniki w postaci plików z danymi czy skompresowanych obrazów jest niewielki), przesyłanymi w bardzo dużych odstępach czasu;
- ruch generowany przez usługę zdalnego terminala (np. telnet), charakteryzujący się krótkimi wiadomościami, przesyłanymi w relatywnie krótkich odstępach czasu;
- ruch generowany przez usługę WWW charakteryzuje się krótkimi wiadomościami (tekst, niewielkie obrazy – np. logo firmy, małe elementy graficzne), przesyłanymi w bardzo krótkich odstępach czasu (związanymi z przesyłaniem zawartości strony WWW), po których następują relatywnie długie okresy braku aktywności (związane z przeglądaniem zawartości stron przez użytkownika); podobnie, jak w przypadku poczty elektronicznej, procentowy udział długich wiadomości jest niewielki;
- ruch generowany przez aplikacje realizujące transmisję informacji multimedialnej w czasie rzeczywistym – charakteryzuje się przesyłaniem bardzo krótkich (np. transmisja głosu) lub bardzo długich wiadomości (np. transmisja ramek wideo), lub ich złożenia (np. równoczesna transmisja głosu, obrazu, wiadomości tekstowych); cechą charakterystyczną źródła ruchu multimedialnego jest generowanie wiadomości w stałych odstępach czasu (np. generowanie ramek wideo co 40 ms lub co ok. 33 ms).

11. Zarządzanie zasobami sieci

komputerowej.

Typy sieci ze względu na zarządzanie zasobami:

Sieci równorzędne (każdy-z-każdym)

P2P (od ang. peer-to-peer – równy z równym) – model komunikacji w sieci komputerowej, który gwarantuje obydwu stronom równorzędne prawa (w przeciwieństwie do modelu klient-serwer). W sieciach P2P każdy komputer może jednocześnie pełnić zarówno funkcję klienta, jak i serwera. Każdy węzeł sieci (czyli komputer użytkownika) odgrywa rolę serwera przyjmując połączenia od innych użytkowników danej sieci, jak i klienta, łącząc się i pobierając dane z innych maszyn działających w tej samej sieci. Wymiana danych jest zawsze prowadzona bez pośrednictwa centralnego serwera.

Sieci oparte na serwerach (klient-serwer)

W sieciach klient-serwer zasoby często udostępniane gromadzone są w komputerach odrębnej warstwy, zwanych serwerami. Serwery zwykle nie mają użytkowników bezpośrednich. Są one raczej komputerami wielodostępnymi, które regulują udostępnianie swoich zasobów szerokiej rzeszy klientów. W sieciach tego typu zdjęty jest z klientów ciężar funkcjonowania jako serwery wobec innych klientów.

Narzędzia do zarządzania zasobami: procesory, pamięć, dyski, połączenia sieciowe itp.

Narzędzia pozwalające na równoważenie obciążenia w środowisku heterogenicznym systemów komputerowych: algorytmy rozdziału zasobów, języki i sposoby (narzędzia) specyfikacji zasobów, mapowanie zasobów do aplikacji. Funkcjonalność zarządzania zasobami jest ograniczona i charakteryzuje się takimi wadami jak:

1. niewykorzystanie zasobów obliczeniowych w czasie oczekiwania na wszystkie żądane zasoby;
2. konieczność upewnienia się, czy komponenty aplikacji nie rozpoczną działania zanim moduł przydzielający zasoby określi czy żądanie się powiedzie

12. Metody naprawiania błędów w systemach teleinformatycznych.

Do detekcji i korekty pojedynczych błędów transmisji stosuje się blokowe sekwencje znaków kontrolnych. Powszechnie stosowaną korekcją jest sekwencja BCC (Block Check Character) przedstawiająca znak lub sekwencję znaków generowaną przez algorytm kontrolny przed wysłaniem wiadomości w łączy transmisji danych. Urządzenie odbiorcze porównuje odtworzoną sekwencję kontrolną z sekwencją odebraną, aby stwierdzić, czy wystąpiły błędy transmisji.

Przy korekcji CRC blok informacyjny traktuje się jako wielomian, który w nadajniku dzieli się modulo 2 przez wielomian CRC, zwykle szesnastego stopnia (CCITT zaleca kilka, popularnym jest $x^{16}+x^{12}+x^5+1$). Otrzymana reszta tworzy 16-bitową sekwencję kontrolną FCS (Frame Check Sequence) transmitowaną na końcu bloku. W odbiorniku odebrany blok informacyjny również dzieli się przez taki sam wielomian. Przez porównanie otrzymanej reszty z dzielenia z odebraną sekwencją kontrolną można stwierdzić wystąpienie błędu transmisji. Brak zgodności sekwencji wymusza przesłanie odpowiedniej informacji kanałem sprzężenia powrotnego i retransmisję błędnych bloków.

13. Konceptcje dostarczania jakości usług w sieciach teleinformatycznych.

- kształtowanie i ograniczanie przepustowości
- zapewnienie sprawiedliwego dostępu do zasobów
- nadawanie odpowiednich priorytetów poszczególnym pakietom wędrującym przez sieć
- zarządzanie opóźnieniami w przesyłaniu danych
- zarządzanie buforowaniem nadmiarowych pakietów: DRR, WFQ, WRR
- określenie charakterystyki gubienia pakietów
- unikanie przeciążeń: Connection Admission Control (CAC),

14. Pojęcie systemu decyzyjnego oraz komputerowego systemu wspomagania decyzji.

- **Decyzja** jest świadomym (niełosowym) wyborem jednego z możliwych w danej sytuacji wariantów działania.
- **Sytuacja decyzyjna** charakteryzuje się istnieniem co najmniej dwóch możliwych wariantów działania różniących się między sobą stopniem korzyści.
- **Problem decyzyjny** to porównanie stanu oczekiwanego ze stanem rzeczywistym, pomiar odchyłeń, oraz przymus wyboru jednego z wariantów działania.
- **Decydent (DM; decision maker)** – podmiot podejmujący decyzję.
- **Przedmiot decyzji**– rzeczywistość (proces, system, obiekt) DP, której decyzja dotyczy; obiekt podejmowania decyzji - obiekt decyzyjny.
- **System decyzyjny** to struktura składająca się z obiektu decyzyjnego i wpływającego nań

decydenta.

Trzy wymogi systemu decyzyjnego (DS):

- Źródło informacji o DP (przedmiocie decyzji) - ekspert, hurtownia danych
 - informacja podstawowa (FI) - np. prawa fizyki
 - Obserwacja (O) - uzyskana w wyniku badania
- Opis przedmiotu, czyli model przedmiotu decyzji (DPM) lub ogólniej - reprezentacja wiedzy o przedmiocie (KR).
- System informatyczny, opracowany przez inżyniera oprogramowania, wyznaczający decyzję

Jeśli decyzja podlega akceptacji użytkownika, jest to system wspomaganie decyzji (czasem zwany systemem doradczym).

Systemy Wspomagania Decyzji są zorganizowanym zbiorem ludzi, procedur, baz danych i urządzeń wykorzystywanych w celu wspomaganie podejmowania decyzji na wszystkich etapach tego procesu, poczynając od rozpoznania czyli zdefiniowania problemu i zaklasyfikowania go do określonej grupy standardowej, następnie poprzez wybór odpowiednich danych stworzenie i analizę modelu informacyjnego opisującego rzeczywistość, dalej pomagając w generowaniu wariantów dopuszczalnych rozwiązań oraz w wyborze najlepszego rozwiązania.

Jakie decyzje potrzebują teorii?

Metody teorii decyzji wykorzystuje się wszędzie tam, gdzie podjęcie decyzji jest z pewnych powodów trudne. Przykładowo przyczynami mogą być:

- *duża liczba możliwych wariantów* – np. wybór najlepszego kandydata na dane stanowisko
- *skomplikowana sytuacja decyzyjna* – np. opracowanie takich tras i rozkładów jazdy autobusów, aby zapewnić wysoki poziom obsługi przy jak najniższym koszcie
- *możliwość wysokich korzyści lub dużych strat* – np. wybór sposobu ulokowania oszczędności
- *skomplikowany proces decyzyjny* – np. podejmowanie grupowych decyzji w dużych organizacjach
- *waga problemu decyzyjnego* – np. ustalenie okręgów wyborczych w wyborach prezydenckich

15. Czynności techniki systemów.

Teoria i technika systemów zajmuje się wspólnymi problemami, metodami i technikami dotyczącymi opisu, własności i sposobów rozwiązywania zadań, których przedmiotem są

systemy o różnej naturze. W szczególności t.s. zajmuje się:

- kreowaniem modeli i modelowaniem,
- identyfikacją i rozpoznawaniem,
- analizą i projektowaniem,
- sterowaniem (kierowaniem, zarządzaniem).

Identyfikacja systemów lub **procesów** to termin opisujący zespół metod i narzędzi i [algorytmy](#), które mają na celu zbudować dynamiczny model [systemu](#) lub [procesu](#) na podstawie danych pomiarowych zebranych z wejścia i wyjścia. Model taki może opisywać:

- właściwości wejściowo-wyjściowe systemu - jeżeli jest tworzony w oparciu o sekwencje sygnałów wejściowych i towarzyszące im sekwencje sygnałów wyjściowych,
- przebieg wyjścia systemu o wejściach pomiarowo niedostępnych - jeżeli jest tworzony jedynie w oparciu o mierzoną sekwencję sygnału wyjściowego.

Analiza - Przeciwną metodą do identyfikacji jest modelowanie analityczne. Polega ono na tym, że system dzielony jest na podsystemy, których właściwości oraz prawa fizyczne nimi rządzące dają się opisać modelami matematycznymi. Metoda ta jest zależna od skali problemu, może być bardzo czasochłonna i prowadzić do uzyskania modeli matematycznych zbyt skomplikowanych, by nadawały się do dalszego wykorzystania.

Sterowanie - Stworzony model pozwala na syntezę [układu regulacji](#) poprzez wprowadzenie [regulatora sterującego](#) danym [obiektem](#) lub procesem tak, by ten zachowywał się w pożądanym sposób.

Wydaje mi się, że niektóre pojęcia może wprowadzać wikipedia dla hasła: system

16. Problemy decyzyjne dla kompleksu operacji.

Kompleks operacji jest obiektem złożonym, którego elementami są operacje, powiązane ze sobą na zasadzie kolejności czasowych, to znaczy rozpoczęcie wykonywania niektórych operacji może się rozpocząć po zakończeniu wykonywania innych operacji. Graficznym sposobem przedstawienia kompleksu operacji może być graf. Operacje wykonuje się po to, aby osiągnąć pewien określony cel, wyrażony zadaniem. Oczywiście do wykonania operacji potrzebny jest realizator (zasób), który musi zostać przydzielony do danej operacji.

Podstawowym problemem decyzyjnym jest problem alokacji, który polega na rozdziale, czyli alokacji zasobów i zadań do operacji w kompleksie operacji. Jako kryterium alokacji zwykle przyjmuje się koszt lub czas wykonania kompleksu operacji.

Kolejnym problemem decyzyjnym jest problem szeregowania zadań. Problem ten jest szeroko znany i omawiany. Kluczowymi pojęciami występującymi w problematyce szeregowania są: zadanie, które wystąpiło już w poprzednim punkcie dotyczącym alokacji, oraz realizator rozumiany tu jako podmiot wykonujący zadanie i mogący mieć różną naturę i interpretację. Problem szeregowania można ogólnie określić jako wyznaczenie takiego dopuszczalnego przyporządkowania elementów jednego zbioru elementom drugiego zbioru, które jest najlepsze ze względu na przyjęte kryterium szeregowania.

Kolejny problem występuje w sytuacji w której nie mamy pewności co do zbiorów zasobów, zadań, operacji, czy parametrów tych elementów. Mamy do czynienia z tak zwanym problemem probabilistycznym (stochastycznym), w którym informacja o pewnych wielkościach jest określona z wykorzystaniem rozkładów prawdopodobieństwa i ma charakter stochastyczny.

Problem w sytuacji gdy mamy do czynienia z ruchomymi realizatorami. Występuje to w sytuacji gdy np. pracownicy na liniach produkcyjnych wykonują operacje na różnych liniach produkcyjnych w ramach jednego zadania. Wtedy do czasu realizacji zadania, dochodzi czas potrzebny do „przejścia” realizatora od jednego stanowiska do innego.

17. Podstawowe problemy, metody i algorytmy optymalizacji dyskretnej.

OPTYMALIZACJA DYSKRETNA

Większość deterministycznych problemów planowania i sterowania w dyskretnych systemach wytwarzania jest formułowana jako zagadnienia optymalizacji, w których wszystkie zmienne decyzyjne (bądź ich część) przyjmują wartości dyskretne, całkowitoliczbowe lub binarne. Zadania takie często nazywa się problemami optymalizacji dyskretnej lub dyskretno-ciągłej, należą do klasy problemów wyjątkowo kłopotliwych z obliczeniowego punktu widzenia. Zagadnienia te sprowadzają się do zadania minimalizacji funkcji celu $K(x)$ na zbiorze rozwiązań dopuszczalnych X , określonym przez zestaw warunków ograniczających. Głównymi powodami tych kłopotów są:

- częsty brak „klasycznych”, analitycznych własności (różniczkowalność, liniowość, itp.),
- wielo-ekstremalność ze znaczną liczbą ekstremów lokalnych.
- NP.-trudność większości problemów pochodzących z praktyki

- przekleństwo wielowymiarowości

Wielokrotnie, w celu uniknięcia kłopotów, próbuje się zamiast rozwiązywać problem dokładnie, wyznaczyć pewne jego rozwiązanie przybliżone. Dokładność tego przybliżenia posiada tendencję przeciwną do czasu obliczeń, tzn. uzyskanie dokładniejszego rozwiązania wymaga dłuższego czasu trwania algorytmu, przy czym ta ostatnia zależność posiada charakter silnie nieliniowy. Jest to czynnikiem powstania wielu rodzajów zarówno modeli jak i metod rozwiązywania, zwykle dedykowanych dla wąskich klas zagadnień. Często dla tego samego problemu NP.-trudnego występuje w literaturze kilka, kilkanaście różnych algorytmów o istotnie różnych cechach numerycznych.

Wprowadzenie dyskretnych zmiennych decyzyjnych umożliwia niekiedy względnie łatwe rozwiązanie zadań, w których zbiór rozwiązań dyskretnych nie jest zbiorem wypukłym lub też jest zbiorem niespójnym.

Metody rozwiązywania zadań optymalizacji dyskretnych można podzielić na dwie grupy:

- metody płaszczyzn tnących,
- metody heurystyczne.

Pierwsze dwie metody są metodami dokładnymi, metody heurystyczne dostarczają rozwiązań przybliżonych.

METODA PODZIAŁU I OGRANICZEŃ (branch & bound)

Do rozwiązywania dyskretnych zadań decyzyjnych stosuje się tzw. metodę podziału i ograniczeń. Idea metody polega na tym, że tzw. przegląd zupełny (pełny) zbioru ograniczeń D zastępujemy przeglądem ukierunkowanym. Pozwala to ocenić pośrednio pewne podzbiory rozwiązań i ewentualnie je odrzucić lub czasowo pominąć, bez utraty rozwiązania optymalnego, co znacznie przyspiesza uzyskanie rozwiązania

<http://optlab-server.sce.carleton.ca/POAnimations2007/BranchAndBound.html>

Metoda AHP http://www.mm.pl/~mmiszczyński/index/UL/Eksoc/BO2/WOD_1.pdf s.2

Analitical Hierarchy Process „szkoła amerykańska” – Thomas L. Saaty

Działania w metodzie AHP da się ująć w trzech etapach.

I. Budowa macierzy porównań parami dla n obiektów osobno w ramach każdego kryterium (macierze $(1) A$, $(2) A$, ..., $(K) A$) oraz dla samych kryteriów (macierz $(0) A$). Porównania te prowadzi do powstania $K+1$ macierzy porównań parami $((0) A, (1) A, (2) A, \dots, (K) A)$. Ważnym uzupełnieniem etapu I jest badanie spójności ocen decydenta.

II. Wyznaczanie rankingów indywidualnych dla każdej z macierzy etapu I.

III. Wyznaczanie rankingu wielokryteriowego n obiektów.

Metody przybliżone

Metoda przybliżona wyznacza pewne rozwiązanie bliskie rozwiązaniu dokładnemu. Metod przybliżonych jest zdecydowanie więcej niż dokładnych, zwykle są one problemowo-zorientowane. Zasadniczo, jakość metody przybliżonej jest oceniana z dwóch punktów widzenia: złożoność obliczeniowa algorytmu oraz dokładność przybliżenia. Dalsza charakterystyka bierze pod uwagę, między innymi, gwarancje zbieżności do rozwiązania optymalnego, szybkość tej zbieżności. Jakość wszystkich wymienionych ocen zależy od metody, problemu oraz konkretnych danych liczbowych podanych do algorytmu.

W ostatnich latach nastąpił burzliwy rozwój metod przybliżonych o dobrych i bardzo dobrych własnościach numerycznych potwierdzonych eksperymentalnie. Znaczna część tych metod czerpie swoje inspiracje z Natury i ma związek z dziedziną Sztucznej Inteligencji oraz Uczenia Maszynowego.

Rodzaje metod przybliżonych (heurystyk):

- algorytm zachłanny
- iteracyjne wspinanie się
- przeszukiwanie tabu

18. Podstawowe metody „obliczeń miękkich (inteligentnych)”.

- Alg. bazujące na naturze:

- **immunologiczne** – stanowią odpowiednik realizacji procesu adaptacji i dywersyfikacji naturalnego systemu immunologicznego. Ich zadaniem jest, poprzez sterowanie populacją przeciwciał, doprowadzenie do otrzymania rozwiązania. Algorytmy immunologiczne można podzielić na populacyjne (selekcja klonalna, selekcja negatywna i sieciowe (sieć idiotypowa)).

Algorytm *selekcji klonalnej* składa się z dwóch etapów: ekspansji klonalnej oraz hipermutacji. Pierwszy odpowiedzialny jest za wyselekcjonowanie najlepiej dopasowanych przeciwciał i ich sklonowanie. Hipermutacja natomiast realizuje dojrzewanie przeciwciał w celu jeszcze lepszego ich dopasowania.

Algorytm *selekcji negatywnej* stosuje się w celu wyeliminowania przeciwciał, które rozpoznają własne struktury jako obce. Algorytm ten znajduje zastosowanie najczęściej w problemach znajdowania anomalii.

Model *sieci idiotypowej* proponuje system immunologiczny charakteryzujący się dynamicznym działaniem nawet w przypadku braku antygenów ciał obcych. Model ten różni się od selekcji, czyniąc limfocyty zdolne do rozpoznawania siebie nawzajem. Znajdują one zastosowanie głównie w problemach rozpoznawania obrazów, analizie danych, maszynowym uczeniu i

problemach optymalizacji.

<http://www.mini.pw.edu.pl/~mandziuk/2010-05-26.pdf>

http://www.swo.ae.katowice.pl/_pdf/289.pdf

- **rojowe** (mrówkowe, pszczele, świetlikowe, kukułcze) - wywodzące się z algorytmu optymalizacji kolonii cząstek - particle swarm optimization PSO. W zależności od stworzenia, na którego obserwacjach bazuje algorytm, przeszukiwanie dziedziny i wybór najlepszych rozwiązań następuje w różny sposób, wszystkie mają jednak jedną bazę i w zapisie formalnym różnią się nieznacznie.

- **oparte o teorię chaosu** - bazujące na deterministycznych układach, tak wrażliwych na drobne zmiany warunków początkowych, że podczas obserwacji uchodzą za działające losowo (niedeterministycznie)

- **oparte o transformatę falkową** - będącą rozwinięciem transformaty Fouriera dla sygnałów niestacjonarnych, czyli zmieniających widmo w czasie. Transformata falkowa pozwala na wyznaczenie momentu występowania poszczególnych składowych harmonicznych.

- **Logika rozmyta typu 2** - wprowadzająca stany pośrednie w logice, umożliwiające określenie stopnia przynależności obiektu do zbioru.

Dzieli się na przedziałowe i uogólnione:

- przedziałowe: współczynniki przynależności są przedziałami ostrymi, umożliwiają modelowanie niepewności w przeciwieństwie do logiki rozmytej typu 1, a operacje and i or są proste i szybkie,

- uogólnione: współczynniki przynależności nie są ostre, należą do zbioru rozmytego. Podejście na razie raczej teoretyczne ze względu na słabo poznane zasady matematyczne i złożoność obliczeniową (zamiast operacji and i or - meet i join, bardzo wymagające obliczeniowo).

Przykładowo:

- typ-1: Karol jest w 0.72 wysoki,

- typ-2 przedziałowy: Karol jest w [0.62-0.82] wysoki, rozkład przynależności równomierny

- typ-2 uogólniony: Karol jest w $N(0.72, 0.1)$ wysoki, rozkład przynależności normalny (mam nadzieję że nie zakręciłem) czy jakkolwiek inny niż równomierny

- **Techniki agentowe** - system oparty o autonomiczne byty zwane agentami, rozproszone podejście do rozwiązania problemu.

Agent:

- oddziałuje na środowisko,
- komunikuje się z innymi agentami,
- działa realizując wyznaczone cele,
- ma dostęp i dysponuje zasobami,
- posiada jakiś zbiór umiejętności,
- posiada ograniczoną percepcję,
- posiada wiedzę nt środowiska lub ją gromadzi
- czasami może się rozmnażać (pewnie jak spotka ładną agentkę :))

- **Zbiory przybliżone** - rough sets, dla zbiorów o nieregularnych zakresach definiujemy przybliżenie górne i dolne o zakresach regularnych. Dzięki temu możemy określić nieostre pojęcie w ścisły sposób. Przynależność sprawdza się na podstawie klas równoważności R, zwanych atomami. Obiekty należące do tej samej klasy równoważności są nierozróżnialne.

- aproksymacja dolna: składa się z obiektów, które z całkowitą pewnością należą do zbioru X

- aproksymacja górna: składa się z obiektów, które MOGĄ należeć do zbioru X

- obszar brzegowy: różnica między aproksymacją dolną i górną

- zbiór dokładny (crisp): obszar brzegowy nie zawiera żadnych obiektów

- zbiór przybliżony (rough): obszar brzegowy zawiera jakieś obiekty

- **Hybrydy** - rozwiązania powstałe poprzez połączenie powyższych z innymi rozwiązaniami, np. rozmyte sieci neuronowe, transformata falkowa połączona z sieciami lub też połączenie między sobą itd.

- prócz tego:

- **sieci neuronowe (zagadnienie 42)**
- **algorytmy ewolucyjne**

Najczęściej działanie algorytmu przebiega następująco:

1. Losowana jest pewna populacja początkowa.
2. Populacja poddawana jest ocenie (**selekcja**). Najlepiej przystosowane osobniki biorą udział w procesie **reprodukcji**.
3. Genotypy wybranych osobników poddawane są operatorom ewolucyjnym:
 - a. są ze sobą kojarzone poprzez złączanie genotypów rodziców (**krzyżowanie**),
 - b. przeprowadzana jest **mutacja**, czyli wprowadzenie drobnych losowych zmian.
4. Rodzi się drugie (kolejne) **pokolenie**. Aby utrzymać stałą liczbę osobników w populacji te najlepsze (według funkcji oceniającej fenotyp) są powielane, a najgorsze usuwane. Jeżeli nie

znaleziono dostatecznie dobre rozwiązanie, algorytm powraca do kroku drugiego. W przeciwnym wypadku wybieramy najlepszego osobnika z populacji - jego genotyp to uzyskany wynik.

Działanie algorytmu genetycznego obejmuje kilka zagadnień potrzebnych do ustalenia:

1. ustalenie genomu jako reprezentanta wyniku
2. ustalenie funkcji przystosowania/dopasowania
3. ustalenie operatorów przeszukiwania

- **heurystyki**

Zasadnicza różnica między postępowaniem algorytmicznym a heurystycznym polega na tym, że pierwsze podejście zawsze daje rozwiązanie (choć czas oczekiwania na rozwiązanie może być nawet nieskończenie długi), podczas gdy podejście twórcze może być zawodne. Ze względu na to metody algorytmiczne stosowane są najczęściej w przypadku zbadanych, znanych już problemów, heurystyczne natomiast wszędzie tam, gdzie algorytmy nie wystarczają do rozwiązania zadania, gdzie wymagane są uzupełnienia, poszukiwane nowe metody i sposoby odnajdywania odpowiedzi czy rozwiązania zapytań.

Heurystyka informacyjna dotyczy tego, jak szybko i efektywnie wyszukać dokładnie tę informację, której użytkownik potrzebuje oraz tego, z jakich narzędzi, pamięci lub sprzętów służących do procesu poszukiwawczego będzie korzystał. Optymalne dotarcie do rozwiązania określa szybkość oraz cenę dostępu do właściwego wyniku, czyli odnalezienie dokumentów relewantnych przy minimalnej liczbie operacji w procesie wyszukiwania.

Dwie **naczelne zasady heurystyki informacyjnej** to:

1. zasada wyczerpania (kompletności)
2. zasada właściwego doboru materiału (relewantności)

Pożądany stopień trafności i kompletności zależy w dużej mierze od przeznaczenia wykorzystania informacji, tzn. do czego **informacja** jest w rzeczywistości potrzebna. Nie zawsze użytkownikowi zależy w jednakowym stopniu na osiągnięciu dużej trafności i kompletności wyszukiwania, tym bardziej, że podniesienie jednego wskaźnika powoduje z reguły obniżenie drugiego, tj. zwiększenie trafności obniża kompletność i odwrotnie. Przy ustalaniu zdolności potrzeb informacyjnych pamiętań należy, że istotną cechą relewantności jest jej subiektywny charakter, jest to jednak podstawowa cecha każdej informacji, która nie może istnieć bez odbiorcy.

- **sieci bayesowskie**

Formalnie taka sieć jest modelowana za pomocą **skierowanego grafu acyklicznego**, w którym wierzchołki reprezentują zdarzenia, a łuki związki przyczynowe pomiędzy tymi zdarzeniami. Jeśli od wierzchołka A prowadzi **ścieżka** do wierzchołka B to B jest potomkiem A. Podstawowym założeniem sieci bayesowskiej jest niezależność danego zdarzenia od wszystkich innych, które nie są jego potomkami.

<http://zsi.ii.us.edu.pl/~nowak/si/wyk4.pdf>

19. Podejmowanie decyzji w warunkach niepewności.

Niepewność jako pojęcie teorii decyzji oznacza sytuację, w której określone decyzje mogą

spowodować różne skutki, w zależności od tego, który z możliwych stanów rzeczy zajdzie, przy czym nie są znane prawdopodobieństwa wystąpienia poszczególnych z nich.

Typy niepewności: (Slajdy Kwaśnickiej)

1. Wejściowe fakty są niepewne lub mają przypisane prawdopodobieństwa
2. Reguły, nawet kiedy fakty są absolutnie pewne, generują nowe nowe fakty z pewnym stopniem ufności; np., IF organism is a gram positive coccus growing in chains (100% certainty) THAN organism is a streptococcus (with 70% certainty)
3. Kombinacje przypadków 1 i 2

Rachunek prawdopodobieństwa jest formalnym, poprawnym mechanizmem wnioskowania

Lecz:

Stosowanie rachunku prawdopodobieństwa wymaga od użytkownika dostarczenia pewnej liczby prawdopodobieństw warunkowych Nawet niezależne fakty początkowe nie propagują niezależności w procesie wnioskowania (patrz przykład)

Wnioskowanie z założeniem niezależności produkuje rygorystyczne (złe) wyjścia

Certainty Factor (CF) – Czynniki Pewności

$$CF(h) = MB(h) - MD(h)$$

(MB – Measure of Belief),

(MD – Measure of Disbelief)

0 MB 1; 0 MD 1; -1 CF 1

- Ważne cechy CF:
 - Przesuwa zaufanie produkowanych hipotez asymptotycznie do pewności kumulując miary otrzymywane z kolejnych reguł produkujących rozważane hipotezy.
 - CF jest symetryczną miarą, tzn., jest niezależny od uporządkowania zapalanych reguł
- Lecz:
 - $CF=0$ – niemożliwe jest rozróżnienie czy MB i MD są prawie takie same (konflikt) czy obie miary bliskie zeru (hipoteza nie może być potwierdzona lub zanegowana)

Logika rozmyta - wszystkim znana

Formalnie, **decyzjami podejmowanymi w warunkach niepewności** nazywamy taką klasę problemów decyzyjnych, w której dla przynajmniej jednej decyzji nie jest znany rozkład prawdopodobieństwa konsekwencji.

Przykład: Mamy pomysł na nowy produkt i chcemy zdecydować, czy otworzyć firmę zajmującą się produkcją i sprzedażą tego produktu. Nie jesteśmy w stanie określić prawdopodobieństwa sukcesu naszej firmy, jednak pomimo tego decydujemy się zaryzykować. Podjęliśmy decyzję w warunkach **niepewności**.

W praktyce prawie zawsze w wypadku niepewności określamy prawdopodobieństwo subiektywne zajścia danej konsekwencji.

Ze względu na posiadane informacje, możemy podzielić problemy decyzyjne na trzy grupy:

- **decyzja podejmowana w warunkach pewności** – każda decyzja pociąga za sobą określone, znane konsekwencje
- **decyzja podejmowana w warunkach ryzyka** – każda decyzja pociąga za sobą więcej niż jedną konsekwencję, znamy zbiór możliwych konsekwencji i prawdopodobieństwa ich wystąpienia
- **decyzja podejmowana w warunkach niepewności** – nie znamy prawdopodobieństw wystąpienia konsekwencji danej decyzji.

Metody podejmowania decyzji w warunkach niepewności:

- zbiory przybliżone
- sieci bayesowskie

Teoria zbiorów przybliżonych – Zbiór przybliżony (ang. *rough set*) to obiekt matematyczny zbudowany w oparciu o logikę trójwartościową. W swym pierwotnym ujęciu zbiór przybliżony to para klasycznych zbiorów: przybliżenie dolne i przybliżenie górne. Istnieje również odmiana zbioru przybliżonego, definiowana przez parę przybliżeń będących zbiorami rozmytymi (ang. *fuzzy set*). Dany element może należeć do obydwu przybliżeń, do żadnego lub tylko do przybliżenia górnego. Ten ostatni przypadek jest o tyle ciekawy, że pozwala na modelowanie niepewności.

sieci bayesowskie:

Sieć Bayesowska to acykliczny graf (DAG, Directed Acyclic Graph), składający się z: zbioru wierzchołków odpowiadających zbiorowi zmiennych zbioru skierowanych krawędzi łączących pary węzłów – intuicyjne znaczenie połączenia od węzła A do B oznacza, że A bezpośrednio wpływa na B

- graf nie zawiera cykli
- Węzły, od których dochodzą krawędzie do danego węzła to węzły rodzicielskie

- Każdy węzeł zawiera tabelę prawdopodobieństw warunkowych, określających wpływ węzłów 'rodzicielskich' na dany węzeł Cechy BN
- łatwiej ekspertom określić bezpośrednie zależności warunkowe w dziedzinie, niż podawać aktualne prawdopodobieństwa
- po zbudowaniu topologii BN, należy określić prawdopodobieństwa warunkowe dla węzłów, które są ze sobą bezpośrednio połączone; te prawdopodobieństwa są wykorzystywane do obliczania każdego innego prawdopodobieństwa

20. Metody i algorytmy rozpoznawania.

Rozpoznawanie obiektów – przypisanie do klasy na podstawie cech (algorytm rozpoznawania)

schemat: obiekt opisany cechami → pomiar cech → przetwarzanie cech → klasyfikacja

Rozpoznawanie w oparciu o ciąg uczący – rozpoznawanie z nauczycielem

Algorytmy odległościowe:

- **Algorytm NM** (najbliższej średniej) – obliczamy wartość średnią dla każdej klasy. Obliczamy odległość x od każdego elementu średniego, bierzemy min odległość i przypisujemy do klasy odpowiadającej tej wartości średniej

- **Algorytm KNN** (k najbliższych sąsiadów) – liczymy odległość x od każdego elementu ciągu uczącego, sortujemy w porządku niemalejącym, bierzemy k pierwszych pozycji i wybieramy najczęściej występującą klasę

Algorytmy probabilistyczne

- **Naiwny Bayes**

http://www.statsoft.pl/textbook/stathome_stat.html?http%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fstnaiveb.html

Drzewa klasyfikacyjne

algorytm (ogólny) - 2 kroki:

1) reguła podziału wierzchołków

2) reguła uznania wierzchołka za końcowy oraz wiążąca się z tym reguła przypisania wierzchołka

http://zsi.tech.us.edu.pl/~nowak/smad/SMAD_DT.pdf

Algorytm ID.3 (alg. budowy drzewa klasyfikacyjnego)

- 1) Dla każdej cechy liczymy entropię warunkową (w tym celu wyliczamy prawdopodobieństwa, że przykład przybierze daną wartość cechy oraz prawdopodobieństwa warunkowe takie że dla danej wartości cechy przykład będzie należał do klasy oraz dla danej wartości cechy przykład nie będzie należał do klasy)
- 2) Wybieramy cechę o najmniejszej wartości entropii i umieszczamy jako wierzchołek. Od wierzchołka odchodzą krawędzie etykietowane wartościami cechy (teraz będziemy brać pod uwagę tylko przykłady zawierające wartość cechy z tej krawędzi) itd.

<http://www.cvc.uab.es/~jbernal/Old%20Page/ID3%20COMPLETE%20EXAMPLE.pdf>

21. Postulaty metodologii nauk.

Metodologia nauk – nauka zajmująca się metodami stosowanymi przy formułowaniu twierdzeń i teorii naukowych. Metodologia nauk analizuje nie tylko procedury badawcze, lecz także jej wytwory: pojęcia, hipotezy, twierdzenia.

Metodologia nauk, w aspekcie pragmatycznym – nauka o metodach działalności naukowej i stosowanych w nauce procedurach badawczych; w aspekcie teoretycznym – nauka o elementach i strukturze systemów naukowych

- Metodologia nauk nie prowadzi badań empirycznych „w terenie”, nie obserwuje rzeczywistych, konkretnych naukowców przy pracy (A)
- Metodologia nauk pyta o procedury, schematy postępowania akceptowane w nauce w ogóle i w poszczególnych dyscyplinach naukowych (B)
 - jako źródła poznania, wiedzy
 - jako sposoby uzasadniania stwierdzeń, wiedzy
- Pytanie – skoro metodologia nie zajmuje się (A), to skąd właściwie wie jakie są (B)?
 - Nie wie, postępuje apriorycznie, jest działem logiki, zajmuje się metodami poprawnego myślenia w ogóle, m.in. bada wnioskowanie od zdania do zdania, mówi raczej o tym jakie powinny być właściwe metody prowadzące do wiedzy uzasadnionej (Bocheński, 1992, s. 20-25)
 - Odnosi się do nauki jako mega-faktu, rzeczywiście istniejącego (Bocheński, 1992, s. 139-142)

Postulaty:

- falsyfikowalność - jakkolwiek żadne doświadczenie nie może wykazać prawdziwości teorii, to istnieją doświadczenia, które mogą wykazać jej fałszywość
- sprawdzalność - wiedza naukowa poddaje się sprawdzeniu, kontroli; powinno być powiedziane w jaki sposób doszliśmy do takich, a nie innych stwierdzeń, wniosków, przy użyciu jakich metod, na podstawie jakich danych, a także jak inni mogą sprawdzić nasze wyniki
- prostota - wiedza naukowa powinna być zaprezentowana w sposób możliwie najprostszy, dostępny dla jak najszerzej grupy odbiorców (???)
- bezstronność - przeciwstawienie tendencyjności, obiektywna sprawdzalność, niezależność od podmiotu sprawdzającego

22. Współczesne metody naukometrii.

Naukometria (naukozn. dziedzina naukoznawstwa) - zajmuje się badaniem rozwoju nauki jako procesu informacyjnego; stosuje metody statystyczno-ilościowe (liczba publikacji, przyznanych stopni nauk. i nagród, placówek nauk.), pozwalające na określenie aktualnego stanu danej dyscypliny nauk. i prognozowanie perspektyw jej rozwoju.

Dużym wkładem w rozwój współczesnej naukometrii wykazało się czasopismo Journal Citation Reports zawierające informacje o czasopismach naukowych a także wskaźniki ich jakości. Czasopismo to, jak również wspomniane wskaźniki są częścią tzw. Science Citation Index powstałego w 1960 roku, posiadającego informacje nt. cytowań artykułów z największych czasopism naukowych.

Współcześnie do oceny jakości czasopism używa się następujących wskaźników:

- Impact factor - stosunek liczby cytowań do artykułów z czasopisma z ostatnich dwóch lat do liczby artykułów w tych latach, czyli średnia ilość cytowań do artykułów w ostatnim czasie
- Immediacy Index - stosunek liczby cytowań do artykułów z czasopisma z aktualnego roku do liczby artykułów opublikowanych w tym roku
- Half-Life cytujących - mediana wieku artykułów, które były cytowane przez artykuły w czasopiśmie (w danym roku)
- Half-Life cytowanych - mediana wieku artykułów z czasopisma, które były cytowane przez artykuły w innych czasopismach (w danym roku)
- liczba cytowań
- liczba artykułów

PS. Poniżej zamieściłem 3 różne miary stosowane w naukometrii: co prawda tekstu trochę jest,

ale chodzi o to żeby każdą z tych miar zrozumieć. Wydaje mi się, że ich zrozumienie jest wystarczające żeby mówić i mówić

Impact factor (IF) – w tłumaczeniu „Miara oddziaływania” – to wskaźnik prestiżu i siły oddziaływania czasopism naukowych, ustalany przez Instytut Filadelfijski (Institute of Scientific Information, obecnie części koncernu wydawniczego Thomson), na podstawie prowadzonego przez ten instytut indeksu cytowań publikacji naukowych.

Zazwyczaj jest tak, że jeśli dana publikacja wnosi coś istotnego do nauki to jest też często cytowana przez autorów innych publikacji. Stąd, liczba cytowań danej publikacji jest dobrą miarą jej wartości. Rozciągając ten tok rozumowania na czasopisma, można logicznie uznać, że średnia liczba cytowań wszystkich artykułów, które się w danym czasopiśmie ukazały jest dobrą miarą prestiżu i siły oddziaływania tego czasopisma.

IF jest ustalane wg wzoru:

$$IF = B/C$$

gdzie

B – to łączna lista cytowań które nastąpiły w danym roku kalendarzowym.

C – to liczba cytowalnych publikacji (zwykle nie są brane pod uwagę takie publikacje jak np. listy do redakcji), które ukazały się w danym czasopiśmie, w ciągu ostatnich dwóch lat.

Krytycy tego wskaźnika twierdzą, że:

- Jest on zanadto „mechaniczny” i często pokazuje nie tyle wartość naukową czasopism (i publikacji) lecz raczej aktualnie panujące mody i trendy w nauce. Czasopismo specjalizujące się w dziedzinie, która jest aktualnie modna siłą rzeczy publikuje artykuły, które opisują też „modne” badania. W modnych dziedzinach panuje większy ruch niż w niemodnych, co się przekłada bezpośrednio na liczbę cytowań.
- IF jest tworzony przez instytucję, która sama jest nastawiona na przynoszenie dochodu i ma ona w pewnym sensie „władzę” kreowania trendów w nauce – choćby poprzez arbitralne decyzje dopisywania lub wykreślenia czasopism ze swojej listy. Istnieje więc ryzyko, że trendy te są kreowane w taki sposób, aby obracało się to na korzyść samego Instytutu – na zasadzie samospełniających się proroctw.
- Bardzo często podnoszony jest argument, że IF preferuje badania, które są modne w USA. Istotnie ponad 50% czasopism na liście Instytutu Filadelfijskiego jest wydawana w USA, zaś ponad 80% czasopism na tej liście to czasopisma anglojęzyczne. Argument ten jest szczególnie istotny w naukach społecznych i ekonomicznych, których większa część odbywa się w innych obszarach językowych.

PIF - Wydaje się, że Przewidywalny Impact Factor (predicted impact factor) to „miara” przeznaczona dla czasopism, która już została zaindeksowana przez Thomson Reuters, ale nie

mają jeszcze wyznaczonego wskaźnika Impact Factor. W wyjaśnieniu (FAQ) do nowych zasad jest napisane:

Kto będzie obliczał index cytowań PIF (Przewidywalny Impact Factor)? PIF będzie obliczany automatycznie przez program po zasięgnięciu liczby cytowań czasopism naukowych umieszczonych w bazach Thomson Reuters Scientific.

ŹRÓDŁO: [FAQ](#)

Z tego można wyczytać (?), że czasopismo, które nie jest w tej bazie, nie może mieć obliczonego PIF. Ale nie dyskwalifikuje to czasopisma: może otrzymać punktację na „wykazie B”.

Index h - <http://www.ebib.info/2008/92/a.php?rek>

Według definicji wydawcy SCI Expanded, *indeks h* to liczba naturalna, określająca, ile spośród wyszukanych dokumentów według zadanego kryterium było tyle samo lub więcej razy cytowanych. Podany przykład $h=20$ mówi, że w badanym zbiorze jest 20 dokumentów - co najmniej 20 razy cytowanych.

23. Algorytmy ewolucyjne w systemach informacyjnych.

<http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm>

Algorytmy ewolucyjne to nazwa ogólna, obejmująca takie metody szczegółowe, jak np.:

- algorytmy genetyczne,
- programowanie genetyczne,
- strategie ewolucyjne.

Ich cechą wspólną jest wykorzystanie schematu działania wzorowanego na teorii doboru naturalnego i ewolucji. Jest to więc kolejna - po sieciach neuronowych - technika inspirowana analogiami biologicznymi. Główne pole zastosowań algorytmów (metod) ewolucyjnych to problemy optymalizacji - jak wiemy, wiele problemów praktycznych można przedstawić w języku problemów optymalizacyjnych; większość z nich można w związku z tym próbować rozwiązywać ewolucyjnie.

Zasadę działania algorytmów ewolucyjnych można nieformalnie streścić następująco: zamiast szukać na oślep (lub konstruować z kawałków, jak w przypadku strategii zachłannej) właściwego rozwiązania problemu, postaramy się je "wychodować". Niech różne rozwiązania konkurują ze sobą, krzyżują się, rozmnażają, naśladując mechanizmy odpowiadające za coraz

lepsze dostosowanie organizmów żywych do środowiska. Skoro ewolucja naturalna jest tak skuteczna w konstruowaniu złożonych, dobrze dostosowanych organizmów, to możemy liczyć na to, że ewolucja symulowana przyniesie nam dobre (zbliżone do optymalnych) rozwiązania problemu. Musimy tylko zadbać o to, by nasze symulowane "środowisko" promowało te osobniki (tzn. rozwiązania problemu), które lepiej spełniają nasze kryteria optymalizacyjne.

Podstawowe pojęcia

Osobnik - podstawowa jednostka podlegająca ewolucji. Zakładamy zwykle, że ów osobnik przebywa w pewnym środowisku, do którego może być lepiej lub gorzej przystosowany. "Celem" ewolucji jest stworzenie osobnika możliwie dobrze przystosowanego do danego środowiska. W przypadku algorytmów ewolucyjnych osobnikiem będzie przykładowe rozwiązanie zadania (punkt z przestrzeni stanów).

Populacja - zespół osobników zamieszkujących wspólne środowisko i konkurujących o jego zasoby.

Fenotyp - ujawniające się "na zewnątrz" cechy danego osobnika. W przypadku algorytmów ewolucyjnych są to te parametry (cechy) rozwiązania, które podlegają ocenie.

Genotyp - "plan konstrukcyjny", kompletny i jednoznaczny opis osobnika zawarty w jego genach. W przypadku algorytmów ewolucyjnych cechy rozwiązania (punktu z przestrzeni stanów) kodowane są w określony sposób, np. za pomocą ciągów binarnych ustalonej długości (odpowiednikiem genotypu osobnika jest w tym przypadku ciąg bitów).

Chromosom - miejsce przechowywania genotypu osobnika.

Kodowanie rozwiązań - sposób zapisania dowolnego dopuszczalnego rozwiązania problemu w postaci genotypu osobnika (np. ciągu bitów). Kodowanie musi zapewniać jednoznaczność dekodowania - każdemu genotypowi (np. każdej kombinacji bitów) musi odpowiadać pewne rozwiązanie zadania, czyli punkt z przestrzeni stanów. Musimy też się postarać, by każde rozwiązanie dało się zapisać w postaci genotypu.

Funkcja przystosowania (fitness) - funkcja pozwalająca dla danego osobnika określić jego jakość (z punktu widzenia rozwiązywanego problemu). Zakładamy, że jej wartości są rzeczywiste nieujemne oraz że wyższa wartość funkcji oznacza zawsze, że dany osobnik jest lepszy. W przypadku ewolucji naturalnej odpowiednikiem takiej funkcji jest ogólna ocena przystosowania osobnika do danego środowiska. W praktyce funkcja ta jest zwykle niewielką modyfikacją funkcji celu rozwiązywanego problemu.

Ewolucja przebiega według następujących ogólnych zasad:

- Genotyp danego osobnika ulega modyfikacjom podczas rozmnażania się. Zmiany te mogą wynikać albo z niewielkich, losowych mutacji, albo ze zmieszania (skrzyżowania) cech osobników rodzicielskich.
- Zmiany w genotypie powodują zmiany fenotypu osobników potomnych, co wpływa na stopień ich przystosowania do środowiska (podlega ocenie za pomocą funkcji celu).
- Zmiany w genotypie mają charakter przypadkowy. Zmiany korzystne dla osobnika zdarzają się równie często jak niekorzystne lub obojętne.

- Osobniki są oceniane poprzez porównanie ich przystosowania do danego środowiska. Te, które są lepiej przystosowane, mają większą szansę rozmnożyć się.
- Osobniki gorzej przystosowane przegrywają konkurencję o ograniczone zasoby środowiska i giną.
- Zmianom (mutacja, krzyżowanie) podlega genotyp osobnika, podczas gdy selekcji poddawane są fenotypy.

24. Jakość oprogramowania i jakość danych w systemach informacyjnych.

http://aragorn.pb.bialystok.pl/~mkret/Lectures/io_11_d.pdf

Jakość oprogramowania

- Zapewnienie jakości jest rozumiane jako zespół działań zmierzających do wytworzenia u wszystkich zainteresowanych przekonania, że dostarczony produkt właściwie realizuje swoje funkcje i odpowiada aktualnym wymaganiom i standardom
- Problem jakości, oprócz mierzalnych czynników technicznych, włącza dużą liczbę niemierzalnych obiektywnie czynników psychologicznych
- Podstawa obiektywnych wniosków co do jakości oprogramowania są pomiary pewnych parametrów użytkowych (niezawodności, szybkości, itd.) w realnym środowisku, np. przy użyciu metod statystycznych
- Obiektywne pomiary cech produktów programistycznych są utrudnione lub niemożliwe
- Jakość gotowych produktów programistycznych jest bardzo trudna do zmierzenia ze względu na ich złożoność, wieloaspektowość oraz niską przewidywalność wszystkich aspektów ich zastosowań w długim czasie

Zakres zapewnienia jakości

- Modele i miary służące ocenie kosztu i nakładu pracy
- Modele i miary wydajności ludzi
- Gromadzenie danych
- Modele i miary jakości
- Modele niezawodności
- Ocena i modelowanie wydajności oprogramowania
- Miary struktury i złożoności
- Ocena dojrzałości technologicznej
- Zarządzanie z wykorzystaniem metryk
- Ocena metod i narzędzi

Celem ostatecznym jest poprawa jakości oprogramowania

Trudności z oceną jakości oprogramowania

- Oceny jakości najczęściej muszą być znane zanim powstanie gotowy, działający produkt, co wyklucza zastosowanie obiektywnych metod pomiarowych
- Wiele czynników składających się na jakość produktu jest niemierzalna
- Produkty programistyczne mogą działać w różnych zastosowaniach, o różnej skali, przez co pomiary jakości mogą okazać się nieadekwatne przy zmianie skali (np. zwiększonej liczbie danych lub użytkowników), w innym środowisku, ...
- Pomiary mogą okazać się bardzo kosztowne, czasochłonne lub niewykonalne (z powodu niemożności stworzenia środowiska pomiarowego przed wdrożeniem);
- Nie ma zgody co do tego, w jaki sposób zmierzone cechy danego produktu składają się na syntetyczny wskaźnik jego jakości
- Oceny jakości produktów programistycznych są skazane na metody spekulacyjne, oparte na uproszczeniach oraz dodatkowych założeniach, algorytmach, wzorach i heurystykach

Zarządzanie przez jakość

- Koncepcja zarządzania przez jakość (TQM) wymyślona przez Japonczyka Eiji Toyode dla potrzeb naprawy japońskiego przemysłu motoryzacyjnego (~1950)
„Jakość jest najważniejszym kryterium oceny przydatności produktów dla klienta, a to właśnie klient umożliwia funkcjonowanie wytwórcy tych produktów”
- W związku z tym, że to klient stanowi o rentowności przedsiębiorstwa, to należy tak sterować wszystkimi fazami procesu produkcyjnego wyrobu, aby klient był zadowolony z jakości tego wyrobu
- Wniosek: producent wytwarzający produkty kiepskie powinien wypaść z rynku
- TQM została sformalizowana przez Amerykanów (W.E.Deming, P.Crosby, J.M.Juran, A.V.Feigenbaum), Japonczyków (E.Toyoda, M.Imai, K.Ishikawa) i Brytyjczyka J.Oaklanda

Więcej w zagadnieniu 36: Standardy zapewnienia jakości oprogramowania

http://www.ptzp.org.pl/zp/images/stories/zp_tekst_full/zp_2_2008/4%20Rostek.pdf

Jakość danych:

Kryteria jakości danych (jakie są dane jakościowe?):

- poprawność
- kompletność
- spójność
- aktualność

Zarządzanie jakością danych to zapewnienie zbiorowi danych wyżej wymienionych cech.

Sposoby zarządzania jakością danych:

- OODA (Observe-Orient-Decide-Act - obserwacja, orientacja, decyzja, akcja)
 - obserwacja - zdiagnozowanie defektów w danych w oparciu o błędy zgłaszane przez użytkowników i informacje z systemów weryfikacji danych
 - orientacja - klasyfikacja zdiagnozowanych w poprzednim etapie problemów oraz zaplanowanie działań zmierzających do ich rozwiązania, także opracowanie kosztów oraz korzyści wynikających z poszczególnych prac naprawczych w celu

- o ułatwienia opracowania poprawnego harmonogramu ich realizacji.
 - o decyzja - podjęcie decyzji o wdrożeniu opracowanych harmonogramów prac naprawczych
 - o akcja - początkowo wdrożenie pilotażowe w celu porównania osiągniętych wyników z zakładanymi, jeśli wyniki zadowalające - wdrożenie pełne
- 14 punktów jakości:
 1. Wprowadzenie stałych i klarownych zasad polityki zarządzania jakością informacji w całej organizacji,
 2. wykorzystanie filozofii zarządzania jakością informacji w zarządzaniu całym biznesem
 3. zaprzestanie lokalnego, fragmentarycznego poprawiania jakości informacji
 4. przy zakupie technologii informatycznych kierowanie się nie tylko ceną, ale i możliwością zarządzania jakością informacji,
 5. traktowanie poprawy jakości zarządzania informacji jako procesu ciągłego,
 6. szkolenie wszystkich pracowników w zakresie zarządzania jakością informacji,
 7. wprowadzenie przywództwa i pełnej odpowiedzialności kierownictwa w zarządzaniu jakością informacji,
 8. eliminacja kar za generowane błędy lub braki danych,
 9. efektywna współpraca pomiędzy poszczególnymi działami przedsiębiorstwa
 10. eliminacja haseł i sloganów o zarządzaniu jakością informacji na rzecz efektywnego działania,
 11. eliminacja planowania ilościowego na rzecz planowania jakościowego,
 12. eliminacja barier jakościowych na stanowisku pracy,
 13. wzmacnianie potrzeby edukacji i samodoskonalenia wśród pracowników,
 14. stopniowe, iteracyjne wdrażanie zarządzania jakością informacji.
- TIQM (Total Information Quality Management) - 6 etapów:
 1. Ocena definicji danych oraz architektury modeli danych (wyznaczenie mierników jakości danych uwzględniających potrzeby użytkowników)
 2. Ocena jakości informacji (definicja mierników jakości informacji takich jak szczegółowość, kompletność, aktualność, spójność, jednoznaczność)
 3. Ocena kosztów i ryzyka błędnych informacji (ocena konsekwencji korzystania z błędnych i niepełnych informacji - niezadowolenia klienta, spadku sprzedaży, utraty rynku)
 4. Projektowanie poprawy jakości danych
 5. Utrwalanie efektów poprawy jakości danych (przetestowanie planu na fragmentarycznym obszarze danych, jeśli pozytywne wyniki, kompletne wdrożenie)
 6. Tworzenie środowiska wysokiej jakości informacji (etap bez określonego początku i końca, długofalowy, przebiega równolegle do pozostałych etapów, ma na celu stopniowe wdrażanie 14 zasad zarządzania jakością informacji)

Przykłady błędów danych:

- niejednołitość wynikająca z własnej interpretacji danych w - przykładowo - różnych

działach przedsiębiorstwa

- niejednolite formatowanie (2012-06-20 lub 20 czerwca 2012 lub 20.06.2012, albo: J.Kowalski, Jan Kowalski, Kowalski Jan - to ta sama osoba, ale w bazie 3 różne wartości mimo, że występować powinna tylko jedna z nich - potrzebne są reguły zapisu danych!)
- błędy typograficzne:
 - pominięcie liter
 - przestawienie liter
 - błędy ortograficzne
 - zdrobnienia
 - brak pełnej nazwy
- problem zastępowania polskich znaków

25. Metoda COCOMO szacowania kosztów projektów informatycznych

Z "ZPI Trawinski 07-08 Szacowanie kosztów projektu":

Krótko o szacowaniu kosztów:

National Estimating Society zdefiniowało szacowanie kosztów jako:

- Sztukę aproksymowania prawdopodobnych kosztów projektu na podstawie informacji dostępnej w danym momencie

Szacowanie kosztów nie może:

- Być stosowane jak przepis z książki kucharskiej, musi być dostosowane do danego systemu
- Zastępować solidnych ocen, zarządzania i kontroli
- Dawać rezultatów lepszych aniżeli wejściowe dane
- Podejmować ostatecznych decyzji

Pomimo tych ograniczeń, szacowanie kosztów jest silnym mechanizmem, ponieważ :

- prowadzi do lepszego zrozumienia problemu,
- polepsza zrozumienie zarząd problemu alokacji zasobów,
- dostarcza obiektywnego odniesienia dla rozwoju miar.

COCOMO (ang. COnstructive COst MOdel) - model szacowania kosztów (empiryczny, oparty na doświadczeniu), pierwsza wersja z 1981 roku nosi nazwę COCOMO-81, aktualna wersja to COCOMO 2

- Model COCOMO 81 był dopasowany do klasycznego (kaskadowego) procesu projektowania produktów informatycznych od podstaw.
- Model COCOMO II uwzględnia nowe trendy na rynku budowy oprogramowania:
 - Nowe modele procesów: iteracyjne, przyrostowe, cykliczne, oparte na ponownym

- użyciu kodu, COTS, inżynierii wstecznej,
- Nowe narzędzia informatyczne: generatory aplikacji, języki obiektowe,
- Nowe metody pomiaru wielkości kodu: punkty funkcyjne, punkty obiektowe.

COCOMO-81 składał się z 3 modeli pozwalających oszacować całkowity koszt projektu na podstawie oszacowanej wielkości w postaci liczby linii kodu oraz modelu złożoności projektu.

- Model organiczny (ang. organic)
 - projekt wykonywany przez stosunkowo małe zespoły, złożone z osób o podobnych wysokich kwalifikacjach. Dziedzina jest dobrze znana. Projekt jest realizowany przy pomocy dobrze znanych metod i narzędzi.
- Model pół-oderwany (ang. semi-detached)
 - Członkowie zespołu różnią się stopniem zaawansowania. Pewne aspekty problemu nie są dobrze znane.
- Model wbudowany (ang. embedded)
 - Projekt realizuje system o bardzo złożonych wymaganiach. Dziedzina problemu, stosowane narzędzia i metody są w dużej mierze nieznane. Większość zespołu nie ma doświadczenia w realizacji podobnych zadań.

COCOMO 2 jest modelem trójpoziomowym który pozwala na szacowanie kosztów z wzrastającą dokładnością wraz z postępem cyklu rozwoju oprogramowania

- Poziom prototypowania (application composition model)
 - Faza prototypowania, poprzedzająca systematyczne modelowanie i projektowanie systemu. Występuje prototypowanie, wykorzystanie istniejących komponentów, podstawą oszacowań są punkty obiektowe
- 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

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 kreske.

RFID

RFID jest systemem identyfikacji radiowej. Co to oznacza? Przedmiotem jakiegokolwiek systemu identyfikacji radiowej jest przechowywanie pewnej ilości danych w wygodnych urządzeniach nadawczo-odbiorczych, których ogólną nazwą angielską jest tag (my będziemy to urządzenie określać w języku polskim jako znacznik), następnie odczytanie tych danych w sposób zautomatyzowany w odpowiednim (dogodnym) czasie i miejscu tak, aby uzyskać pożądaną skutek dla danej aplikacji. Zawarte w znaczniku informacje mogą opisywać poszczególne części na linii produkcyjnej, towary w czasie transportu, położenie przedmiotów, identyfikować pojazdy, zwierzęta lub osoby. Poprzez dołączenie do znacznika dodatkowych informacji można wzbogacić aplikację o możliwości wspomagające jej działanie z uwzględnieniem specyficznych informacji o przedmiocie, do którego przynależy znacznik.

Generalnie system RFID składa się zawsze z dwóch komponentów:

- znacznika umieszczonego na obiekcie identyfikowanym;
- czytnika, którego rolą jest odczytanie zawartych w znaczniku danych.

Schematycznie jest to przedstawione na poniższym rysunku:



Czytnik zawiera moduł transmisji radiowej (jest to zarówno nadajnik i odbiornik), układ sterowania oraz element łączący ze znacznikiem. Dodatkowo wiele czytników zawiera interfejs łączący z komputerem PC, pozwalający na przesyłanie z i do PC wszelakich danych

aplikacyjnych i systemowych.

Znacznik zawiera mikroukład scalony oraz element łącznikowy z czytnikiem. Znacznik jest nośnikiem danych systemu RFID.

Znacznik generalnie jest urządzeniem pasywnym, gdy znajduje się poza strefą oddziaływania czytnika. Aktywacja znacznika następuje z chwilą umieszczenia go w strefie oddziaływania czytnika. Energia wymagana do zasilenia znacznika jest przekazywana bezprzewodowo z czytnika poprzez element łącznikowy, antenę. W ten sam sposób przekazywany jest sygnał zegarowy - taktowanie oraz dane.

RFID ma kilka istotnych zalet:

- Znaczniki RFID mogą być odczytane znacznie szybciej niż kody kreskowe, a do tego nie jest istotne ich położenie względem czytnika,
- Znacznie większe możliwe odległości odczytu niż w przypadku kodów kreskowych,
- Możliwość zapisywania informacji na znacznikach RFID,
- Znaczniki są trudniejsze do zniszczenia lub usunięcia.

Optyczne rozpoznawanie znaków

Jest to technologia służąca do przetworzenia obrazu na dane cyfrowe. Początkowo możliwa była jedynie zamiana znaków zapisanych graficznie na odpowiadające im znaki alfanumeryczne. Obecnie technologia ta umożliwia rozpoznanie kroju pisma, formatowania tekstu, a nawet grafiki. Przetwarzane znaki są porównywane ze wzorcami zgromadzonymi w bazie danych, a w przypadku trudności z jednoznaczną identyfikacją znaku, program prosi użytkownika o odpowiedź. Technologie OCR radzą sobie dobrze z pismem drukarskim, jednak ciągle nie ma dobrych algorytmów do rozpoznawanie pisma odręcznego. Zastosowania OCR to m. in. przemysł, administracja oraz bankowość.

Technologie biometryczne

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

Identyfikacja głosowa polega na przetworzeniu dźwięku, który jest falą akustyczną na odpowiednie dane cyfrowe, a następnie porównaniu ich z przechowywanym wzorcem. Początkowo do tego celu wykorzystywane były specjalizowane procesory sygnałowe (ang. Digital Signal Procesor, DSP). Wraz z rozwojem techniki elektronicznej okazało się, że zadanie to można wykonać na zwykłych mikroprocesorach, co wpłynęło na znaczne obniżenie cen

rozwiązań wykorzystujących tą technologię.

Daktyloskopia jest działem kryminalistyki i służy do ustalania tożsamości człowieka na podstawie konfiguracji linii papilarnych. Analiza daktyloskopijna polega na porównaniu dwóch odbitek odcisków palców i stwierdzeniu, czy na obydwu występują identyczne cechy linii papilarnych. Daktyloskopia jest również szeroko wykorzystywana poza kryminalistyką, głównie do kontroli dostępu. W takim przypadku przy chronionym przejściu znajduje się czytnik linii papilarnych połączony z bazą danych, w której przechowywane są wzory odcisków palców osób uprawnionych do wejścia.

Karty inteligentne

Są to karty wielkości standartowej karty kredytowej posiadające zintegrowany układ scalony. Pierwszymi szeroko rozpowszechnionymi kartami inteligentnymi były wprowadzone we Francji w 1983 roku karty telefoniczne. W celu nawiązania połączenia z systemem, kartę należy umieścić w czytniku, gdzie dochodzi do kontaktu galwanicznego pomiędzy stykami czytnika i karty. W ten sposób do karty dostarczane są sygnał zegarowy oraz zasilanie. Wymiana danych odbywa się poprzez dwukierunkową magistralę szeregową. Budowę oraz działanie kontaktowych kart inteligentnych opisują normy ISO/IEC 7810 i ISO/IEC 7816.

Główną zaletą kart inteligentnych jest ochrona przechowywanych danych przed niepożądanym 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. Największą 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.

Najczęstsze zastosowania kart inteligentnych to: karty kredytowe lub debetowe, karty SIM do telefonów komórkowych, karty dostępu, karty pełniące rolę biletów komunikacyjnych.

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

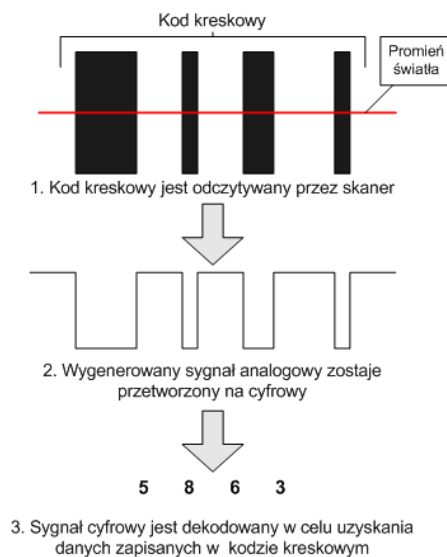
Karty pamięciowe wyposażone są zazwyczaj w pamięć typu EEPROM bądź ROM. Czytnik kart pamięciowych odczytuje zapisane dane, traktując kartę jako maszynę stanów. Dodatkowo można zastosować metody kryptologiczne w celu zwiększenia bezpieczeństwa danych przechowywanych w pamięci. Największą zaletą tego rodzaju kart jest ich stosunkowo niska cena.

Karty mikroprocesorowe jak sama nazwa wskazuje 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.

Kody kreskowe

Technologia ta polega na odczycie przy pomocy czytnika informacji zakodowanej graficznie na powierzchni przedmiotu.



Schemat odczytu kodu kreskowego.

Obecnie istnieje około 270 różnych symbolik kodów kreskowych, przy czym około 50 jest szeroko rozpowszechnionych. Każdą z symbolik można przypisać do jednej z trzech kategorii:

- Liniowe - symboliki tego rodzaju składają się z pionowych linii różnej szerokości, rozdzielonych białymi przestrzeniami,
- Dwuwymiarowe - mają one dużo większą pojemność danych, jeden kod dwuwymiarowy może pomieścić do 3,5 tys. znaków, gdy kod liniowy do 50 znaków,
- Trójwymiarowe (wypukłe) - jest to tak naprawdę kod liniowy, tylko zamiast czarnych i białych linii wykorzystuje wgłębienia i wypukłości.

Najpopularniejsze symboliki kodów liniowych to:

UPC (Universal Produkt Code), w odmianach UPC-A (składa się z 12 cyfr) oraz UPC-E (składa się z 7 cyfr).



EAN (European Article Numbering) - europejska wersja wykorzystywanego w Stanach Zjednoczonych kodu UPC. Również posiada 2 odmiany EAN-13 i EAN-8.

Kod EAN-13 zawiera 13 cyfr:

- Pierwsze trzy cyfry są numerem kraju. Polska przystępując do EAN International (obecnie GS1) otrzymała numer 590, zatem towary wytworzone przez producentów zarejestrowanych w Polsce będą miały prefiks 590.
- Kolejne cztery do siedmiu cyfr oznaczają numer producenta lub dystrybutora i przydzielane są przez organizację krajową podczas rejestracji.
- Następne dwie do pięciu cyfr stanowią numer produktu, który przydzielany jest przez samego producenta (jednostkę kodującą) dla asortymentu wyrobów o tej samej nazwie, cenie, wadze, pojemności, kolorze, wielkości, składzie, itp. Jeżeli inna seria towarów różni się którąkolwiek z tych cech konieczne jest nadanie innego numeru. Do obowiązków producenta należy pilnowanie, by ten sam numer produktu nie został przydzielony dwóm różnym produktom.
- Ostatnia cyfra jest cyfrą kontrolną wyliczaną według specjalnego algorytmu i służy do kontroli poprawności odczytu.



Kod EAN-8 zawiera osiem cyfr i składa się z numeru kraju, numeru produktu i cyfry kontrolnej. Numer produktu w tym przypadku jest przydzielany przez organizację krajową. Numer towaru w kodzie EAN-8 przydziela się tylko w uzasadnionych przypadkach, gdy nie ma żadnej możliwości umieszczenia na opakowaniu kodu EAN-13.



Szczegółowe informacje na temat kodu EAN można znaleźć na stronie internetowej GS 1 Polska www.gs1pl.org.

Code 39 - pozwala na zakodowanie 43 znaków w tym wszystkich dużych liter alfabetu i cyfr.



Code 128 - pozwala na zakodowanie 128 znaków ASCII.



Wśród kodów dwuwymiarowych można wyróżnić:

PDF417 (Portale Data Format) - opracowany przez Symbol Technologies, składa się z wielu kodów liniowych poukładanych jeden na drugim. Symbolika ta pozwala na kompresję i szyfrowanie danych, zawiera również mechanizmy wyszukiwania i korekcji błędów.



Data Matrix - jest to dwuwymiarowa symbolika macierzowa, pozwala na zapisanie do 3116 znaków ASCII w jednym kodzie.



QR Code - umożliwia zapisanie w jednym kodzie do 7089 cyfr, 4296 znaków alfanumerycznych lub 2953

bajtów danych.



MaxiCode - symbolika o stałym rozmiarze, składa się z 884 sześciennych modułów i pozwala na zapisanie maksimum 93 znaków alfanumerycznych lub 138 cyfr.



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

Z "ZPI Trawinski 05 Pomiary wielkości"

Metryki rozmiaru oprogramowania (s.16):

- Liczba linii kodu źródłowego
 - mierzona za pomocą SLOC (ang. Source Line Of Code) - liczby fizycznych lub logicznych linii programu źródłowego, zazwyczaj z pominięciem linii pustych i komentarzy
 - charakterystyka:
 - możliwość pomiaru dopiero po etapie kodowania (a to przecież stosuje się do szacowania, prognozy, a nie po utworzeniu gotowego projektu)
 - zależność od narzędzi programowych
 - zależność od jakości wykonania
 - paradoks produktywności (niższy poziom języka = więcej linii kodu = wyższa produktywność; inny przypadek: bardziej zwięzły kod = mniej linii kodu = mniejsza produktywność - szkodliwe dla programistów tworzących zwięzły kod)
- Liczba punktów funkcyjnych (FP - Functional Points)
 - metoda opiera się na charakterystyce funkcjonalnej systemu, obliczana na podstawie jego szczegółowej specyfikacji, nie bazuje na długości kodu
 - charakterystyka:
 - mierzona uniwersalna funkcjonalność zw. z przetwarzaniem danych
 - niezależna od technologii informatycznych (w pewnym zakresie) - od języków programowania i metodyki konstruowania programów.
 - niezależna od jakości wykonania
 - łatwo zastosować we wczesnych etapach projektu
 - niezależna od nakładu pracy potrzebnego na implementację danej funkcjonalności
 - proces pomiaru wymaga identyfikacji i klasyfikacji poszczególnych klas funkcjonalności
 - 5 kluczowych elementów w 2 kategoriach (s.26):
 - Przepływy danych
 - Wejścia danych (external inputs **EI**) - elementarne działanie, w którym dane przekraczają granicę systemu do jego wnętrza.
 - Wyjścia danych (external outputs **EO**) - elementarne działanie, w którym dane przekraczają granicę systemu na zewnątrz
 - Zapytania (external inquiries **EQ**) - elementarne działanie obejmujące operacje wejścia i wyjścia (bez przetwarzania danych, które są przesyłane do wewnątrz lub na zewnątrz)
 - Dane składowane
 - Wewn. zbiory danych (internal logical files **ILF**) - logicznie

powiązane zbiory danych znajdujące się całkowicie w granicach systemu, zasilane przez wejścia danych

- Zewn. zbiory danych (external interface files **EIF**) - logicznie powiązany zbiór danych całkowicie znajdujący się poza granicami systemu, dane takie można tylko odczytywać. Zewn. zbiór danych jest wewn. zbiorem innego systemu.
- Sposób liczenia:
 - identyfikacja przepływów danych i danych składowanych
 - określenie ich stopnia złożoności
 - przemnożenie poszczególnych elementów przez wagi złożoności i suma wszystkiego => powstaje nieunormowana liczba punktów funkcyjnych UFP (ang. Unadjusted Function Points)
 - wyznaczenie TCF (ang. Technical Complexity Factor) współczynnika złożoności technicznej i za jego pomocą korekta (normalizacja) liczby punktów funkcyjnych
- Punkty obiektowe (OP - Object Points)
 - stosowane tylko, gdy implementacja opiera się o języki 4GL
 - Oparte o:
 - liczbę wyświetlanych ekranów
 - liczbę generowanych raportów
 - liczbę modułów 3GL koniecznych do wykonania operacji na bazach danych
- Punkty aplikacji (AP - Application Points)
 - modyfikacja punktów obiektowych: dodano skale oceny do określenia stopnia złożoności każdej instancji obiektu (elementu) jako prosty, średni lub trudny.
 - z wyników eksperymentów równie skuteczne jak FP, ale czas ich szacowania jest 2x mniejszy
- Punkty przypadków użycia (UCP - Use Case Points)
 - Wymagane:
 - najważniejsze przypadki użycia
 - diagram przypadków użycia
 - wszystkie przypadki użycia i aktorzy o odpowiedniej klasyfikacji: proste, średnie, złożone
 - Oblicza się nieskorygowane punkty przypadków użycia UCCP (Unadjusted Use Case Points) = UAW (Unadjusted Actor Weights) + UUCW (Unadjusted Use Case Weights), następnie koryguje o współczynniki złożoności technicznej TCF (Technical Complexity Factor, takie jak łatwość instalacji, łatwość użycia, konieczność uzyskania rozproszenia, przenośności, dobrego czasu odpowiedzi itp.) i środowiskowy EF (Environmental Factor, takie jak trudność języka programowania, doświadczenie, zaznajomienie z projektem, motywacja, stabilność wymagań itp.)

28. Mobilność w systemach informacyjnych.

Mobilność to ogólny termin określający zdolność do używania technologii w ruchu (w przeciwieństwie do przenośności, gdzie urządzenia mogą być wykorzystywane dopiero po ich rozłożeniu w konfiguracji stacjonarnej).

Charakterystyka mobilności:

- ograniczone zasoby urządzeń mobilnych
- ograniczona energia
- obniżone bezpieczeństwo
- częste zmiany warunków pracy urządzeń mobilnych

Mobilność != rozproszoność

Aplikacje mobilne:

- wertykalne
 - służą wąskiej, niszowej domenie
 - przykłady: centrale (taksówki, policja, straż pożarna), śledzenie przesyłek (kurier, poczta)
 - łatwe do wdrożenia dzięki:
 - ograniczeniom i założeniom
 - ujednoliconym urządzeniom mobilnym
 - ograniczonej liczbie użytkowników
- horyzontalne
 - szerokie, masowe zastosowanie dla rozległego grona odbiorców
 - przykłady: e-mail, wiadomości, katalogi, biblioteki, wyszukiwanie informacji w pobliżu
 - są napędem dla badań dot. mobilności

Rodzaje sieci bezprzewodowych:

- komórkowe (GSM, TDMA, CDMA)
- sieci pagerów - rozległy zasięg, bardzo niska przepustowość
- satelitarne (GEOS, MEOS, LEOS) - rozległy zasięg, niska przepustowość, drogie
- Wireless LAN (802.11) - niski zasięg, relatywnie wysoka przepustowość

Charakterystyka bezprzewodowości:

- niska przepustowość
- spora zawodność
- częste rozłączenia - spodziewane lub niespodziewane
- asymetryczna komunikacja (występuje medium rozgłoszeniowe)

- droga (opłaty za połączenie, wiadomość, pakiet)

Charakterystyka przenośności:

- ograniczenia wynikające z pojemności baterii (konieczność wygaszania/usypiania poszczególnych elementów urządzenia mobilnego)
- ograniczenia zasobów (konieczność obniżania wielkości aplikacji oraz stopnia obciążenia przez nie urządzeń)
- małe ekrany
- zmiany lokalizacji
- heterogeniczność usług (zmienność, ograniczenia przepustowości)
- dane podawane w zależności od lokalizacji
- problemy z bezpieczeństwem
- ograniczenia interfejsu użytkownika

Istotne pojęcia/kwestie:

- Handoff - zmiana stacji bazowych przy przenoszeniu urządzenia między komórkami (obszarami zasięgu stacji bazowych), może być miękki (tymczasowo urządzenie podłączone do 2 stacji) lub twardy (odłączenie od pierwszej stacji, podłączenie do drugiej)
- Mobilne IP - w związku z przełączaniem się urządzenia między różnymi stacjami zmienia się jego rzeczywiste IP, aby do niego dotrzeć, ma ono dodatkowe IP stanowiące niezmienny identyfikator pozwalający na odnalezienie urządzenia mobilnego w sieci

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

<http://www.google.pl/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CFUQFjAA&url=http%3A%2F%2Fwww.wsiie.olsztyn.pl%2F~hanka%2Fwsiie%2Fpsi%2FModelowanie%2520funkcji%2520systemu.doc&ei=J4fTT8jeL8XLswaMzYiDw&usg=AFQjCNEFeelg54qGhXJeflx8-DCPqQvxRw&sig2=02sfhZ-y-0baQm1KQf29oA>

Techniki modelowania:

1. hierarchie funkcji - metodą modelowania funkcji, czyli Wymagania funkcjonalne opisują funkcje (czynności, operacje) wykonywane przez system. Prezentowany jest w postaci w

Diagramów Hierarchii Funkcji (ang. Function Hierarchy Diagram - FHD). Metoda ta jest bardzo szybka w realizacji, umożliwia analitykowi dokładne zdefiniowanie zakresu przyszłego systemu, który można łatwo kontrolować, uzgadniać oraz wykorzystywać w dalszej pracy np. przy tworzeniu diagramów przepływów danych i diagramów związków encji.

Konstruowanie hierarchii funkcji polega na tym, że:

- każda funkcja na diagramie określa, co system ma robić, a nie w jaki sposób;
- funkcje tworzą hierarchię - pierwsza funkcja ogólna opisuje główną funkcję systemu, dzieli się następnie na funkcje bardziej szczegółowe, dochodząc aż do funkcji elementarnych;
- każda funkcja nadrzędna powinna całkowicie opisywać funkcje szczegółowe oraz wszystkie funkcje szczegółowe muszą opisywać funkcję nadrzędną;
- wykorzystuje się metody z góry na dół (*top-down*) od ogółu do szczegółu;
- w celu przeprowadzenia weryfikacji diagramu wykorzystuje się metody od dołu do góry (*bottom-up*);
- każda funkcja powinna być opisana w języku naturalnym na tzw. formularzu opisu zawierającym: nazwę funkcji, opis, dane wejściowe, źródło danych wejściowych, wynik - rezultat wykonania funkcji, warunek wstępny - co musi być wcześniej zrealizowane, warunek końcowy - do czego wykorzystuje się efekty realizacji tej funkcji, cel realizacji funkcji.

2. Diagram Przepływu Danych

Jedną z metod wykorzystywanych na etapie analizy i projektowania służących do modelowania funkcji systemu jest **Diagram Przepływu Danych** (ang. Data Flow Diagram - DFD).

Przedstawia on, w jaki sposób dane przepływają w systemie oraz opisuje procesy przetwarzające dane. Tworzenie diagramu DFD opiera się na następujących kategoriach pojęciowych: proces, przepływ danych, magazyn danych, terminator i odpowiadających im symbolach graficznych. Szczegółowy opis wszystkich kategorii występujących na DFD zawiera słownik danych omawiany w podrozdziale 3.2. // w linku do dokumentu szczegółowo są omówione wymienione wyżej zagadnienia

3. Dekompozycja funkcji

Modelowanie funkcji systemu polega na stworzeniu hierarchicznej grupy diagramów DFD. Zazwyczaj pierwszy diagram DFD nazywa się diagram kontekstowy systemu albo diagram środowiska (otoczenia systemu).

Diagram kontekstowy (inaczej diagram DFD poziomu zerowego) jest specjalnym graficznym schematem przepływu danych, który definiuje zakres i granice systemu. System przedstawiony jest na diagramie jako pojedynczy proces powiązany bezpośrednio przepływami danych z terminatorami. Całość ma na celu przedstawienie powiązań systemu ze środowiskiem zewnętrznym.

Następnie tworzony jest **diagram systemowy**, ukazujący główne funkcje systemu. Każda funkcja jest przedstawiona w postaci hierarchicznej grupy diagramów niższego poziomu. Diagramy niższych poziomów to **diagramy szczegółowe**. Zbiór diagramów DFD dla systemu wraz z opisem elementów występujących na diagramie w słowniku danych stanowi model

funkcji systemu. Jest to graficzna mapa procesów ukazująca przepływ danych między procesami w systemie oraz między światem zewnętrznym a systemem.

30. Modelowanie struktur wymiany danych.

Przy analizie danych powszechnie stosowane są dwa podejścia: „od dołu do góry” i „od góry do dołu”

Analiza danych według podejścia „od góry do dołu” oznacza zastosowanie technik diagramowych odzwierciedlających to, co uważamy za obiekty zainteresowania przedsiębiorstwa i związki między tymi obiektami. Analiza „od góry do dołu” jest często uważana za modelowanie koncepcyjne, ponieważ jest to działanie na wysokim poziomie, często abstrakcyjne. Efektem końcowym takiego modelowania jest diagram związków encji, który w prosty sposób może być przekształcony w relacyjny schemat struktur tablicowych.

Analiza danych „od dołu do góry” nie zajmuje się abstrakcyjnymi pojęciami, lecz danymi konkretnymi. Aby wykonać analizę „od dołu do góry” musimy mieć pulę danych, uzyskanych np. przez badanie istniejącej dokumentacji przedsiębiorstwa. Do tej puli danych stosujemy szereg reguł. Podejście „od dołu do góry” w analizie danych nosi nazwę normalizacji.

Diagram związków obiektów

Model danych służy do wyrażenia struktury danych projektowanego lub istniejącego systemu. Określa on:

- typy danych występujących w systemie,
- wzajemne powiązania między danymi,
- ograniczenia nałożone na dane.

Podział modeli danych:

- Modele konceptualne – opisują dane za pomocą pojęć z których korzystają użytkownicy. Taki opis jest całkowicie niezależny od rozważań na temat pamięci, efektywności i innych szczegółów implementacyjnych.
- Modele fizyczne – opisują dane w kategoriach sposobu ich przechowywania w pamięci komputera i nie są zrozumiałe dla użytkowników.
- Modele implementacyjne - - operują pojęciami w zasadzie zrozumiałymi dla użytkowników, ale bliżej związanymi z konkretną strukturą (logiczną) danych w pamięci komputera. Modele :relacyjny sieciowy i hierarchiczny.

Podstawowe koncepcje i definicje:

- Encja jest rzeczą lub obiektem mającym dla nas znaczenie, rzeczywistym bądź wyobrażonym, o którym informacje muszą być znane.
- Każda encja musi być jednoznacznie identyfikowana – każda instancja (wystąpienie) musi być wyraźnie odróżniana od wszystkich innych instancji tego typu encji.
- Związek jest nazwanym istotnym powiązaniem istniejącym między dwiema encjami. Wyróżniamy związki: wiele do jeden, jeden do jeden, wiele do wiele; wymagane lub opcjonalne. Związki pomiędzy obiektami reprezentują powiązania świata rzeczywistego np. encja WYRÓB *składa się* z encji CZĘŚCI. Obiekt ZAMÓWIENIE *ma* wiele obiektów POZYCJA_ZAMÓWIENIA.
- Atrybut jest dowolnym szczegółem służącym do kwalifikowania, identyfikowania, klasyfikowania, określania ilości lub wyrażania stanu encji – jest dowolnym opisem mającym znaczenie dla encji. Musi opisywać encję przy której występuje. (Atrybut jest funkcją, przypisującą obiektowi wartość cechy ze zbioru wartości tej cechy, czyli z dziedziny).
- Aby każdą instancję encji można było odróżnić od wszystkich innych instancji tego typu encji, każda musi być jednoznacznie identyfikowalna. Jednoznacznym identyfikatorem może być atrybut, kombinacja atrybutów, kombinacja związków, lub kombinacja atrybutów i związków.

Zasady tworzenia diagramów związków encji

Model danych systemu zbudowany na podstawie modelu encja - atrybut –związek reprezentuje wszelkie obiekty oraz powiązania między nimi istotne z punktu widzenia funkcjonowania systemu.

Pięć kroków budowy modelu danych:

1. Identyfikacja (wydzielenie) zbioru typów obiektów wraz z ich atrybutami kluczowymi;
2. Identyfikacja powiązań bezpośrednich między obiektami (ewentualnie z wykorzystaniem tablicy krzyżowej powiązań) oraz ich rodzaju;
3. Przekształcenie tablicy krzyżowej powiązań w pojęciowy model danych i identyfikacja pozostałych atrybutów obiektów.
4. Przekształcenie każdego z powiązań typu wiele do wiele na dwa powiązania jeden do wiele i identyfikacja dodatkowych atrybutów charakterystycznych dla nowo powstałych obiektów.
5. Sprawdzenie poprawności otrzymanej struktury poprzez porównanie z wymaganiami systemu.

----- Inna definicja (nieco łatwiejsza ale o niebo bardziej lakoniczna) -----

Diagram związków encji (z ang. ERD - entity relationship diagram) jest jednym ze sposobów przedstawienia modelu logicznego bazy danych. Jest to rysunek encji i związków pomiędzy nimi, który ułatwia analizę wybranego obszaru świata rzeczywistego modelowanego w bazie danych. Do wspomagania procesu projektowania systemu informatycznego służy m.in. pakiet aplikacji Oracle o nazwie Designer/2000.

----- Tutaj przedstawiłem całe opracowanie na ten temat ... -----

ERD (Entity Relationship Diagram) – diagram związków encji, jest to model danych służący do wyrażenia struktury danych projektowanego lub istniejącego systemu. Określa on typy danych występujących w systemie, wzajemne powiązania między danymi i ograniczenia nałożone na dane.

Podstawowymi pojęciami tego modelu są: encja, atrybut i związek.

Komponent	Opis
Encja	<p>Rzecz mająca znaczenie, rzeczywista lub wymyślona, o której informacje należy znać lub przechowywać. Jednoznacznie identyfikowalny składnik badanej rzeczywistości, o którym informacja jest lub może być zbierana i przechowywana. Przykładami encji są: PRACOWNIK, KLIENT, DOSTAWCA, ZAMÓWIENIE, MAGAZYN, FAKTURA, POZYCJA, PRZECENA, KONTO. Encja jest urzeczywistniona poprzez wystąpienie. Przykładowo, wystąpieniami encji KLIENT są: <i>Nowak, Dobrowolski, Kwiatkowski</i> itd.</p> <p>Uwaga: encje opisuje się za pomocą rzeczowników lub wyrażen rzeczownikowych w liczbie pojedynczej..</p>
Atrybut	<p>Cecha, element charakteryzujący encje i związki w badanej dziedzinie przedmiotowej. Atrybut ma jedno z pięciu zadań: identyfikować, opisywać, klasyfikować, określać ilość lub wyrażać stan encji.</p> <p>Zestaw atrybutów, który jednoznacznie opisuje encję, nazywa się wiązką atrybutów. Wiązka powinna składać się, z co najmniej dwóch atrybutów opisujących daną encję. Szczególną rolę z zakresu atrybutów encji pełni klucz, zwany identyfikatorem. Pozwala on na jednoznaczne określenie wystąpienia encji. Jeśli używa się jednego atrybutu dla określenia encji, to mamy do czynienia z kluczem pojedynczym, a jeśli w tym celu używa się więcej niż jednego atrybutu, to z kluczem złożonym. Znak kratki (# - hash) z lewej strony atrybutu oznacza, że ten atrybut jest unikalnym identyfikatorem dla encji lub wchodzi w jego skład.</p> <p>Wymagania dla atrybutu można przedstawić następująco: nazwa atrybutu musi być unikalna w ramach encji; atrybut musi być obowiązkowy (tzn., że wartość atrybutu musi być zawsze</p>

	<p>określona) lub opcjonalny (tzn., że atrybut nie musi mieć wartości). Symbolu „*” używa się dla atrybutu obowiązkowego, zaś symbolu „o” dla opcjonalnego; atrybut musi mieć format lub typ;</p> <p>Z każdym atrybutem związana jest dziedzina atrybutu, stanowiąca zbiór wartości, które są jemu podporządkowane oraz zbiór reguł walidacji. Atrybut jest, więc funkcją, wiążącą encje lub związki z dziedzinami wartości. Dana dziedzina może przybrać postać wartości (np. z przedziału liczb naturalnych), skończonego zestawu wartości (np. rodzaje stawek podatku VAT: zw, 0%, 7%, 12%, 22%) lub wartości binarnych (np. T lub N, O lub 1). Każdemu atrybutowi w danym wystąpieniu encji lub związku przyporządkowaną jest tylko jedna wartość z jego dziedziny.</p>
Związek	<p>Związek stanowi naturalne powiązanie pomiędzy dwoma lub więcej encjami w badanej dziedzinie przedmiotowej. W identyfikowaniu i modelowaniu związków encji bierze się pod uwagę następujące cechy: stopień związku (lub liczebność związku) i opcjonalność (lub uczestnictwo encji).</p>
Przykład prostego diagramu związków encji	<p>The diagram illustrates a 1:1 relationship between two entities: 'KLIENT' and 'FAKTURA'. 'KLIENT' is represented by a rectangle with a stippled pattern and contains three attributes: 'Nazwa' (marked with an asterisk *), 'Adres' (marked with an asterisk *), and 'e_mail' (marked with a circle o). 'FAKTURA' is represented by a solid grey rectangle. A horizontal line connects the two entities, with a double vertical bar (mandatory one) at the 'KLIENT' end and a vertical bar with a crow's foot symbol (mandatory many) at the 'FAKTURA' end. Three arrows point upwards towards the diagram: one to the 'KLIENT' entity labeled 'Atrybuty', one to the relationship line labeled 'Związek', and one to the 'FAKTURA' entity labeled 'Encja'.</p>

Stopień związku – oznacza stosunek ilościowy między liczebnością wystąpień poszczególnych encji, uczestniczących w danym związku. Stopień związku mówi o tym, ile wystąpień encji jednego rodzaju jest powiązanych z iloma wystąpieniami encji innego rodzaju.

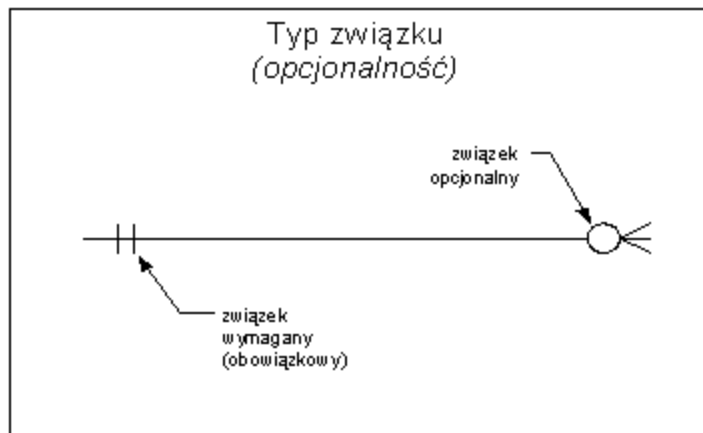
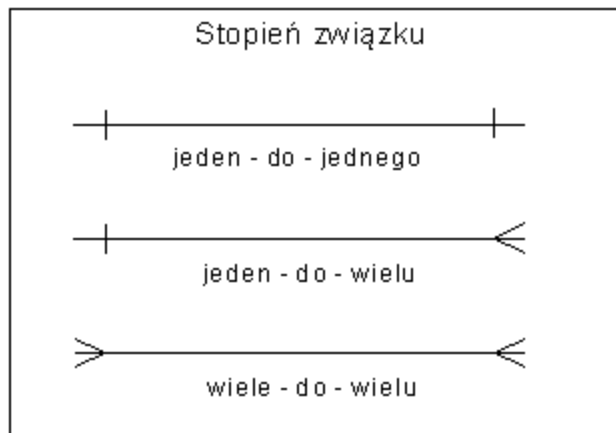
Stopień związku	Przykład	Znaczenie
1:1	Dziekan-Wydział	Każde wystąpienie encji Dziekan jest powiązane tylko z jednym wystąpieniem encji Wydział . Zatem jeden Dziekan kieruje jednym Wydziałem

1:M 1: wiele	Wydział-Student	Każde wystąpienie encji Wydział powiązane jest jednym lub wieloma wystąpieniami encji Student , przy czym każde wystąpienie encji Student powiązane jest tylko jednym wystąpieniem encji Wydział . Zatem Wydział posiada wielu Studentów , natomiast Student studiuje wyłącznie na jednym Wydziale
M:N Wiele : wiele	Książka - Autor	Każde wystąpienie encji Książka powiązane jest z wieloma wystąpieniami encji Autor i odwrotnie każde wystąpienie encji Autor powiązane jest z wieloma wystąpieniami encji Książka . Jest to sytuacja, gdzie Książka może być napisana przez jednego lub wielu autorów i jeden Autor jest podpisany pod jednym lub wieloma tytułami Książek .




Opcjonalność dotyczy zaangażowania encji w związek. Z uwagi na tę cechę wyróżnia się dwa typy związków:

wymagane (obowiązkowe) – zachodzi wówczas, jeśli wszystkie wystąpienia encji muszą uczestniczyć w związku;

opcjonalne - zachodzi wówczas, jeśli istnieje, co najmniej jedno wystąpienie encji, które nie uczestniczy w związku.

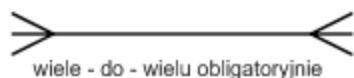
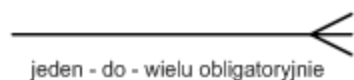
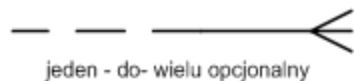


Typy encji	Notacja Martina
------------	-----------------

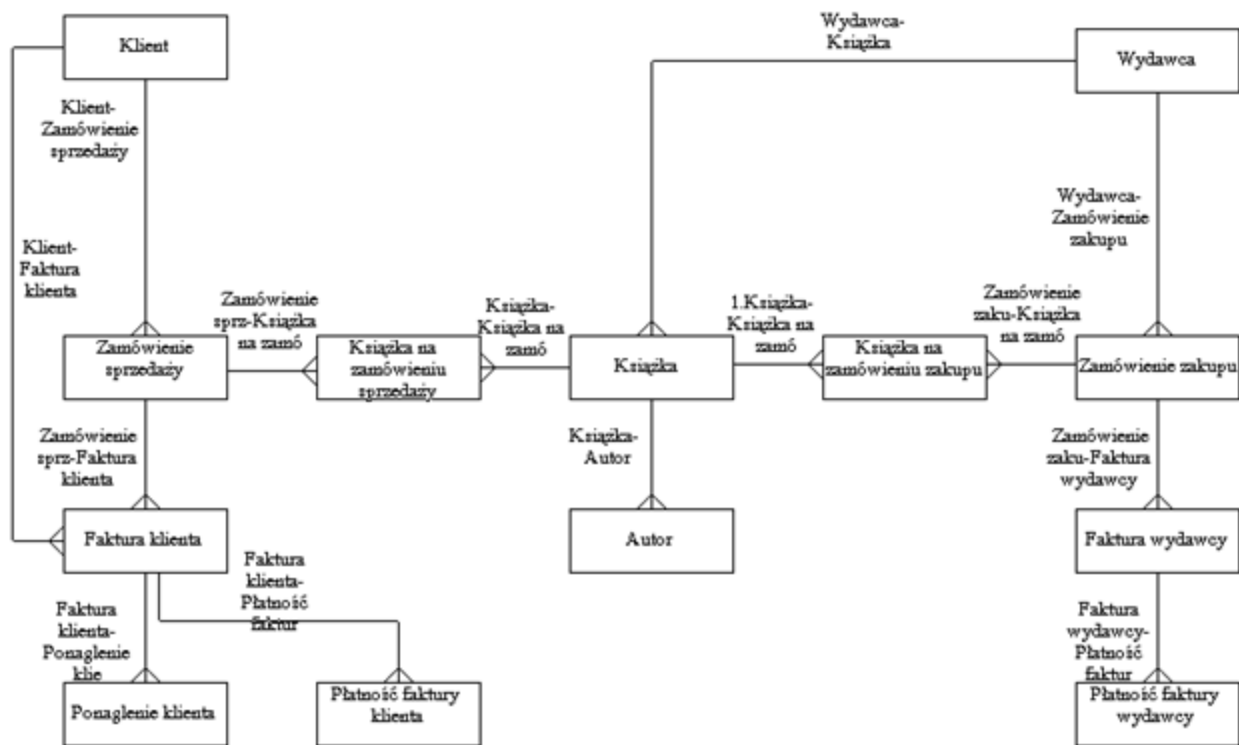
<p>Encja regularna– oznacza dowolny znaczący element, o którym informacja powinna być znana albo utrzymywana (częściowe uczestnictwo w związku).</p>	
<p>Encja słaba – jest to encja, która może istnieć tylko wtedy, gdy jest związana z innymi encjami lub też nie posiada własnych atrybutów kluczowych (całkowite uczestnictwo w związku)</p>	
<p>Encja – obiekt asocjacyjny – przechowuje informacje o związku pomiędzy dwiema encjami.</p>	

Diagramu ERD jest narzędziem do komunikowania za pomocą, którego osiąga się z użytkownikiem końcowym porozumienie, co do wymagań biznesowych. Identyfikuje i opisuje on dane wymagane przez system oraz pokazuje jak w systemie powiązane są ze sobą elementy danych. Dodatkowo należy również sformułować ograniczenia nałożone na związek, które noszą nazwę **reguł integralności**. Stanowi on podstawę do projektowania bazy danych.

Uwaga: z innymi metodami projektowania ERD można się spotkać np. używając narzędzi Oracla. Rysunek zawiera przykładową notację związków:



Przykładowy diagram encji:



XML i JSON

Usługi sieciowe wykorzystują do komunikacji ustandaryzowane dane tak, by możliwa była ich wymiana bez potrzeby implementowania wielu metod kodujących i dekodujących te dane. Najpowszechniejszymi formatami danych używanymi w technologii web service są XML oraz JSON. W poniższej tabeli zaprezentowany jest przykład danych zapisanych w obu formatach:

XML	JSON
<pre> <?xml version="1.0" encoding="UTF-8"?> <newOrderSingle type="8"> <accountNumber>55-55</accountNumber> <expireDate>2011-08-16</expireDate> <orderQty>10.0</orderQty> <price>120.5</price> <currency>PLN</currency> <side>1</side> <symbol> <shortName>COMARCH</shortName> <shortName>CMR</shortName> </symbol> </newOrderSingle> </pre>	<pre> { newordersingle:{ type:8, accountnumber: 55-55', expiredate:'2011-08-16', orderqty:10, price:120.5, currency:'PLN', side:1, symbol:{ shortname:['COMARCH', 'CMR'] } } } </pre>

	} }
--	--------

Format XML używa do opisu znaczników znanych z języka HTML. Początki języka XML sięgają jednak SGMLa, który został znacznie uproszczony tak, by przetwarzanie przez parsery XML odbywało się szybciej. W pierwszej linii XML zawiera deklarację wersji oraz kodowania. Obecnie istnieją dwie wersje XML-a: 1.0 oraz 1.1. Definicja kodowania znaków pozwala poprawnie interpretować zapisane znaki. W kolejnych liniach zapisana jest właściwa treść dokumentu. Każdy element posiada znacznik otwierający jak i zamykający (np. odpowiednio: `<price>` oraz `</price>`). Każdy element może posiadać dowolną liczbę atrybutów, które są oddzielone spacją obok nazwy elementu. W powyższym przykładzie atrybutem jest `type`, natomiast wartość (w tym wypadku - liczbową) wynosi 8. Dany element może zawierać kilka podelementów o identycznych nazwach. Tworzona jest wtedy swego rodzaju tablica. W powyższym przykładzie tablicę tą stanowią elementy o nazwie `<shortName>`. Pliki XML mogą być z powodzeniem konwertowane do innej postaci XML-a bądź dowolnego innego typu plików przy użyciu transformacji XSLT. Najczęściej używane jest to do prezentacji danych na stronie internetowej. Ciekawym narzędziem są również schematy XML Schema, które pozwalają walidować dany dokument XML według podanego w schemacie wzorca. W schemacie można określać nie tylko, że dany element musi przechowywać wartość tekstową czy liczbową, ale także zakres wartości czy maksymalną długość przechowywanego ciągu znaków.

Format JSON zorientowany jest na prostotę. Twórcy tego formatu uznali, że zamykające znaczniki są nadmiarowe, dlatego zrezygnowano z nich. Początek i koniec danego elementu oznaczone są przy użyciu nawiasów klamrowych. Przed nawiasem otwierającym znajduje się dwukropek, natomiast przed dwukropkiem nazwa elementu. Tablice rozpoczyna się nawiasem kwadratowym a poszczególne elementy oddziela przecinkiem. Warto zauważyć, że poszczególne elementy nie posiadają żadnych atrybutów. Konwertując dane z XML-a do JSON-a, wszelkie atrybuty elementu stają się jego podelementami. JSON wspierany jest przez wiele języków programowania. W JavaScript jest on obsługiwany natywnie bez potrzeby korzystania z dodatkowych bibliotek. Bardzo częstym zastosowaniem JSON-a jest serializacja obiektów. Kiedy chcemy zapisać obiekt będący instancją danej klasy programistycznej, kompilator często wręcz bez naszej wiedzy dokonuje serializacji do JSON-a a następnie zapisuje go do pliku bądź bazy danych. Istnieją również odmiany tego formatu, które mają większe możliwości niż prosty JSON. Przykładowo JsonML potrafi rozróżniać elementy od atrybutów, jednak rozmiar pliku wygenerowanego w tym formacie znacząco wzrasta. Ponadto obsługa tego języka nie jest już tak prosta w wielu językach programowania jak obsługa czystego JSON-a.

Analiza porównawcza wad i zalet XML i JSON.

Różnice omówionych standardów zostały przedstawione w poniższej tabeli:

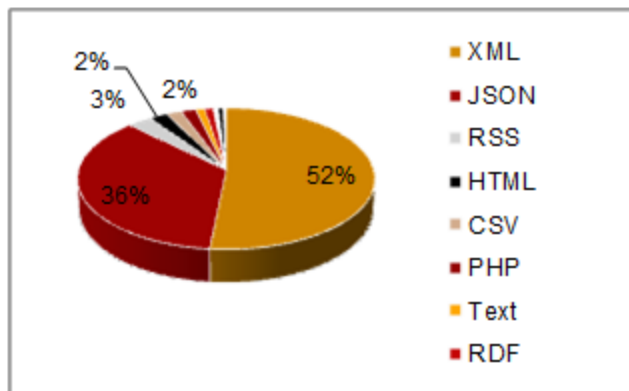
	XML	JSON
Kodowanie znaków	Zapisane w deklaracji (1. linia)	Określane ręcznie lub odczytywane z nagłówka HTTP (<i>Content-Encoding</i>)
Objętość	Nadmiarowość: 312B	Prostota: 159B
Możliwości	Bardziej złożone struktury	Proste struktury zapisu
Walidacja	Poprzez procesor XML Schema	Mało popularny JSON Schema

Korzystając z formatu JSON pewnym problemem może być określanie kodowania znaków użytych w dokumencie. O ile XML jasno i wyraźnie ma zapisane kodowanie w swym nagłówku, tak JSON musi posilkować się innymi, nieco mniej skutecznymi sposobami. Niekiedy kodowanie zapisuje w elemencie stworzonym specjalnie do tego celu. Można również polegać na wartości zapisanej w nagłówku wiadomości HTTP (*Content Encoding*), jednak praktyka pokazuje, że niewiele usług sieciowych ustawia prawidłowe wartości w tym nagłówku.

Zdecydowaną zaletą JSON-a w porównaniu z XML-em jest bez wątpienia prostota, która prowadzi do znacznego zmniejszenia rozmiaru dokumentu zapisanego w formacie JSON. Przykładowo informacje zapisane w poprzednim rozdziale w formacie XML zajmują 312 B, natomiast te zapisane w formacie JSON jedynie 159 B. Dla niektórych prostota może być jednak wadą, ponieważ struktura formatu JSON jest uboższa. Naszym zdaniem w większości przypadków JSON jest zupełnie wystarczający, a korzyści przewyższają pewne ograniczenia tego formatu. Do niedawna największą zaletą XML-a w stosunku do JSON-a było istnienie wielu różnych walidatorów plików XML korzystających ze schematów XML Schema. Ostatnio powstaje jednak coraz więcej narzędzi dokonujących podobną walidację przy użyciu schematów JSON Schema.

W użyciu istnieje również wiele innych formatów innych niż XML, JSON oraz ich pochodne. Poniższy wykres pokazuje udział poszczególnych formatów na rynku usług sieciowych (dane pochodzą z końca 2011 roku). Widać wyraźnie, że znakomita większość usług sieciowych korzysta z XML-a lub JSON-a. Pozostałe formaty danych wykorzystywane są w pewnych specjalistycznych zastosowaniach, które jednak ze względu na małą popularność nie będę tu szerzej omawiane.

Warto również wspomnieć, że format JSON zdobywa coraz większą popularność a jego udział za 1-2 lata powinien być największy w środowisku usług sieciowych.



Złote zasady przy projektowaniu WebService'ów:

- Używaj standardowych kodów odpowiedzi HTTP
- **Używaj przyjaznych kodów, skrótów/Nie używaj nic nieznaczących oznaczeń.**
- **Np.: s (zamiast 1) - sprzedaż, k (zamiast 2) - kupno**
- **Pozwól ograniczyć rozmiar odpowiedzi**
- **Utrzymuj zawsze aktualną dokumentację**
- Udostępniaj przykładowe zapytania do WebService'u
- Informuj o nowościach/zmianach

Metodyka tworzenia schematu XML

Konwersja danych do modelu hierarchicznego rodzi potrzebę zdefiniowania schematu opisu struktury danych oraz typu i zakresu prezentowanych informacji, zgodnych ze schematem modelu relacyjnego. Specyfikacja języka XSD dostarcza zestawu narzędzi dla poprawnego odwzorowania elementów modelu relacyjnego (tabele, kolumny) wraz z zachowaniem występujących zależności oraz integralności istniejących danych. Realizowane jest to z jednej strony poprzez możliwość zastosowania określonego zestawu elementów oraz atrybutów języka XSD (key, unique, field, selector), umożliwiających wymuszenie unikalności wartości elementu (idea klucza własnego z modelu relacyjnego), z drugiej zaś strony poprzez określenie relacji zależności pomiędzy elementami schematu hierarchicznego (keyref, refer, selector, field), odzwierciedlającymi związki występujące w modelu relacyjnym.

Należy zauważyć, iż ze względu na specyfikę i różnorodność systemów DBMS, niemożliwym wydaje się opracowanie jednolitych zasad tworzenia schematów. Istniejące rozwiązania realizują funkcje konwersji pomiędzy schematami danych najczęściej w oparciu o:

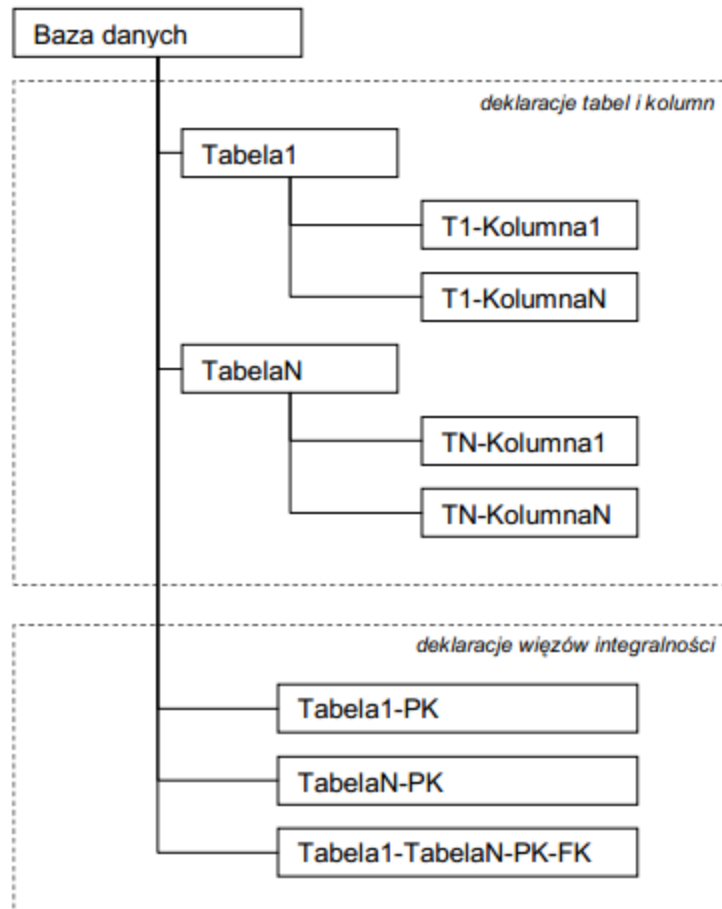
- **model hierarchiczny** (hierarchical schema model), w którym odwzorowanie elementów struktury relacyjnej do struktury hierarchicznej następuje w oparciu o deklarację

elementów i/lub zbioru atrybutów języka XSD, natomiast integralność referencyjna pomiędzy tabelami modelu relacyjnego zachowana jest przy użyciu dostępnych konstrukcji językowych schematu (elementy i atrybuty: key, unique, keyref, selektor, field, refer, xpath),

- **model płaski** (flat schema model), odnoszący się do standardu ISO-ANSI, definiujący zasady mapowania bazy danych do schematu XML, w którym integralność referencyjna danych zachowana jest bez użycia dostępnych konstrukcji językowych XSD, natomiast występujące zależności definiowane są przy użyciu składni annotation/appinfo, gdzie deklaruje się klucze własne oraz klucze obce wiążące poszczególne elementy, odzwierciedlając tym samym zależności występujące w strukturze relacyjnej.

Odwzorowanie elementów modelu relacyjnego realizowane może być zarówno poprzez zastosowanie deklaracji elementów, jak i zbioru atrybutów w schemacie XML. Ze względu jednak na istniejące ograniczenia w użyciu atrybutów, wynikające ze składni języka XSD, a także ze względu poprawę czytelności zapisu, preferowane jest wykorzystanie wyłącznie deklaracji elementów. Dla tak przyjętych założeń, algorytm tworzenia schematu XML na podstawie struktury relacyjnej bazy danych można określić w sposób następujący:

1. W tworzonym dokumencie XSD dodaj element reprezentujący relacyjny model danych (DataSet).
2. Dla każdej tabeli z modelu relacyjnego dodaj element potomny w DataSet (Tabela1..TabelaN).
3. W każdym z elementów Tabela1..TabelaN utwórz elementy potomne zawierające deklaracje kolumn, odnoszących się do tabeli relacyjnej.
4. Dla każdego klucza własnego występującego w tabelach modelu relacyjnego dodaj element potomny (Tabela1-PK..TabelaN-PK) w elemencie DataSet, wymuszający unikalność danych.
5. Dla odwzorowania zależności występujących w schemacie relacyjnym oraz zachowania istniejących więzów integralności, utwórz elementy potomne (Tabela1-TabelaN-PK-FK) w DataSet, określając jednocześnie referencje do kluczy własnych (Tabela1-PK..TabelaN-PK).



Zaproponowany algorytm oparty jest na **modelu hierarchicznym**. Użycie modelu płaskiego uwarunkowane jest implementacją standardu SQL/XML w systemie zarządzania bazą danych.

Praktyczna realizacja tworzenia schematu XML w oparciu o relacyjną strukturę danych wiąże się z koniecznością rozstrzygnięcia szeregu kwestii, których podłoże tkwi przede wszystkim w różnicach występujących w językach modelowania danych relacyjnych (DDL) i hierarchicznych (XSD) oraz użytej metodzie dostępu do danych. Szczegółowa analiza tych zagadnień pozwala określić klasy problemów, jakie należy rozwiązać przed przystąpieniem do konwersji schematów. W szczególności jest to:

- określenie metody dostępu do DBMS (API, ADO, ODBC, JDBC) oraz uwzględnienie ograniczeń z tego wynikających,
- wybór elementów struktury relacyjnej podlegających transformacji (tabele użytkownika, widoki, tabele systemowe) przy jednoczesnym uwzględnieniu praw dostępu (w zależności od stosowanego DBMS),
- odwzorowanie elementów schematu struktury relacyjnej (tabele, kolumny) przy

- pomocy deklaracji elementów, atrybutów lub ich kombinacji w schemacie XML,
- odwzorowanie relacji mnogościowych oraz zachowanie więzów integralności (deklaracja kluczy własnych, kluczy obcych oraz występujących zależności),
 - tworzenie elementów i atrybutów dla wartości pustych oraz wartości null,
 - dobór nazw dla elementów i atrybutów (eliminacja kolizji nazw),
 - konwersja typów danych, formatu oraz zakresu dopuszczalnych wartości, w szczególności wartości binarnych oraz formatu daty i czasu.

W chwili obecnej dostępny jest na rynku szeroki wachlarz narzędzi realizujących tworzenie schematu XML na podstawie struktur relacyjnej bazy danych: Altova XMLSpy (SPY), Microsoft Visual Studio (MVS) oraz Oxygen (OGX).

Dokonując analizy uzyskanych wyników należy stwierdzić, iż żadne z wykorzystanych narzędzi nie pozwoliło w pełni na uzyskanie całkowicie poprawnych rezultatów. Mimo niewielkiego stopnia złożoności struktury relacyjnej zaobserwować można szereg niezgodności w utworzonych schematach, wśród których najważniejsze to:

- błędna interpretacja występującej w bazie danych relacji m-n (SPY),
- brak zachowania więzów integralności w odwzorowywanej strukturze relacyjnej (OGX),
- nieprawidłowe mapowanie typów danych (MVS),
- niepoprawne określenie zakresu dla danych znakowych oraz dopuszczalnych wartości dla danych numerycznych (MVS),
- brak określenia wymaganych wartości w schemacie XML, w szczególności dla mapowanych kluczy własnych (MVS, OGX).

Należy stwierdzić, iż wymienione nieprawidłowości, mimo zastosowania narzędzi mających charakter komercyjny, nie pozwalają bezpośrednio na wykorzystanie utworzonych schematów XML. Dla uzyskania pełnej ich zgodności ze strukturą bazy danych niezbędne są dodatkowe modyfikacje wygenerowanych przez badane aplikacje dokumentów XSD. Uzyskane rezultaty mogą jednak stanowić punkt wyjściowy na drodze do tworzenia schematów zgodnych z relacyjną strukturą danych.

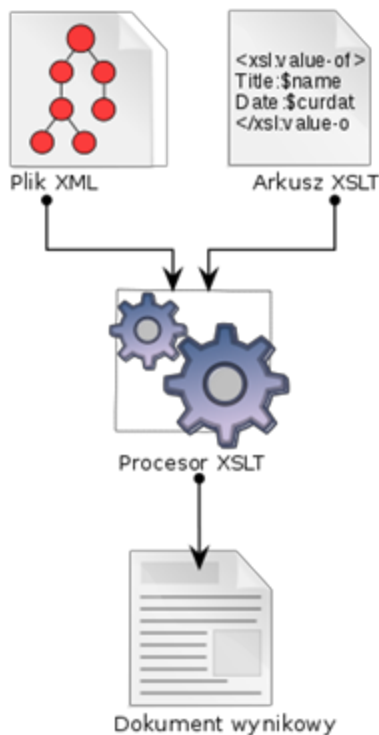
Rozwiązaniem może stać się gromadzenie i przechowywanie informacji w dedykowanych bazach danych XML, zachowujących strukturę dokumentu, co w efekcie eliminuje potrzebę konwersji pomiędzy formatami opisu danych. Na rynku dostępnych jest szereg produktów natywnych baz danych, jednakże na chwilę obecną nie istnieje jednolity standard języka kwerend (XQuery), co może uniemożliwiać szersze wykorzystanie tego rozwiązania.

XSLT

XSLT (ang. XSL Transformations, Extensible Stylesheet Language Transformations, w wolnym tłumaczeniu Przekształcenia Rozszerzalnego Języka Arkuszy Stylów) – oparty na

XML-u język przekształceń dokumentów XML. Pozwala na przetłumaczenie dokumentów z jednego formatu XML na dowolny inny format zgodny ze składnią XML-a (np. na stronę WWW XHTML, wzór matematyczny MathML lub dokument biurowy ODF), jak również na zwykły HTML i czysty tekst.

Dzięki dużej sile wyrazu, łatwości implementacji i powszechnemu stosowaniu XML-a jako standardu dla zapisu informacji, XSLT jest uniwersalnym narzędziem znajdującym zastosowanie w wielu rodzajach oprogramowania. Najbardziej popularne to generowanie stron WWW w serwisach internetowych oraz konwersja pomiędzy alternatywnymi formatami np. w pakietach biurowych.



XSLT jest rozwijany przez W3C jako część rodziny języków XSL (obok XPath i XSL-FO). Powstał pod wpływem języków funkcyjnych oraz języków opartych na dopasowywaniu wzorców (ang. pattern matching) jak awk. Jego bezpośrednim poprzednikiem jest DSSSL, odpowiednik XSLT dla SGML-a.

Najnowszą wersją rekomendacji XSLT jest 2.0, ale ze względu na jej powolne rozpowszechnienie (vide brak obsługi w Xalanie, czy "fabrycznie" na platformie .NET) rekomendacja 1.0 jest nadal powszechnie wykorzystywana.

Omówienie

XSLT przypomina języki funkcyjne - arkusze XSLT zbudowane są z reguł opisujących, w jaki sposób zamienić poszczególne elementy wejściowego XML-a. Warsztat programisty

XSLT obejmuje m.in. instrukcje sterujące, możliwość definicji własnych funkcji (tzw. szablonów nazwanych), funkcje wbudowane realizujące na przykład sortowanie. Do znajdowania i wskazywania elementów źródłowego XML-a używany jest XPath.

Składnia i semantyka

Arkusze XSLT są poprawnymi dokumentami XML. Elementem głównym jest `xsl:stylesheet`. Prefiks `xsl:` w używanych w tym dokumencie nazwach elementów oznacza jedynie, że należą one do przestrzeni nazw XSLT - tak więc `xsl:stylesheet` należy czytać jako element `stylesheet` z przestrzeni nazw XSLT.

Algorytm transformacji

Każdy procesor XSLT posługuje się przedstawionym poniżej w skrócie algorytmem.

1. Przygotowanie do transformacji:

- Parsowany jest arkusz XSLT oraz wejściowy XML oraz budowane są ich drzewa. Uwzględniony jest fakt, że arkusz XSLT może się składać z wielu plików (instrukcje `xsl:include` i `xsl:import`).
- Z dokumentów usuwane są nadmiarowe białe znaki.
- Do drzewa XSLT dołączane są standardowe reguły.

2. Transformacja:

- Tworzony jest główny element drzewa wyjściowego (root node).
- Główna część: przetwarzane są elementy drzewa wejściowego, począwszy od elementu głównego.
- Zwracane jest drzewo wyjściowe, w formacie określonym przez `xsl:output`.

Każdy element drzewa wejściowego przetwarzany jest następująco:

- Znajdowany jest najlepiej pasujący szablon. Ze wszystkich szablonów pasujących do przetwarzanego elementu (każdy szablon nienazwany ma wzorzec - atrybut `match`) wybierany jest ten o najwyższym priorytecie (obliczonym na podstawie atrybutu `priority`, postaci wzorca oraz pozycji w dokumencie - elementy zaimportowane mają zawsze najniższy priorytet).
- Znaleziony szablon jest aplikowany. Elementy szablonu znajdujące się w przestrzeni nazw XSLT (zazwyczaj te z prefiksem `xsl:`) traktowane są jak instrukcje i odpowiednio interpretowane. Reszta jest zwyczajnie kopiowana do drzewa wynikowego.
- Jeśli w szablonie umieszczona jest instrukcja `xsl:apply-templates`, procesor przechodzi w tym miejscu do **rekurencyjnego** przetwarzania listy elementów wskazanych atrybutem `select` lub - jeśli go brak - wszystkich dzieci aktualnego elementu. Jeśli w szablonie brak jest instrukcji `xsl:apply-templates`, żadne z elementów aktualnego poddrzewa (dzieci i ich następniki) nie są w tym miejscu dopasowywane (przetwarzane). Należy jednak pamiętać, że mogą zostać

przeznaczone do dopasowania (za pomocą instrukcji `xsl:apply-templates`) z innego szablonu.

31. Narzędzia integracji systemów informacyjnych.

Webserwisy:

- Webserwis 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
- SOA - Service Oriented Architecture - cele: niezależność od implementacji i lokacji
- UDDI - Universal Description Discovery and Integration - zapewnia mechanizm katalogowania, klasyfikowania i zarządzania webserwisami, dzięki czemu mogą być odkryte i użyte
 - cele:
 - publikowanie informacji o webserwisach
 - kategoryzacja webserwisów
 - odnajdowanie webserwisów
 - określanie protokołów transportu i bezpieczeństwa oraz parametrów danego webserwisu
 - dostarczanie środków do izolacji aplikacji przed błędami i zmianami webserwisów
 - typy:
 - białe strony - lista webserwisów
 - żółte strony - pewne informacje na podstawie ustandaryzowanych lub zdefiniowanych przez użytkownika reguł
 - zielone strony - pełne opisy konkretnych webserwisów
- WSDL - Web Service Description Language - oparty na XML język używany do opisywania funkcjonalności webserwisów. Zapewnia odczytywany maszynowo opis tego, jak serwis ma być wywołany, jakich parametrów oczekuje i jakie dane zwraca.
- SOAP - Simple Object Access Protocol - specyfikacja protokołu wymiany ustrukturalizowanych informacji w implementacji webserwisów w sieciach komputerowych. Opiera się na XML (Extensible Markup Language) jako formacie wiadomości i zazwyczaj na protokołach warstwy aplikacji, najczęściej HTTP (Hypertext Transfer Protocol) i SMTP (Simple Mail Transfer Protocol) dla negocjacji i transmisji wiadomości.
 - Składa się z:

- koperty SOAP - opisującej, co zawiera wiadomość, do kogo dostarczona i czy wymagana/opcjonalna
 - nagłówek wiadomości
 - ciała (body) wiadomości - właściwej treści
- zasad kodowania SOAP - definicji typów danych używanych przez aplikację
- reprezentacji RPC - definiuje zasady opisu zdalnych procedur i ich odpowiedzi

ESB (Enterprise Service Buses) - architektura oprogramowania używana do projektowania i implementowania interakcji i komunikacji między aplikacjami przy użyciu SOA. Promuje asynchroniczny sposób przesyłu wiadomości pomiędzy aplikacjami. Głównym zastosowaniem jest integracja heterogenicznych i złożonych aplikacji.

- Właściwości:
 - inwokacja - zdolność do wysyłania zapytań i odbierania odpowiedzi od serwisów i zintegrowanych zasobów
 - routing
 - oparty na treści - nie trzeba definiować celu wysyłanych wiadomości, wiadomości są filtrowane i wysyłane w określone miejsce, gdy treść spełnia dane kryteria
 - oparty na liście adresatów - wysyłanie do wielu, podział wiadomości i wysyłanie różnych części do różnych osób
 - oparty na agregacji - zbieranie powiązanych wiadomości z wielu miejsc
 - mediacja - ESB może wykonać požądane transformacje i "przetłumaczenia" protokołu, formatu i treści wiadomości
 - adaptory zmniejszające nakład pracy związane z integracją
 - bezpieczeństwo: szyfrowanie, uwierzytelnianie, kontrola dostępu
- Wspierane standardy:
 - SOAP
 - WSDL
 - JMS (Java Message Service)
 - JCA (Java EE Connector Architecture)
 - BPEL (Business Process Execution Language) - język opisujący typy, kolejność akcji, z których składają się całe procesy biznesowe
 - JBI (Java Business Integration) - specyfikacja SOA dla Javy
 - JDBC (Java Database Connectivity)

Pytanie: czy wystarczą webserwisy i ESB, czy EDI i inne też należy opisać? (są w wykładzie Kazienki o webserwisach)

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.

Kiedyś takie systemy wykorzystywały modele strukturalne procesów. Dzięki rozwojowi metod obiektowych, twórcy tych systemów wprowadzali także takie modele, lecz używali różnych metodologii. Dopiero gdy wymyślono UML – stał się on dobrym standardem, a jego obsługa implementowana jest we wszystkich liczących się CASEach.

Kiedyś systemy były wykonywane w technologii z góry wybranej – wybrać trzeba było środowisko systemowe, język programowania czy też bazę danych. Teraz można wybrać kilka technologii (J2EE, .NET) i oprzeć się na kilku silnikach bazodanowych (lub też zuniwersalizować kwestię wyboru bazy danych).

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 wspomaga pierwsze fazy budowy systemu – analizę organizacyjną i funkcjonalną i procesową, modelowanie funkcji, procesów, obiektów, modelowanie struktur i potrafi tworzyć wszelkie diagramy. Te narzędzia zajmują się bardziej opisem i modelowaniem rzeczywistości, modelowaniem struktury systemu, bez wszelkich faz implementacji.

Lower-CASE natomiast wspomaga rzeczywiste budowanie oprogramowania – modelowanie bazy danych, generowanie kodu i testy.

Czasami wyróżnia się także systemy **Middle-CASE**, które pozwalają określić samą strukturę systemu informatycznego, oraz **Integrated-CASE**, czyli systemy łączące Upper- i Lower-CASE.

3. 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.

- **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 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.
- **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 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 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.

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.

4. 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** – otwarta, dostępna na zmodyfikowanej licencji GPL platforma UML/MDA dla systemu Windows, która umożliwia import projektów z takich komercyjnych aplikacji jak Rational Rose czy Borland Together. 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** - profesjonalne narzędzie, działające na platformach Windows i Linux. Obsługuje UML 2.0.
- **IBM Rational Rose** – jedno z najstarszych, profesjonalnych narzędzi. Bardzo rozbudowane, obsługujące UML 2.0.

Wiecej: <http://www.eioba.pl/a/1lc4/narzedzia-case#ixzz1xlornfaR>

33. Podejścia do zarządzania zespołami realizującymi projekty informatyczne.

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

Nowoczesne podejście polega na wsparciu siły roboczej, umożliwianiu wykonania pracy oraz dbaniu o rozwój poprzez szkolenie i zachęcanie do samorozwoju.

Istnieje wiele metodyk, które opisują sposoby postępowania w zarządzaniu zespołem informatycznym (patrz niżej).

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

Główni 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.

*** 14 funkcji lidera ***

1. Wykonawca. Ostateczny decydent, najwyższy koordynator polityki i jej realizacji.
2. Planista. Wyznacza sposób osiągnięcia celów przez zespół lub grupę.
3. Twórca polityki. Wyznacza, wspólnie z innymi, ale jako decydujący, cele i zasady postępowania w grupie.
4. Ekspert. Odgrywa rolę eksperta, chociaż korzysta z rad innych ekspertów.
5. Reprezentant. Mówi w imieniu grupy i reprezentuje ją na zewnątrz. Do niego docierają wszystkie informacje i od niego rozchodzą się dalej.
6. Organizator. Tworzy strukturę organizacyjną.
7. Nagradzający. Kontroluje osoby mu podległe, mając uprawnienia do nagradzania i stosowania kar.
8. Dający przykład. Poprzez własne działania daje przykład oczekiwanych zachowań.

9. Arbiter. Ostatecznie rozstrzyga wszystkie problemy i kontroluje relacje interpersonalne w grupie.
10. Symbol. Stanowi centrum zespołu, daje jej poczucie jedności, pomaga w odróżnieniu od innych zespołów.
11. Podpora. Członkowie zespołu mogą go wykorzystywać do podejmowania za nich trudnych decyzji.
12. Ideolog. Zespoły potrzebują wiary, wartości i standardów zachowań. Lider to tworzy.
13. Postać ojca. Skupia pozytywne emocjonalne odczucia osób podległych.
14. Kozioł ofiarny. Skupia negatywne emocjonalne odczucia osób podległych.

**** Metodyki zarządzania projektem ****

*** 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 wносить 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.

*** PROGRAMOWANIE EKSTREMALNE ***

Programowanie Ekstremalne (XP) to lekka, zwinna

(ang. agile) metodyka tworzenia oprogramowania

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 resztą 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 komunikacją 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, że wymagania klienta, rynku i sytuacja w branży ciągle się zmieniają. Nie ma więc 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 ją 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 między 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 całą grupę. Standard powinien jednoznacznie określać wygląd kodu, ale nie powinien być zbyt długi i szczegółowy. Polecane są opracowania jednostronicowe. Standard dokumentowania zakłada, że 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 coś 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 tylko możliwe. Programista po wykonaniu każdego nowego fragmentu programu łączy go z systemem. Najczęściej stosuje się jedną 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.

34. Podpis elektroniczny

Podpis cyfrowy - matematyczny sposób potwierdzania autentyczności cyfrowego dokumentu. Istnieje wiele

schematów podpisów cyfrowych, obecnie jednak najpopularniejszym jest schemat podpisu dokumentów cyfrowych w systemach kryptograficznych z kluczem publicznym i jednokierunkową funkcją skrótu - w systemie tym do oryginalnej wiadomości dołączany jest skrót dokumentu, zaszyfrowany prywatnym kluczem nadawcy. Potwierdzenie autentyczności wiadomości jest możliwe po odszyfrowaniu skrótu kluczem publicznym nadawcy i porównaniu go z wytworzonym skrótem odebranego dokumentu.

Architektura

Podpis elektroniczny służy zapewnieniu między innymi następujących funkcji:

- autentyczności, czyli pewności co do autorstwa dokumentu,
- niezaprzeczalności nadania informacji, nadawca wiadomości nie może wyprzeć się wysłania wiadomości, gdyż podpis cyfrowy stanowi dowód jej wysłania (istnieją także inne rodzaje niezaprzeczalności),
- integralności, czyli pewności, że wiadomość nie została zmodyfikowana po złożeniu podpisu przez autora.

Do zapewnienia wszystkich wymienionych funkcji potrzebne jest zastosowanie trzech środków:

- instrumentów technicznych - algorytmów, protokołów i formatów, które dzięki zastosowaniu technik kryptograficznych zapewniają integralność oraz wiążą klucz prywatny autora z dokumentem, zapewniając autentyczność i niezaprzeczalność
- instrumentów prawnych, czyli dyrektyw, ustaw i rozporządzeń, które osadzają wymienione instrumenty techniczne w obowiązującym prawie,
- instrumentów organizacyjnych, takich jak centra certyfikacji, które występując jako zaufana trzecia strona poświadczają związek klucza prywatnego z konkretną osobą.

Kryptografia

Podpisy cyfrowe korzystają z kryptografii asymetrycznej – tworzona jest para kluczy, klucz prywatny i klucz publiczny – klucz prywatny służy do podpisywania i deszyfrowania wiadomości, klucz publiczny natomiast do weryfikacji podpisu i szyfrowania.

Najważniejszymi kryptosystemami umożliwiającymi podpisywanie cyfrowe są RSA i ElGamal (rozwinięty w standardy DSA, GOST 31.10 czy KCDSA).

Inne znane systemy proponowane dla podpisu elektronicznego to schematy oparte na krzywych eliptycznych (ECDSA, EC-KCDSA, ECNR) i inne wykorzystujące problem logarytmu dyskretnego (np. XTR), algorytmy wielu zmiennych (jak SFLASH czy Quartz) oraz operujące na kratkach (NTRU-Sign).

Systemy podpisu cyfrowego

Najpopularniejsze standardy pozwalające na złożenie podpisu elektronicznego to X.509 oraz PGP.

PGP jest systemem zdecentralizowanym, w którym poziom autentyczności danego klucza jest determinowany przez sumę podpisów, złożonych przez różne osoby znające posiadacza klucza (model Sieć zaufania). System ten jest powszechnie wykorzystywany w Internecie oraz w środowiskach korporacyjnych (np. niektóre systemy EDI). System X.509 jest systemem scentralizowanym, w którym autentyczność klucza jest gwarantowana przez hierarchię centrów certyfikacji formalnie poświadczających związek klucza z tożsamością jego właściciela. Ze względu na jednoznaczną odpowiedzialność, łatwiejszą do osadzenia w prawie, X.509 jest obecnie dominującym systemem, na którym opiera się aktualnie obowiązujące prawodawstwo o podpisie elektronicznym.

Umocowanie prawne

Pojęcia "podpis cyfrowy" i "podpis elektroniczny" w języku polskim są często mylone. W rzeczywistości ich znaczenie w kontekście prawnym i nazewnictwie unijnym jest odmienne. Pojęcie "podpisu cyfrowego" (digital signature) zostało zdefiniowane przez normę ISO 7498-2:1989 jako "dane dołączone do danych lub ich przekształcenie kryptograficzne, które pozwala odbiorcy danych udowodnić pochodzenie danych i zabezpieczyć je przed fałszerstwem".

Na podstawie tej definicji przyjmuje się, że pojęcie "podpis cyfrowy" jest stosowane wobec szerokiego spektrum mechanizmów matematycznych i technicznych związanych z podpisem elektronicznym. Podpis cyfrowy nie musi

być generowany przez człowieka, do tej kategorii zaliczają się więc liczne zastosowania matematycznej operacji "podpisywania cyfrowego" wykorzystywane np. w protokołach kryptograficznych (np. IPsec), które "podpisują" np. tymczasowe liczby losowe w celu potwierdzenia posiadania klucza prywatnego.

Pojęcie podpis elektroniczny (electronic signature) jest natomiast wprowadzone przez unijną Dyrektywę 1999/93/EC i jednoznacznie określa, że jest to operacja podpisywania konkretnych danych (dokumentu) przez osobę fizyczną.

Osobny artykuł: Podpis elektroniczny (prawo).

W polskiej terminologii prawniczej termin "podpis" jest przywiązany tylko i wyłącznie do osoby fizycznej, co również ogranicza stosowanie pojęcia "podpis elektroniczny". Z tego powodu w ustawie o podpisie elektronicznym operacje, które z technicznego punktu widzenia są "podpisem cyfrowym" ale nie są składane przez osobę fizyczną zostały zdefiniowane pod innymi nazwami. "Podpis" składany przez centrum certyfikacji pod certyfikatem osoby fizycznej nazywa się "poświadczeniem elektronicznym", zaś sam certyfikat centrum - "zaświadczeniem certyfikacyjnym".

35. Specyficzne własności struktur baz danych w systemach informacyjnych.

Krótko o samych bazach każdy powinien umieć powiedzieć.

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)
2. 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.

Z “ZPI Trawinski 09 Zarządzanie jakością”:

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

„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.

ZJO można interpretować jako zmniejszanie w trakcie realizacji projektu różnicy między poziomem jakości wymagany przez użytkownika, a poziomem jakości, jaki reprezentuje system w danym momencie realizacji (wykres s.19). Rzeczywista jakość systemu powinna -

najlepiej przed wypuszczeniem produktu na rynek - przekroczyć (wyprzedzić) poziom wymagań użytkownika.

Istotne kryterium: ryzyko utraty jakości, najczęstsze jego czynniki to:

- nowość projektu,
- złożoność projektu,
- niedostateczne wykształcenie 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. 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

Normy ISO 9000:

- ISO - International Organization for Standardization.
- Normy z serii ISO 9000 dotyczą zapewnienia jakości we wszelakich obszarach działalności.
- Zapewnienie jakości oznacza działanie według odpowiednich procedur
- ISO 9000 w IT – ISO 9001-3 - jest modelem dla zapewnienia jakości w projektowaniu, rozwoju, produkcji, instalowaniu i serwisie oprogramowania.
- 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 wymagania klienta w celu pozyskania jego satysfakcji z dostarczonego produktu.
- 8 zasad normy ISO 9001:2000
 - Koncentracja na kliencie
 - Przywództwo
 - Zaangażowanie ludzi

- Podejście procesowe
- Podejście systemowe do zarządzania
- Ciągłe doskonalenie
- Podejmowanie decyzji na podstawie faktów
- Wzajemnie korzystne powiązania z dostawcami
- Podstawowe elementy ISO 9001:2000
 - Doskonalenie systemu i procesów jest w normie ISO 9001:2000 działaniem ciągłym i podstawową miarą wartości systemu zarządzania w organizacji.
 - Proces doskonalenia polega na ciągłej analizie i ocenie przebiegu procesów oraz realizacji ustalonych celów, z wykorzystaniem wszelkich źródeł jak: pomiary satysfakcji klientów, audyty wewnętrzne, działania korygujące i zapobiegawcze, przeglądy kierownictwa, reklamacje klientów itd.
 - Umożliwia to podejmowanie decyzji usprawniających przebieg procesów takich jak: redukcja kosztów, poprawa jakości pracy, wzrost jakości wyrobów i usług, umocnienie pozycji na rynku.
 - Podstawowym wyznacznikiem skuteczności funkcjonowania systemu jest spełnienie oczekiwań klienta.
 - Poprzez badanie satysfakcji klienta ocenia się na ile jego wymagania zostały spełnione, co umożliwia tym samym stałe doskonalenie i zapobieganie powstawaniu niezgodności.
 - Podstawą systemu zarządzania jakością jest orientacja na procesy.
 - Aby wdrożyć efektywnie system zarządzania, należy posiadać wiedzę dotyczącą podstawowych procesów biznesowych w organizacji.
 - Przy podejściu procesowym obowiązują następujące zasady:
 - podstawowe procesy organizacji są udokumentowane i poddane analizie
 - powiązania wewnątrz procesów analizowane są przez pryzmat potrzeb klientów
 - powtarzalność i jakość rezultatów procesów zapewniają udokumentowane procedury
 - podstawą określania celów i oceny rezultatów procesów jest pomiar działań
 - zarządzanie procesami opiera się na ich ciągłym doskonaleniu
 - zarządzanie procesami związane jest ze zmianą kultury organizacji

Certyfikacja systemu zarządzania jakością:

- Certyfikacja systemów jakości ma dla wszystkich organizacji szczególne znaczenie.
- Duża konkurencja oraz stawianie wymogu posiadania wdrożonego certyfikowanego systemu zarządzania jakością stwarza nowe wyzwanie dla różnego rodzaju firm.
- Jednostki certyfikujące: polska jednostka: Polskie Centrum Badań i Certyfikacji (PCBC), zagraniczne: BSI, PRS, RW TÜV, KEMA

Capability Maturity Model (CMM) - model dojrzałości procesu wytwórczego

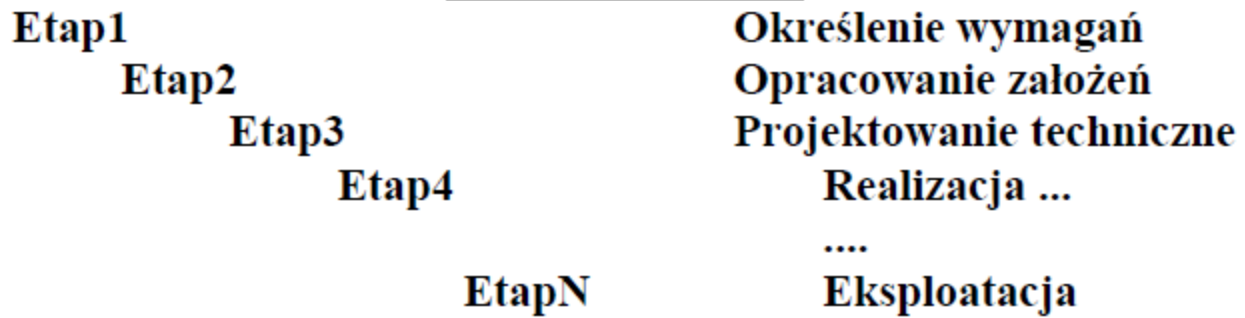
- umożliwia ulokowanie procesów firmy na określonym poziomie dojrzałości

- Wskazuje kierunek zmian w organizacji w celu osiągnięcia większej skuteczności
- Zawiera zasady i ustalone praktyki postępowania określające stopień dojrzałości procesów software'owych
- Może pomóc organizacjom wytwarzającym oprogramowanie w zaplanowaniu ewolucyjnej ścieżki rozwoju
- Założenia:
 - Ulepszanie procesu planowania, produkcji i pielęgnacji oprogramowania,
 - Ocena procesu produkcji oprogramowania we własnej firmie,
 - Ocena zdolności ewentualnych kontrahentów do wywiązywania się ze zobowiązań kontraktowych w zakresie zleceń software'owych.
- Model dojrzałości oprogramowania - etapy dojrzałości:
 - początkowy - słaba definicja procesów, chaos
 - powtarzalny:
 - podstawowe procesy planowane, dokumentowane, monitorowane, kontrolowane
 - śledzenie wymagań, kosztów, planu wykonania
 - zapewnienie jakości i kontrola zmian
 - zdefiniowany:
 - procesy zdefiniowane, ustandaryzowane, zintegrowane, pielęgnowane
 - utworzony w firmie dział odpowiedzialny za wdrażanie polityki jakości i informowanie kierownictwa.
 - zarządzany:
 - przeprowadzane pomiary, kontrole, analizy
 - tam, gdzie to możliwe, następuje poprawa procesów na podstawie zebranych danych pomiarowych
 - optymalizowany
 - Na podstawie wyników pomiarów przeprowadzane są kontrolowane zmiany w procesach celem poprawienia wskaźników.

37. Strategie realizacji systemu informacyjnego.

- Strategia oparta na dokumentach (model kaskadowy)
 - Bardzo dokładna realizacja w oparciu o model kaskadowy.
 - Proces podzielony jest na etapy.
 - Każdy etap kończy się opracowaniem dokumentu.

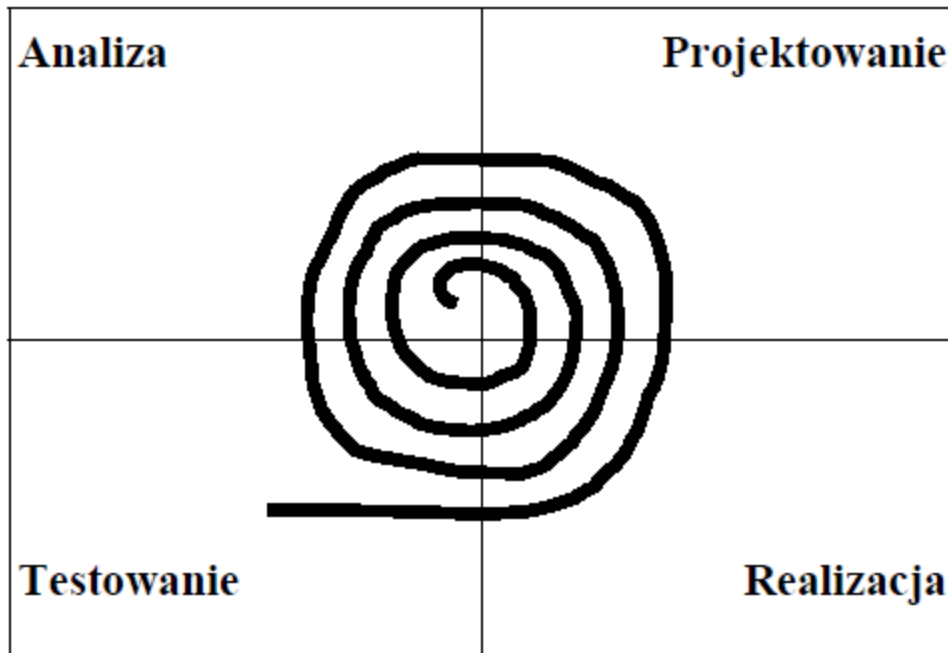
Model kaskadowy



- Prototypowanie
 - Prototyp – ogólny model przyszłego systemu informatycznego, który w kolejnych iteracjach jest udoskonalany aż do osiągnięcia akceptowalnego stopnia szczegółowości.
 - 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
 - Ponadto:
 - możliwość szybkiej demonstracji pracującej wersji systemu
 - możliwość szkoleń zanim zbudowany zostanie cały system.
- Montaż z gotowych komponentów
 - Przyspieszona realizacji systemu na bazie własnych lub kupionych komponentów, modułów, bibliotek programistycznych..
 - 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.
- Programowanie wynalazcze
 - Wymagania do systemu rozpoznawane są w trakcie tworzenia
 - systemu.
- Programowanie przyrostowe
 - Kolejne wymagania rozpoznawane są w trakcie tworzenia systemu i dodawane

do systemu.

- 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.
- Wada:
 - większe koszty realizacji.
- Realizacja w oparciu o model spiralny



- Transformacje formalne

Rozpoznanie i formalna specyfikacja wymagań.

Postać pośrednia.

Postać pośrednia

.....

Kod informatyczny

- Wady:
 - raczej metoda teoretyczna,
 - wąskie zastosowania,
 - język formalnej specyfikacji jest praktycznie językiem programowania,
 - mała efektywność.

38. Systemy biometryczne.

System biometryczny złożony jest najczęściej z czytnika biometrycznego, oprogramowania analizującego oraz bazy danych z pobranymi wzorcami biometrycznymi. Zanim użytkownicy mogą skorzystać z systemu, muszą zostać poddani rejestracji w systemie. Rejestracja polega na pobraniu próbnych odczytów, z wykorzystaniem odpowiedniego algorytmu obliczeniu wzorca biometrycznego dla danego użytkownika. Stosowane algorytmy są różne dla różnych metod biometrycznych oraz dla różnych producentów urządzeń. Jedynie w przypadku analizy tęczówki oka dla wszystkich rozwiązań wykorzystuje się algorytm Johna Daugmana z Cambridge. Algorytmy te cechują się opisaniem relacji pomiędzy wybranymi elementami, charakterystycznymi w przetwarzanym wzorcu. Obliczony wzorzec jest następnie zapisywany w bazie danych wzorców. Istnieją również rozwiązania, w których wzorzec zapisywany jest na karcie elektronicznej, a późniejszy proces uwierzytelniania polega na porównaniu pobranych cech z cechami zapisanymi na tej karcie. Zapewnia to wykorzystanie karty tylko przez właściwą osobę.

Analizując cechy, które mogą być podstawą pomiaru biometrycznego, należy wziąć pod uwagę następujące czynniki:

- powszechność - analizowana cecha powinna występować u większości osób;
- mierzalność - możliwość dokonania wielokrotnego nieinwazyjnego pomiaru danej cechy;
- indywidualność - cecha powinna być charakterystyczna dla danej osoby;
- niezmienność - cecha powinna być stała i nie zmieniać się w związku ze starzeniem się lub przebytymi chorobami;
- niepodrabialność - trudność podrobienia danej cechy.

Pierwszym krokiem jest rejestracja poszczególnych użytkowników w systemie, poprzez pobranie od nich odpowiednich dla systemu cech biometrycznych. Mogą nimi być: odcisk palca, obraz tęczówki, kształt dłoni. Następnie za pomocą odpowiednich algorytmów pobrane dane, zamieniane są w cyfrowe wzorce biometryczne. Wykorzystywane algorytmy są indywidualnie opracowywane przez poszczególnych producentów urządzeń i są one różne nawet w obrębach tych samych technologii biometrycznych. Wyjątek stanowi technologia identyfikacji na podstawie tęczówki oka, która aktualnie w celach komercyjnych wykorzystuje tylko jeden algorytm opracowany na uniwersytecie Cambridge. Wykorzystanie różnych algorytmów w obrębie tej samej technologii powoduje brak kompatybilności pomiędzy systemami konkurencyjnymi.

Wszystkie algorytmy używane do przetwarzania danych biometrycznych (niezależnie od stosowanej metody) nie budują dokładnego modelu cyfrowego, ale analizują i odpowiednio przetwarzają relacje pomiędzy wydzielonymi punktami charakterystycznymi w przetwarzanym wzorcu. W zależności od metody system bierze pod uwagę 30 (odcisk palca), 256 (tęczówka oka) a w geometrii dłoni nawet 30 000 takich punktów. Relacyjna konstrukcja algorytmu biometrycznego jest bardzo ważnym i jednym z najsilniejszych zabezpieczeń danych biometrycznych przed fałszerstwem, czy też wykorzystaniem ich przez osoby niepowołane. Określenie poziomu procentowej zgodności (jest to tzw. próg zgodności) relacji pomiędzy weryfikowanym, a zapisanym wzorcem jest niezbędne do podjęcia decyzji o odrzuceniu lub akceptacji danego użytkownika. Próg zgodności może być różny w zależności od systemu. W przypadku urządzeń rozpoznających tęczówkę oka wynosi on 68%. Dzięki relacyjnej budowie cyfrowych wzorców biometrycznych niemożliwe jest odtworzenie na ich podstawie fizycznego oryginału.

Jedną z ciekawszych cech elektronicznych wzorców biometrycznych jest to, że w efekcie każdego procesu weryfikacji lub identyfikacji, uzyskujemy inny model cyfrowy. Dzięki wykorzystaniu informacji o relacjach pomiędzy poszczególnymi punktami charakterystycznymi system jest w stanie rozpoznać, że ma do czynienia z tą samą osobą pomimo przedstawienia teoretycznie różnych wzorców. W teorii prezentując wielokrotnie te same dane biometryczne nigdy nie uzyskamy takiego samego modelu cyfrowego. Wynika to ze sposobu pobierania próbek biometrycznych. Za każdym razem użytkownik nieco inaczej przykładając palec do czytnika, robi to z różną siłą czy w przypadku systemów tęczówki oka patrzy pod innym kątem lub przesłania powieki. Uzyskane i przetworzone cyfrowo wzorce biometryczne trafiają do bazy danych, która w zależności od systemu może być umieszczona bezpośrednio w urządzeniu, na zewnętrznym serwerze (w przypadku systemów rozbudowanych do dwóch i więcej urządzeń działających w sieci) lub samodzielnym nośniku informacji takim jak np. karta magnetyczna. W tym ostatnim przypadku użytkownik będzie potwierdzał swoją tożsamość poprzez bezpośrednie porównanie swoich cech biometrycznych z

danymi zawartymi na karcie. Wprowadzone do bazy dane mogą podlegać dalszej obróbce informatycznej poprzez dodanie innych informacji, nadaniu odpowiednich uprawnień itp. Tak przygotowane wzorce są wykorzystywane przez system do celów autoryzacyjnych. W wyniku procesu autoryzacji system podejmuje decyzję, czy badany wzorec biometryczny przyjąć czy odrzucić.

Jak wspomniano wcześniej niemożliwe jest dwukrotne uzyskanie identycznego cyfrowego wzorca biometrycznego. Zatem konieczne jest zaprogramowanie dla danego systemu poziomu progu zgodności wzorca. Progi te mogą być różne zarówno w obrębie różnych technologii jak i poszczególnych urządzeń. Zmieniając wartość poziomu zgodności użytkownik może wpływać na bezpieczeństwo i szybkość działania systemu. Zbyt wysoki poziom zgodności może spowodować wzrost liczby odrzuceń użytkowników uprawnionych do korzystania z danego systemu, co znacząco wpłynie na szybkość jego działania. Zbyt niski próg zgodności umożliwi autoryzację osobą nieupoważnioną, co przyczyni się do obniżenia bezpieczeństwa.

<http://www.zbp.pl/photo/ftb/konferencje/biometria25-26.04.08/AutoID.pdf>

39. Systemy pośredniczące w wymianie danych.

Info: wykłady Kazienki + wiki weryfikowane wykładami (wykłady Kazienki były po ang, więc wolałem kopiować z wiki, ale po sprawdzeniu)

System EDI

Elektroniczna wymiana danych (ang. Electronic Data Interchange, EDI) – transfer biznesowych informacji transakcyjnych od komputera do komputera z wykorzystaniem standardowych, zaakceptowanych formatów komunikatów.

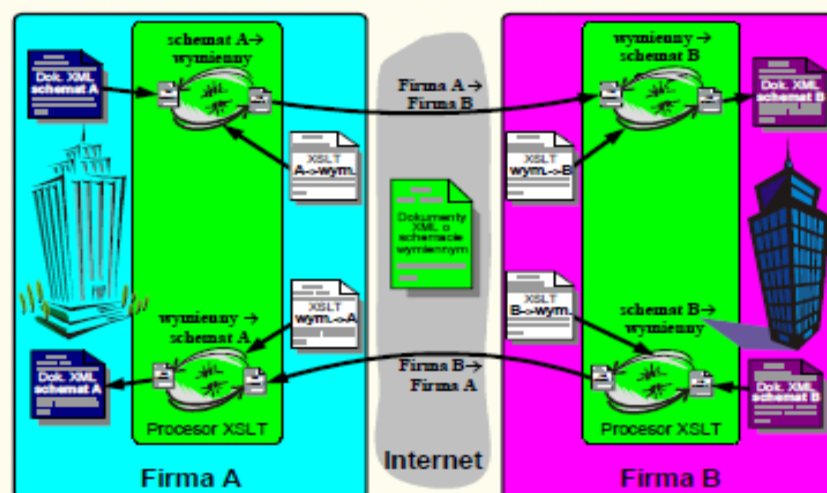
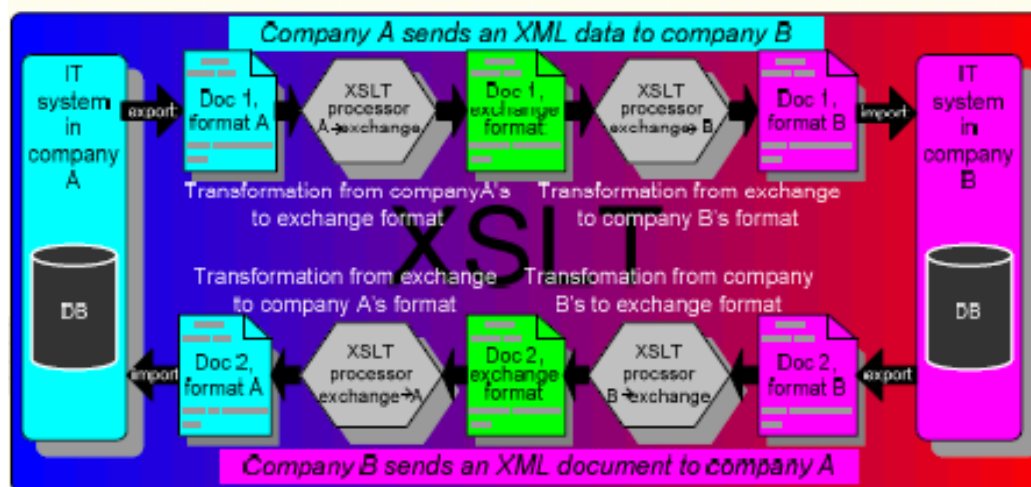
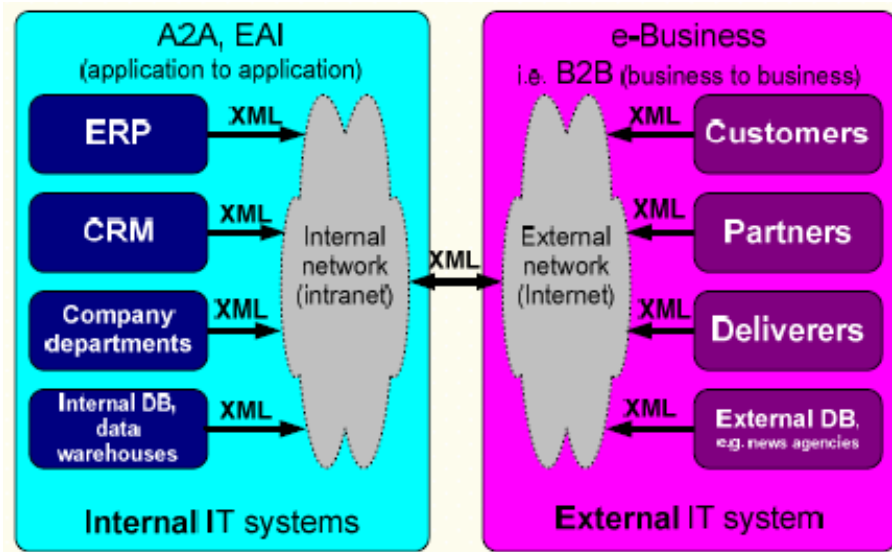
Celem EDI jest wyeliminowanie wielokrotnego wprowadzania danych oraz przyspieszenie i zwiększenie dokładności przepływu informacji dzięki połączeniu odpowiednich aplikacji komputerowych między firmami uczestniczącymi w wymianie. Użycie EDI pozwala poprawić czasową dostępność informacji logistycznej, 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.

Komunikaty EDI są technicznym sposobem zapisu komunikacji biznesowej pomiędzy dwiema stronami (wewnątrz lub na zewnątrz przedsiębiorstwa).

Standardy EDI określają ściśle format przesyłanych dokumentów.

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.

Schematy działania dla EDI

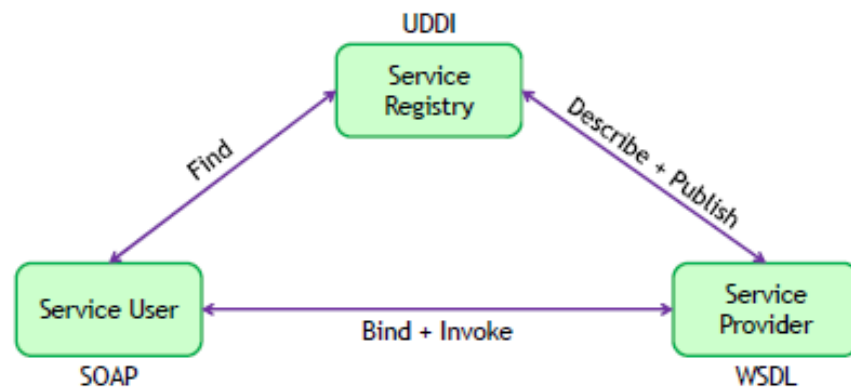


Web service

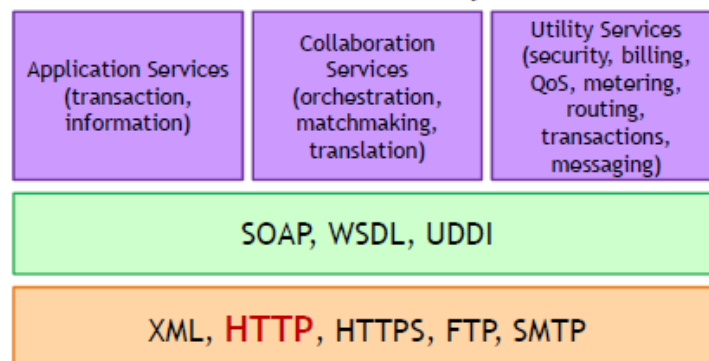
Usługa internetowa (ang. web service) – usługa świadczona poprzez sieć telekomunikacyjną, a w tym sieć komputerową, w szczególności przez Internet.

Usługa internetowa jest w istocie składnikiem oprogramowania, niezależnym od platformy sprzętowej oraz implementacji, dostarczającym określonej funkcjonalności. Zgodnie z zaleceniami W3C, dane przekazywane są zazwyczaj za pomocą protokołu HTTP i z wykorzystaniem XML.

Ten rodzaj usług okazał się skuteczny w sieciach korporacyjnych, za pomocą których przedsiębiorstwa lub instytucje, budowały systemy wymiany danych między swoimi oddziałami, jak również do celów łączności z partnerami i klientami. W takich mniejszych, dobrze kontrolowanych środowiskach, łatwiej jest uzyskać zgodność danych przesyłanych między poszczególnymi komponentami usług a otwartość standardów ułatwia tworzenie rozwiązań klienckich, niezależnie od platformy. Wykorzystanie usług internetowych pozwala komponentom programowym współdziałać ze sobą przez Internet, niezależnie od swojej lokalizacji i szczegółów implementacji. Dzięki temu będą w stanie zastąpić starsze rozwiązania, opracowane dla sieci prywatnych, jak CORBA czy DCOM, zaś dzięki stosunkowo prostej konstrukcji, mogą uzyskać znacznie większą popularność.



Web Services Layers



40. Techniki modelowania danych i przepływu danych.

Model związków-encji (ang. ERD entity-relationships diagram)

Model konceptualny (w uniwersalnym modelu niezależnym od modelu implementacyjnego) wykorzystywany przy projektowaniu relacyjnych baz danych, opisuje właściwości statyczne systemów informatycznych.

Model związków-encji składa się z:

- obiektów świata rzeczywistego reprezentowanych za pomocą encji (entities)
- powiązań między obiektami świata rzeczywistego reprezentowanych za pomocą związków (relationships)

Encja:

- reprezentuje zbiór obiektów opisany tymi samymi cechami (atributami, własnościami)
- informacje o tych obiektach przechowywane w BD
- konkretny obiekt świata rzeczywistego jest instancją encji
- każda encja posiada unikalną nazwę i zbiór cech (atributów)
- encje wchodzą w związki z innymi encjami
- dowolny obiekt może być reprezentowany przez tylko jedną encję
- nazwa encji - rzeczownik w liczbie pojedynczej
- encja posiada unikatowy identyfikator (naturalny - NIP, PESEL; sztuczny - numer pozycji katalogowej, id pracownika) i deskryptory (wszystkie inne atrybuty poza identyfikatorami, opcjonalne lub obowiązkowe)

Związek:

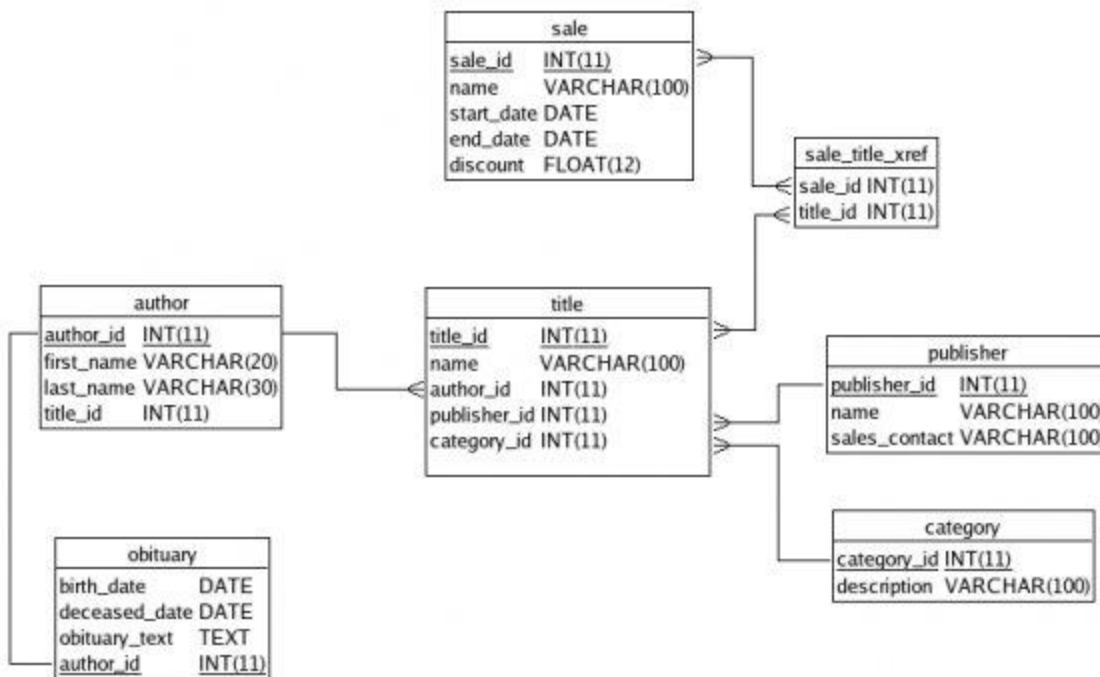
- reprezentuje powiązania pomiędzy obiektami świata rzeczywistego
- łączy encje
- zawiera krótki opis ułatwiający interpretację związku
- posiada 3 cechy:
 - stopień związku
 - unarny
 - binarny (2 encje)
 - ternarny (3 encje)
 - n-arny
 - typ asocjacji
 - jeden-do-jeden (1:1), np. *Każdy dział musi mieć kierownika, natomiast pracownik może być kierownikiem co najwyżej jednego działu.*
 - jeden-do-wiele (1:N), np. *Każdy pracownik pracuje dokładnie w jednym dziale. Dział może zatrudniać (ale nie koniecznie) wielu pracowników*
 - wiele-do-wiele (M:N), np. *Pracownik może brać udział w jednym lub wielu*

projektach; może też nie brać udziału w żadnym projekcie. Każdy projekt realizuje przynajmniej jeden pracownik.

- klasa przynależności
 - opcjonalny
 - obowiązkowy
- posiada atrybuty, np. dla przykładowego związku dla wiele-do-wiele (M:N) dla pracowników, którzy biorą udział w projektach należy zapamiętać ich funkcję, wynagrodzenie oraz daty początku i końca ich udziału w projekcie.

Encja silna / słaba - tylko encja silna posiada własny identyfikator i występuje niezależnie od innych encji, słaba może wystąpić tylko w kontekście innych wystąpień encji (np. dla przykładu asocjacji wiele-do-wiele może to być 'realizacja', która jest wykonywana przez 'pracownika' w ramach 'projektu' - bez 'pracownika' i 'projektu' nie ma realizacji).

Generalizacja - określa, że pewne encje o wspólnym zbiorze atrybutów można uogólnić i stworzyć encję wyższego poziomu - encję generalizacji (dla pracownika i klienta encją generalizacji będzie osoba)



Diagramy przepływu danych (ang. DFD dataflow diagram)

Opisuje właściwości dynamiczne systemów informatycznych.

Diagram przepływu danych składa się z:

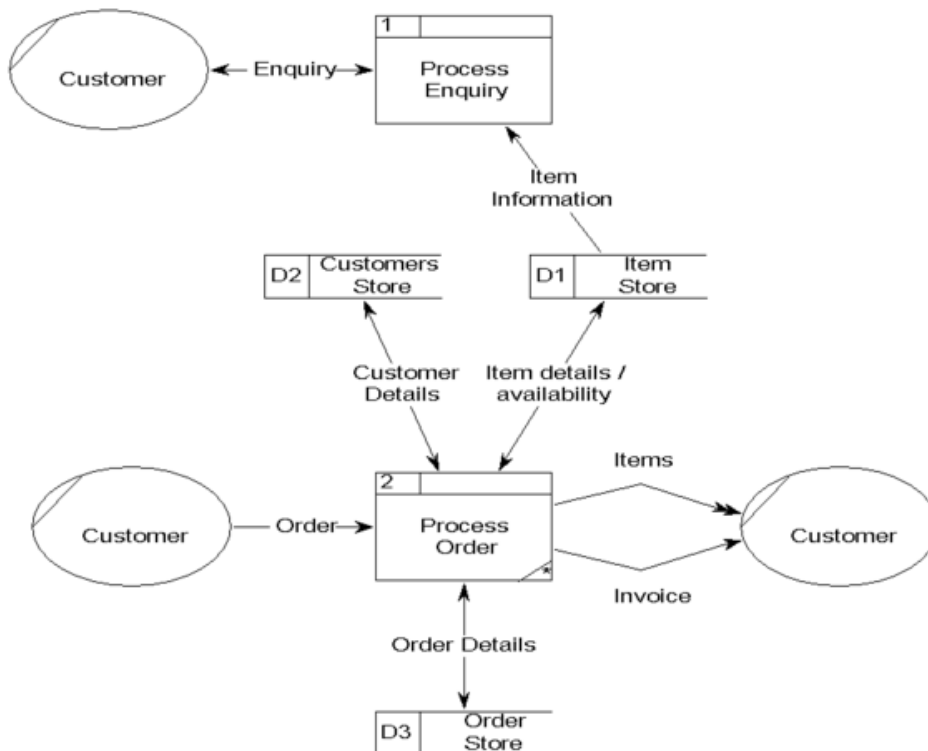
- procesów - czyli pojedynczych funkcji realizowanych przez system (na diagramie procesy reprezentowane są przez kółka)
- przepływów – czyli związków między procesami, funkcjami systemu (na diagramie

przepływy reprezentowane są przez strzałki)

- zbiórów danych – czyli magazynów, służących do przechowywania danych – pliki, bazy danych (na diagramie magazyny reprezentowane są przez elipsy lub nazwa magazynu umieszczona jest między dwiema poziomymi kreskami)
- terminatorów – czyli zewnętrznych obiektów, z którymi komunikuje się system, np. inny dział, inna firma, inny system, inna osoba itp. (na diagramie terminatory reprezentowane są przez prostokąty).

Schematy te wzbogacane są o pomocnicze, tekstowe narzędzia modelowania:

- specyfikację danych – słowniki zawierające definicje struktur danych
- specyfikację procesu – idee i specyfikacje algorytmów.



2

41. Tłumaczenie komputerowe tekstów w językach naturalnych.

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.

- **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.
- **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 zebranych 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.

Podstawowe jednostki języka. Podlegają one abstrakcji.

- **Głoski**, czyli konkretne dźwięki. Abstrakcją głosek są **fonemy**. Fonem to klasa dźwięków, które użytkownicy języka poznają jako posiadające pewne odrębne cechy, wyróżniające je spośród innych dźwięków.
- **Morfy** – są najmniejszymi składnikami języka posiadającymi znaczenie. Ich abstrakcję zwiemy **morfemami**.
- **Wyrazy** – konkretną formę wyrazu, którą używamy, nazywamy wyrazem tekstowym. Abstrakcją dla niego jest leksem. Teraz mały przykład: domem, domy to dwa wyrazy tekstowe jednego leksemu, z kolei wyraz domy składa się z dwóch morfemów: dom – budynku oraz końcówki y wskazującej, że jest ich więcej niż jeden.
- **Frazy** (związki frazeologiczne) – ich abstrakcje to schematy frazy.
- **Zdania** – przyjmuje się, iż są podstawową jednostką tekstu. Ich abstrakcjami są schematy zdań.
- Wypowiedź – jest po prostu ciągiem zdań.

Niekiedy frazy i zdania traktuje się jako jedno i określa mianem **sememów**.

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 wyjaśnienie).

- **Sensowność logiczna** – nie ujawnia się empirycznie. Sensowność ta jest związana z językiem logiki. Jednakże język logiki komunikowalny jest tylko poprzez język naturalny. Powstaje problem przekładu. Wypowiedz jest sensowna logicznie gdy jest skorelowana ze zdaniem logicznym. 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. Ta sensowność jest adaptacyjnością. Pojawia się rozróżnienie na kod kulturowy sytuacji i sens materialny. Sens materialny może być taki sam, a kody kulturowe różne. Sensowność ta jest stopniowalna.
- **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 wyrażań 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 następuje ona trudności przez to, iż wyrazy mają różny zakres pojęciowy. W tłumaczeniu problem ten ulega tylko powiększeniu. Częstość 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, biorąc pod uwagę dotychczasowe rozważania.

42. Własności sieci neuronowych.

Była o tym seminarka u Siemińskiego, też przydałoby się z niej to opisać.

Krótki wstęp teoretyczny:

Sieci Neuronowe (SN) nawiązują do połączeń synaptycznych między neuronami w mózgu.

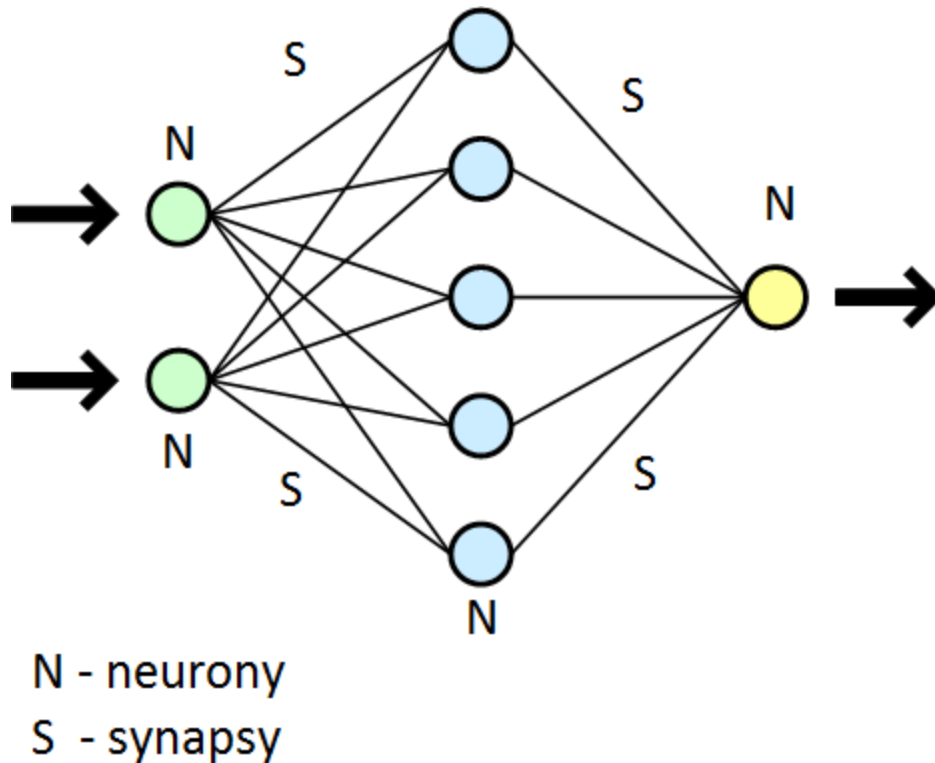
Sztuczne neurony są połączone ze sobą synapsami. Każdy neuron to niezależna jednostka obliczeniowa połączona z pozostałymi równolegle. Im trudniejsze zadanie tym więcej potrzeba neuronów, definicja wymaga więcej niż 1. Neurony mają określoną ilość wejść i wyjść. Każda synapsa ma przypisaną wartość liczbowa zwaną wagą. Pobudzenie (excitation) neuronu to suma ważona wejść pomniejszona o ustalony próg (Threshold).

Funkcja przejścia/przenosząca/aktywacji f (transfer function): liniowa lub (z reguły) nieliniowa (element progowy, funkcja sigmoidalna, tangens hiperboliczny); wylicza wyjście z neuronu na podstawie wejść; ma kluczowe znaczenie dla funkcjonowania neuronu.

Wagi na synapsach są dostosowywane w wyniku uczenia sieci. Uczenie w zależności od modelu może przebiegać w różny sposób.

Własności SN:

- + rozproszony charakter przetwarzania informacji i wynikająca z niego
 - odporność na uszkodzenie/eliminację znacznej nawet części neuronów
 - odporność na uszkodzone i zaszumione wzorce
- + równoległe przetwarzanie informacji (w przypadku implementacji sprzętowych)
 - szybkość działania (w realizacji sprzętowej, wciąż raczej rzadkiej i kosztownej)
- + SN można zastosować w sytuacji gdy programista nie zna algorytmu który rozwiąże dany problem - stosując samouczące się SN.
- powolność większości algorytmów uczących: często setki tysięcy iteracji
- trudności z interpretacją wiedzy nabytej przez sieć (brak lub słabe własności eksplikatywne) w związku z jej (tj. wiedzy) rozproszeniem w sieci (tzw. distributed knowledge representation); czarna skrzynka, blackbox
- zwłaszcza w uczeniu maszynowym: trudności z reprezentacją niektórych typów danych, np. cech/atributów nominalnych o wartościach nie podlegających uporządkowaniu; konieczność stosowania kodowania "1 of n"
- duża ilość parametrów (sieci i algorytmu uczącego), przy jednoczesnym braku ścisłych reguł do estymacji ich wartości



Uproszczony model SN.

43. Zarządzanie ryzykiem w projekcie informatycznym.

Z “ZPI Trawinski 04 Zarządzanie ryzykiem”:

Ryzyko to prawdopodobieństwo wystąpienia niepożądanego zdarzenia oraz niepożądanego wpływu tego zdarzenia na cele projektu.

Ryzyko poprzedza problem.

Ryzyko to prawdopodobieństwo, że w danym punkcie cyklu życia oprogramowania, jego zaplanowane cele nie zostaną osiągnięte w ramach dostępnych zasobów.

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.

Trzy wymiary istotnie wpływające na ryzyko:

- **wielkość projektu**, obejmująca m.in. przewidywany czas realizacji, koszt, liczbę działów organizacji objętych projektem (im większy projekt, tym większe ryzyko)

- **doświadczenie technologiczne**, odnoszące się zarówno do organizacji, jak i firmy tworzącej i wdrażającej system (im mniejsze doświadczenie, tym większe ryzyko)
- **uniwersalność systemu** (mała uniwersalność odpowiada systemom dedykowanym, zaś duża uniwersalność systemom standardowym, wykorzystywanym w wielu organizacjach; im większa uniwersalność, tym większe ryzyko)

Realizacja projektu zakończy się niepowodzeniem, gdy nie uda się w nim osiągnąć spodziewanych wartości w kluczowych czynnikach krytycznych, czyli:

- nie zostanie dotrzymany termin realizacji
- projekt przekroczy zaplanowany budżet
- jakość produktu będzie niska

Funkcje zarządzania ryzykiem:

- **Identyfikacja** – poszukiwanie i określanie ryzyk (zagrożeń) zanim staną się problemami.
- **Analiza** – przetwarzanie danych o ryzyku w informacje niezbędne do podejmowania decyzji.
- **Planowanie** – przetworzenie informacji o ryzyku w decyzje i działania (zarówno bieżące, jak i przyszłe) oraz wdrożenie tych działań.
- **Śledzenie** – monitorowanie wskaźników ryzyka oraz działań przekształcających.
- **Kontrola** – korygowanie odchyleń od zaplanowanych działań postępowania z ryzykami
- **Komunikacja** – dostarczanie informacji oraz zbieranie informacji zwrotnej do wewnątrz, jak i na zewnątrz projektu, na temat działań postępowania z ryzykami, bieżących ryzyk i powstających ryzyk.

Analiza SWOT:

- **SWOT** – jedna z najpopularniejszych heurystycznych technik analitycznych, służąca do porządkowania informacji.
- Bywa stosowana we wszystkich obszarach planowania strategicznego jako uniwersalne narzędzie pierwszego etapu analizy strategicznej.
- Np. w naukach ekonomicznych jest stosowana do analizy wewnętrznego i zewnętrznego środowiska danej organizacji, (np. przedsiębiorstwa), analizy danego projektu, rozwiązania biznesowego itp.
- Technika analityczna SWOT polega na posegregowaniu posiadanej informacji o danej sprawie na cztery grupy (cztery kategorie czynników strategicznych):
 - **S (Strengths)** – mocne strony: wszystko to co stanowi atut, przewagę, zaletę analizowanego obiektu,
 - **W (Weaknesses)** – słabe strony: wszystko to co stanowi słabość, barierę, wadę analizowanego obiektu,
 - **O (Opportunities)** – szanse: wszystko to co stwarza dla analizowanego obiektu szansę korzystnej zmiany,
 - **T (Threats)** – zagrożenia: wszystko to co stwarza dla analizowanego obiektu niebezpieczeństwo zmiany niekorzystnej.
- mocne strony i słabe strony to czynniki zależne od nas (te, na które mamy wpływ planistyczny i zarządczy), a szanse i zagrożenia, to czynniki obiektywne, na które nie

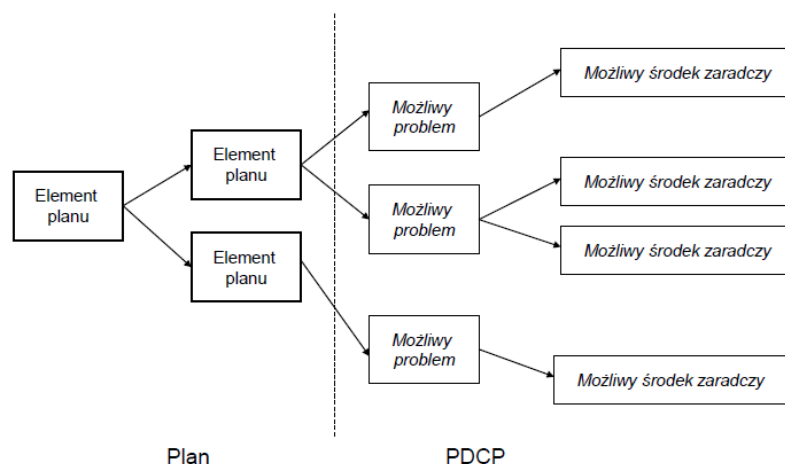
mamy bezpośredniego wpływu sprawczego.

Lista kontrolna ryzyk: 7 podkategorii zagrożeń wg Pressmana

1. Ryzyko związane z rozmiarami systemu informatycznego.
2. Ryzyko związane z niską jakością procesu konstrukcji/wdrożenia systemu oraz niskim postrzeganiem jego definicji.
3. Ryzyko związane z niedostateczną dostępnością lub jakością narzędzi wspomagających prowadzenie projektu
4. Ryzyko związane z nowością i stopniem skomplikowania rozwiązań technologicznych oraz systemu
5. Ryzyko związane z niewystarczającymi co do ilości lub doświadczenia zasobami ludzkimi
6. Ryzyko o charakterze biznesowym
7. Ryzyko wynikające ze specyfiki klienta

Diagram procesu decyzyjnego: Process Decision Program Chart (**PDPC**)

- Jest metodą graficznej prezentacji alternatywnych działań dla realizacji planu w warunkach ryzyka.
- Używana przy tworzeniu planów, do identyfikacji potencjalnych zagrożeń w ich realizacji
- Po zidentyfikowaniu ryzyk, używana do określenia i wyboru zbioru możliwych środków zaradczych
- Używana do planowania sposobów unikania i eliminacji zidentyfikowanych ryzyk
- Daje największą wartość:
 - kiedy ryzyka nie są oczywiste,
 - gdy sytuacja nie jest znana,
 - w wypadku złożonych planów
 - gdy konsekwencje niepowodzeń są poważne
- **Miara ryzyka = prawdopodobieństwo wystąpienia x strata spowodowana ryzykiem**



Są trzy drogi postępowania z ryzykiem:

- **Unikanie ryzyka** – oznacza niepodjęcie działań, których wynikiem jest

zidentyfikowane ryzyko. Zazwyczaj obejmuje znajdowanie alternatywnych działań.

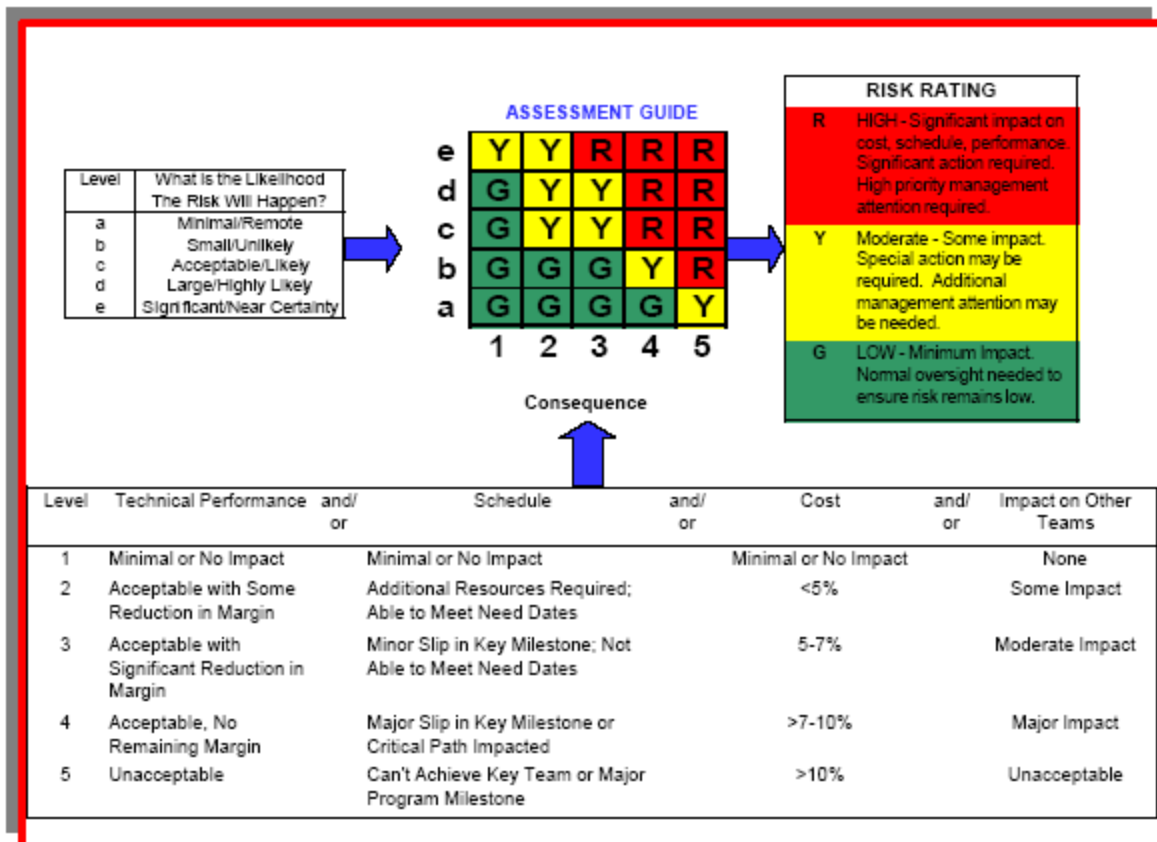
- **Obniżanie ryzyka** – oznacza podejmowanie takich działań, które obniżają, ale nie eliminują całkowicie zidentyfikowanego ryzyka. Obejmują dodatkowe działania, np. dodatkowe testy aby obniżyć ryzyko wystąpienia wad w oprogramowaniu. Ujemnym skutkiem są dodatkowe koszty obniżenia ryzyka.
- **Plany awaryjne** – nie obniżają prawdopodobieństwa wystąpienia ryzyka. Obejmuje tworzenie dodatkowych planów, tak, że gdy wystąpi ryzyko będziemy przygotowani i zdolni do kontrolowania sytuacji przy minimalnych kosztach i zakłóceniach.

Kryteria podejmowania decyzji przy tworzeniu diagramu PDPC:

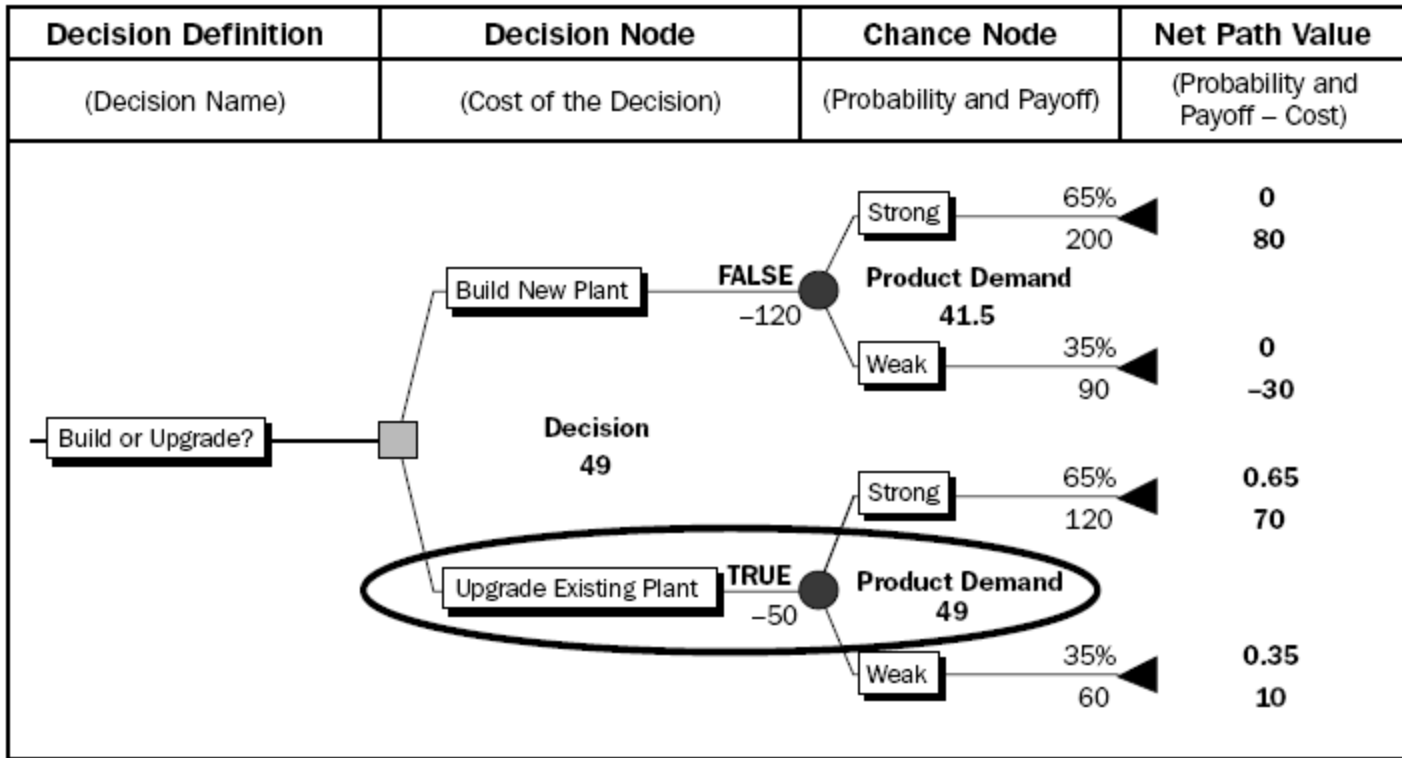
- **Czas.** Jak dużo czasu zabierze ryzyko? Czy leży ono na krytycznej ścieżce harmonogramu. Jak dużo czasu oszczędzi środek zaradczy?
- **Koszt.** Jaki będzie koszt wystąpienia ryzyka? Jaki będzie koszt środka zaradczego? Czy warto go podejmować?
- **Kontrola.** Jak dużo masz kontroli, by zapobiec ryzyku? Jaką będziesz miał kontrolę, gdy ono wystąpi? Jak mógłbyś to zmienić?
- **Informacja.** Jak dużo wiesz o ryzyku? Jakie są symptomy, że ono nadchodzi?

Techniki jakościowej analizy ryzyka

- Prawdopodobieństwo i wpływ ryzyka.
- Macierz szacowania ryzyka (prawdopodobieństwo/wpływ)



- Testowanie założeń projektu
- drzewa decyzyjne



- Ocena dokładności danych - sprawdzanie
 - ocena do jakiego stopnia dane o ryzyku są przydatne do zarządzania ryzykiem
 - zakres zrozumienia ryzyka
 - dostępne danych o ryzyku
 - jakość danych
 - pewność i integralność danych

44. Zarządzanie zmianami i konfiguracjami oprogramowania.

Change & Releases Management

Zmiany w zasobach informatycznych wiążą się z ryzykiem spadku wydajności infrastruktury informatycznej i poszczególnych aplikacji, a nawet rozległymi awariami prowadzącymi do przestojów w działalności firmy. Zarządzanie zmianą, konfiguracją i wersjami zasobów IT w firmie zapewnia usprawnienie obsługi aplikacji i infrastruktury informatycznej, poprzez ujednolicenie zasad i procesów, poprawę dostępności usług i ograniczenie ryzyka związanego ze zmianami w ramach zasobów IT.

Zarządzanie Zmianą (Change Management)

Proces Zarządzania Zmianą pozwala na wdrożenie i stosowanie ustandaryzowanych procedur dla wszystkich zmian w ramach zasobów IT. Zapewnia skuteczność i efektywność wprowadzanych zmian, a także równowagę między potrzebą zmiany oraz ryzykiem negatywnego wpływu zmiany na wydajność usług IT.

Wdrażając w przedsiębiorstwie proces Zarządzania Zmianą zyskujemy możliwość:

- Szacowania korzyści i ryzyka proponowanych zmian,
- Dokładniejszego oszacowania kosztów zmian (przed ich wprowadzeniem),
- Zarządzania testami i implementacją zmian,
- Monitorowania i raportowania implementacji zmian,
- Zmniejszenia niekorzystnego wpływu zmiany na inne usługi IT,
- Zwiększenia dostępności usług IT dzięki kontroli nad całym procesem zmiany.

Zarządzanie Konfiguracją (Configuration Management)

Zarządzanie aplikacjami i konfiguracją różnych systemów IT w infrastrukturze fizycznej lub wirtualnej jest jednym z najtrudniejszych wyzwań stojących przed działami IT. Proces Zarządzania Konfiguracją dostarcza pełnej informacji na temat wszystkich elementów wchodzących w skład zasobów IT przedsiębiorstwa i relacji między nimi (Bazy Konfiguracji CMDB) poprzez ich identyfikację i opisanie. Kontrola danych konfiguracyjnych zasobów informatycznych jest jednym z kluczowych elementów procesu zintegrowanego zarządzania usługami IT.

Wśród najważniejszych zadań procesu Zarządzania Konfiguracją znajdują się:

- Możliwość zliczania i kontrola wszystkich zasobów IT,
- Identyfikowanie wszystkich elementów konfiguracji zgodnie z przyjętym poziomem szczegółowości,
- Dostarczanie dokładnych informacji na temat elementów konfiguracji i ich dokumentacji,
- Umożliwienie uwidocznienia zmian w oprogramowaniu i konfiguracji sprzętu oraz wspieranie i usprawnianie procesu Zarządzania Wersją,
- Poprawa bezpieczeństwa poprzez kontrolowanie wersji elementów konfiguracji będących w użyciu.

Celem zarządzania konfiguracją oprogramowania jest planowanie, organizowanie, sterowanie i koordynowanie działań mających na celu identyfikację, przechowywanie i zmiany oprogramowania w trakcie jego rozwoju, integracji i przekazania do użycia.

Każdy projekt musi podlegać konfiguracji oprogramowania. Ma ono krytyczny wpływ na jakość końcowego produktu. Jest niezbędne dla efektywnego rozwoju oprogramowania i jego późniejszej pielęgnacyjności.

ZKO jest szczególnie ważne, jeżeli projekt może toczyć się przez wiele lat, jeżeli cel lub wymagania na oprogramowanie są niestabilne, jeżeli oprogramowanie może mieć wielu użytkowników, i/lub jeżeli oprogramowanie jest przewidziane na wiele platform sprzętowo-programowych.

W takich sytuacjach złe zarządzanie konfiguracją oprogramowania może całkowicie sparaliżować projekt.

ZKO powinno zapewniać, że ...

- Każdy komponent oprogramowania będzie jednoznacznie identyfikowany;
- Oprogramowanie będzie zbudowane ze spójnego zestawu komponentów;
- Zawsze będzie wiadomo, która wersja komponentu oprogramowania jest najnowsza;
- Zawsze będzie wiadomo, która wersja dokumentacji pasuje do której wersji komponentu oprogramowania;
- Komponenty oprogramowania będą zawsze łatwo dostępne;
- Komponenty oprogramowania nigdy nie zostaną stracone (np. wskutek awarii nośnika lub błędu operatora);
- Każda zmiana oprogramowania będzie zatwierdzona i udokumentowana;
- Zmiany oprogramowania nie zaginą (np. wskutek jednoczesnych aktualizacji);
- Zawsze będzie istniała możliwość powrotu do poprzedniej wersji;
- Historia zmian będzie przechowywana, co umożliwi odtworzenie kto i kiedy zrobił zmianę, i jaką zmianę.

Wszystkie elementy projektu i oprogramowania muszą być przedmiotem ZKO, w szczególności:

- dokumentacja: wymagań, analityczna, projektowa, testowania, użytkownika, itd.
- moduły z kodem źródłowym, kody do konsolidowania, kody binarne,
- ekrany interfejsu użytkownika,
- pliki z danymi tekstowymi (np. komunikatami systemu), bazy danych, słowniki, itd.
- kompilatory, konsolidatory, interpretery, biblioteki, protokoły, narzędzia CASE, konfiguracje sprzętowe, itd.
- oprogramowanie testujące, dane testujące,
- serwery WWW wraz z odpowiednimi stronami HTML i oprogramowaniem,
- ...
- Wyróżnialny element uczestniczący w projekcie lub produkcie będzie określany jako „**pozycja konfiguracyj**”. Jest ona traktowana jako pojedynczy, możliwy do odseparowania komponent projektu lub produktu programistycznego.

45. Zasady tworzenia harmonogramów realizacji systemu.

Z “ZPI Trawinski 03 Planowanie przedsięwzięcia”:

Harmonogramowanie to wybór i zastosowanie najbardziej odpowiednich technik dla stworzenia programu lub sekwencji działań nakierowanych na dotrzymanie kluczowych terminów i celów projektu.

Obejmuje ono również rozpoznanie i uwzględnienie w programie:

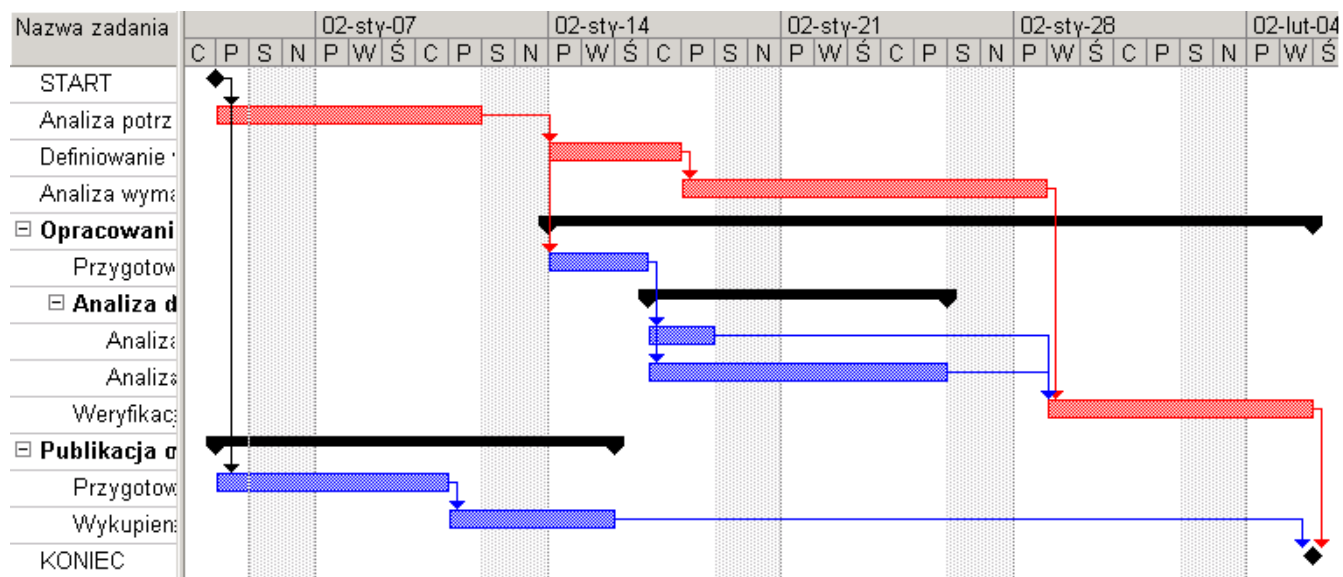
- etapów (faz), dat, punktów kontrolnych (kamieni milowych),
- zapotrzebowania i dostępności zasobów,
- sekwencji zadań i zależności pomiędzy zadaniami,
- limitów czasowych
- ograniczeń wewnętrznych i zewnętrznych

Etapy wykonywania harmonogramu

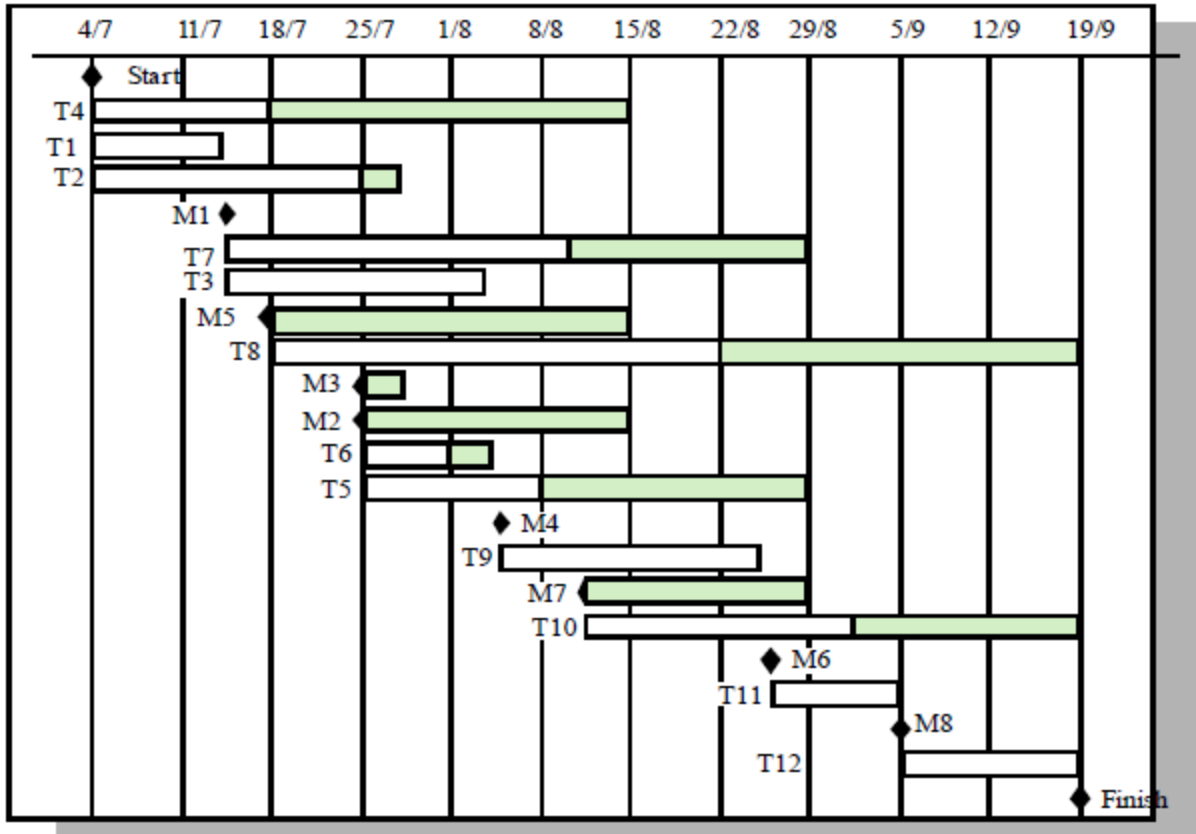
- Podział projektu na czynności
- Określanie zależności pomiędzy czynnościami
- Szacowanie czasu trwania czynności
- Ustalenie chronologii zadań i wyznaczenie tzw. „kamieni milowych”,
- Wykreślenie tzw. ścieżki krytycznej w postaci rysunku wektorowego - grafów.
- Tworzenie harmonogramu
- Kontrola harmonogramu

Techniki harmonogramowania:

- wykres Gantt'a
 - technika zaproponowana przez Henry'ego L. Gantta, przed I-szą wojną światową
 - graficzna prezentacja
 - początkowej chwili realizacji zadania
 - czasu trwania poszczególnych zadań
 - sekwencji zadań (logika prowadzonych prac)
 - systematyczne uzupełnianie informacji o realizacji prac wskazuje na stan zaawansowania projektu
 - Używane symbole:
 - Trójkąty – kamienie milowe (zerowy czas trwania)
 - Czarne prostokąty – zsumowane czynności
 - Jaśniejsze prostokąty – czynności
 - Strzałki – zależności między czynnościami

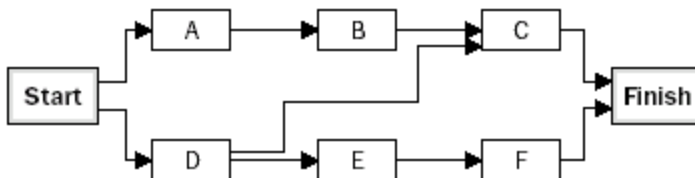


- wykresy paskowe:

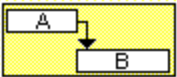
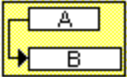
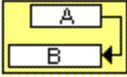
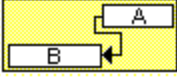


©Ian Sommerville 2000: Inżynieria oprogramowania. Rozdział 4.

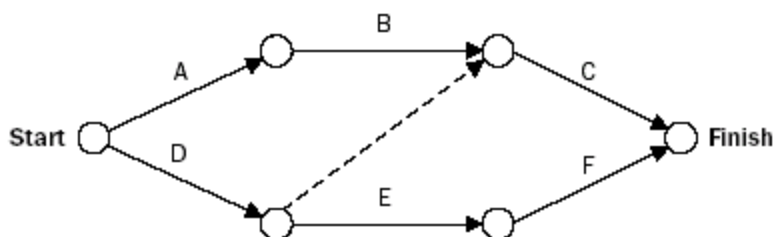
- Notacja graficzna służąca do przedstawiania harmonogramów
- Pokazuje podział przedsięwzięcia na zadania. Zadania nie powinny być duże. Tydzień albo dwa jest wystarczające.
- Wykres paskowy pokazuje harmonogram w terminach dni roku kalendarzowego
- diagramy sieciowe (zadanie na wierzchołku/łuku),
 - Precedence diagramming method (PDM)
 - prostokąty to czynności, a strzałki to zależności między nimi



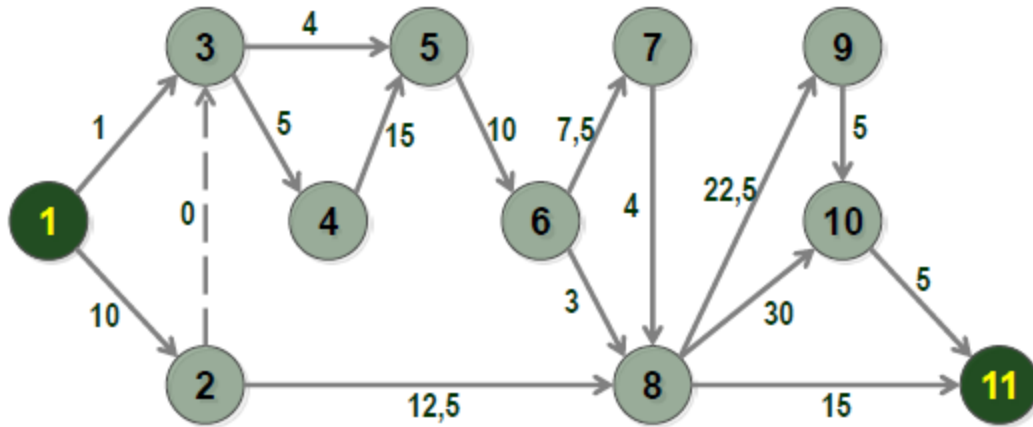
- zależności:

Task dependency	Example	Description
Finish-to-start (FS)		Task (B) cannot start until task (A) finishes.
Start-to-start (SS)		Task (B) cannot start until task (A) starts.
Finish-to-finish (FF)		Task (B) cannot finish until task (A) finishes.
Start-to-finish (SF)		Task (B) cannot finish until task (A) starts.

- Arrow Diagramming Method (ADM)
 - Strzałki to czynności, a węzły (kółka) to początkowe i końcowe punkty czynności
 - Można pokazać tylko zależność finish-to-start



- metoda ścieżki krytycznej (CPM - Critical Path Method),
 - proces sporządzania:
 - 1.Podział projektu na zadania i czynności
 - 2.Ustalenie logicznego następstwa poszczególnych czynności
 - 3.Określenie czasu trwania czynności
 - 4.Wykreślenie sieci
 - 5.Ustalenie najwcześniejszych możliwych i najpóźniejszych dopuszczalnych terminów wystąpienia zdarzeń
 - 6.Wyliczenie rezerw czasu
 - 7.Wykreślenie drogi krytycznej
 - 8.Interpretacja rezerw czasu
 - 9.Ewentualne udoskonalenie sieci (skrócenie ścieżki krytycznej) – powrót do punktu 4.
 - Ścieżka krytyczna – najdłuższy czas realizacji całego projektu



- Program Evaluation and Review Technique (PERT)
 - Wykorzystywana do szacowania czasu trwania projektów, kiedy występuje duży element niepewności
 - Wykorzystuje metody probabilistyczne, bazuje na optymistycznym, najbardziej prawdopodobnym i pesymistycznym czasie trwania projektu
 - Wady: dodatkowa praca (3 szacowania na czynność), są lepsze metody -> rzadko używana w praktyce
 - brak dokładnej informacji o czasie trwania poszczególnych zadań skłania do oszacowania tego czasu
 - doświadczony menedżer jest w stanie określić przedział czasu realizacji poszczególnych czynności (estymaty czasu)
 - najwcześniej jak to tylko możliwe (czas optymistyczny) – a
 - spodziewany czas zakończenia (czas najbardziej prawdopodobny) – m
 - najpóźniej jak to tylko możliwe (czas pesymistyczny) – b
 - czas trwania każdego zadania oszacowany jest za pomocą
 - oczekiwanego czasu t_i oraz wariancji v_i i odchylenie
 - standardowego σ_i

Oczekiwany czas realizacji zadania

$$t_i = \frac{a_i + 4m_i + b_i}{6}$$

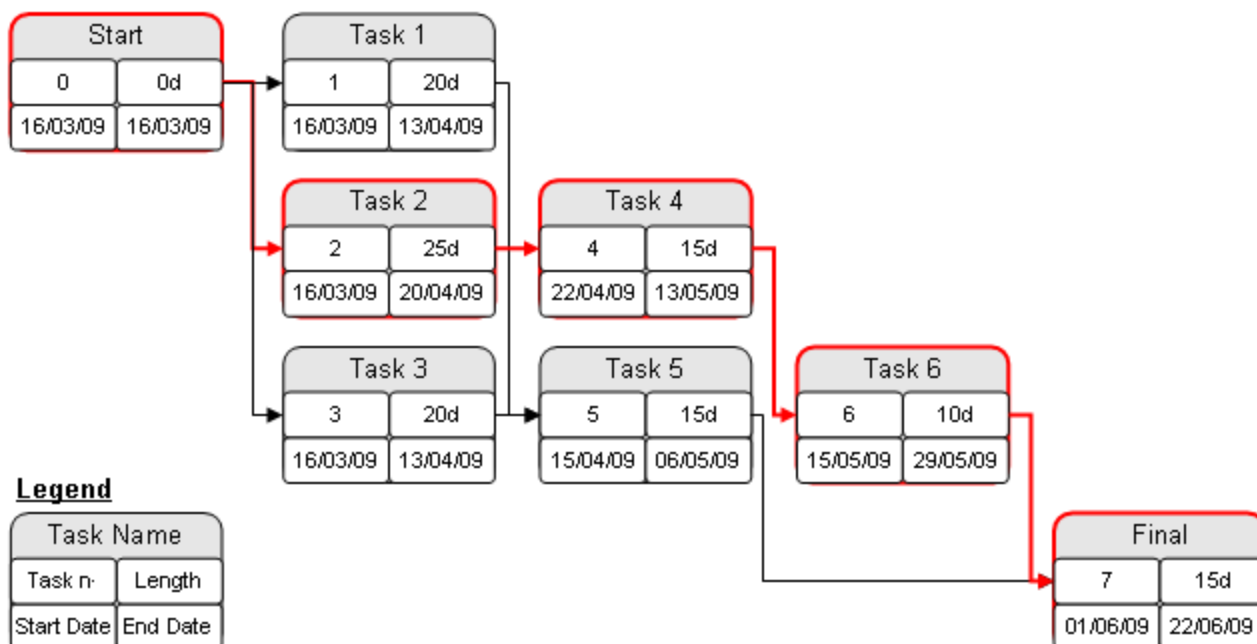
Wariancja czasu realizacji

$$v_i = \frac{(b_i - a_i)^2}{36}$$

Odchylenie standardowe

$$\sigma_i = \sqrt{v_i}$$

- Diagram PERT:



46. Zastosowanie sztucznej inteligencji w systemach informacyjnych.

Znalazłem takie opracowanie:

Zastosowanie sztucznej inteligencji w systemach ekonomicznych i w podejmowaniu decyzji
Sebastian Wyrwał

O sztucznej inteligencji mówi się dużo. Praktycznie każda osoba studiująca informatykę, czy kierunek pokrewny, słyszała o algorytmach genetycznych, czy systemach ekspertowych. Niektórzy jednak wciąż traktują zagadnienia związane ze sztuczną inteligencją z przysłowiowym przymrużeniem oka. Trzeba powiedzieć, że sztuczna inteligencja jest dziedziną obejmującą wiele zagadnień, również tych niepowiązanych bezpośrednio z programowaniem, czy projektowaniem systemów informatycznych. Poniżej czytelnik może znaleźć kilka zagadnień należących, bądź związanych ze sztuczną inteligencją:

Algorytmy Genetyczne,
Programowanie Genetyczne,
Systemy ekspertowe,
Automatyczne pozyskiwanie wiedzy,
Inżynieria wiedzy.

Metody sztucznej inteligencji znajdują zastosowanie:

W medycynie,
Do rozpoznawania obrazów,
Jako metoda optymalizacji,
Do wspomagania decyzji,
Do pozyskiwania wiedzy z baz danych,
W zarządzaniu,
Do projektowania układów elektronicznych,
Do obróbki obrazu,
W ekonomii i ekonometrii,
W grach logicznych,
Do klasyfikacji.

Przydatność algorytmów genetycznych jako metody optymalizacji nie jest kwestionowana np. w przypadku funkcji, których minimum nie można wyznaczyć tradycyjnymi metodami matematycznymi.

//wtrącę się ze swoim opracowaniem z <http://www.cs.put.poznan.pl/amichalski/wsi/A11.rgb.pdf>
Zastosowania SI

- **Gry** – szachy, warcaby
- **Automatyczne wnioskowanie i dowodzenie twierdzeń**, projektowanie i weryfikacja układów cyfrowych i programów komputerowych
- **Systemy eksperckie** - rolnictwo, handel, wojsko, przemysł, nauka, medycyna
- **Automatyczne (maszynowe) uczenie się** – systemy oparte na bazach wiedzy, eksploracja i odkrywanie wiedzy z dużych baz danych
- **Planowanie działań i robotyka** – systemy planowania podróży, automatyczne linie produkcyjne, sterowanie robotów, szeregowanie procesów produkcji
- **Przetwarzanie języka naturalnego** – rozpoznawanie mowy i pisma, tłumaczenie
- **Rozpoznawanie obrazów** – robotyka, zautomatyzowana produkcja, diagnostyka medyczna, nawigacja, kryminalistyka

//koniec wtrącenia

Systemy ekonomiczne, podejście tradycyjne i z zastosowaniem sztucznej inteligencji

Zastosowanie sztucznej inteligencji w systemach ekonomicznych i wspomagających decyzje kredytowe, bądź inwestycyjne jest dyskutowane, aczkolwiek nie negowane. Niespełna rok temu odbywała się prezentacja pewnego systemu służącego do oceny ryzyka przy udzielaniu kredytów. Pomimo tego, że system był bardzo rozbudowany, to do przewidywania wartości (lub mówiąc ściślej rozkładu) pewnych wskaźników ekonomicznych, na podstawie ich wartości historycznych używano prostych modeli matematycznych. Dlaczego? Na takie pytanie padła odpowiedź, że zastosowanie sztucznej inteligencji wymaga czasu i nabrania do niej większego zaufania ze strony nieufnych klientów jakimi są banki. Prawdopodobnie jedyną rzeczą z zakresu sztucznej inteligencji możliwą do zastosowania w tym systemie (choć do nieco innych celów) były sieci neuronowe.

Modele matematyczne polegają - w pewnym uproszczeniu - na przyjęciu pewnej zależności funkcyjnej. Ich zaletą jest to, że wynik otrzymujemy natychmiast, co jest istotne w systemach działających online. Wadą jest to, że należy przyjąć pewien model, czyli innymi słowy, przyjąć pewną zależność funkcyjną. Systemy ekonomiczne - czyli pełniące funkcję doradczą dla inwestorów giełdowych - można roboczo podzielić na dwie grupy:

systemy wspomagające wybór akcji, jakie powinny znaleźć się w portfelu.

Systemy do znajdowania zależności pomiędzy wskaźnikami giełdowymi i przewidywania ich wartości w przyszłości.

Trywializując można by powiedzieć, że giełda jest miejscem gdzie można - posiadając szczęście lub wiedzę - w łatwy sposób osiągnąć duże dochody. Trzeba tylko wiedzieć jakie będą ceny akcji lub przynajmniej znać tendencje zmian tych cen. Metod analizy jest wiele. Istnieją metody wykorzystujące modele matematyczne lub doświadczenie analityków. Niektórzy twierdzą, że ciągi notowań giełdowych są nieprzewidywalne... Przeszukując Internet można znaleźć wiele systemów wspomagających graczy giełdowych. Większość tych systemów jest dość droga, a ich cena, rzędu kilkuset dolarów, może odstraszać. W każdym wypadku trudne, o ile w ogóle możliwe, jest sprawdzenie jak taki system w rzeczywistości działa, tzn. jaką metodę stosuje.

Założmy, że ceny akcji pewnej hipotetycznej spółki Y były następujące:

Przyjmijmy, że mamy dzisiaj 11.03.2000, a my jesteśmy inwestorami posiadającymi kapitał 10000 zł. Stoimy przed dylematem jak zainwestować nasz kapitał. Notowania spółki wykazują tendencje zwyżkową. Jeżeli za posiadany kapitał kupimy 96 akcji spółki, a cena akcji w następnym notowaniu wyniesie 110 zł, to zarobimy nieco ponad 568 zł. (abstrahujemy od podatków itp.). Trywialne jest stwierdzenie, że zarobienie 6% kapitału w ciągu dwóch dni, praktycznie bez pracy, jest bardzo dobrym interesem. Oczywiście nie wiemy jakie notowania w praktyce będzie miała spółka Y 13 marca 2000.

Można założyć, że notowanie spółki Y, nazwijmy je, y zależy w pewien sposób od pewnego wektora X, czyli bardziej formalnie:

$$y(t) = f(X)$$

gdzie:

$X = [x_1, x_2, \dots, x_n]$ jest wektorem

pewnych parametrów

y(t) jest ceną akcji w notowaniu t

Celowo nie uzależniono wartości wektora X od czasu, pozostawiając to zagadnienie na razie otwarte.

Problem polega jednak na tym, że nie znamy ani parametrów wektora X ani zależności

funkcyjnej f . Rozkład notowań w czasie to akcjogram. Dla naszej hipotetycznej spółki jest on pokazany na rysunku 1. Jeżeli postać funkcji f i wektora X była by znana np.

$$y = 2.13 \cdot \text{WIG} + 6$$

gdzie:

y jest notowaniem spółki Y w notowaniu t

WIG jest to Warszawski Indeks Giełdowy w notowaniu t

Czyli bardziej formalnie:

$$f(x) = 2.13 \cdot x + 6$$

$$X = [\text{WIG}]$$

byłby to klasyczny model matematyczny. Funkcja f w tym wypadku jest dobrze znaną ze szkoły funkcją liniową,

$$y = ax + b$$

która jest najprostszym, ale stosowanym w praktyce w systemach, modelem matematycznym. Przy dzisiejszych możliwościach technologicznych system może przechowywać bardzo wielkie ilości różnych danych. Dane te mogą być wykorzystane do budowy i testowania modelu. Model musi być jednak zdefiniowany przez człowieka. System taki dobierze współczynniki tego modelu, w naszym wypadku będzie to a oraz b , ale nie wymyśli za nas postaci zależności funkcyjnej.

Programowanie genetyczne (PG)

Sztuczna inteligencja przychodzi z pomocą, kiedy nie da się zaproponować żadnego modelu matematycznego. Model może być jednak zbudowany przy pomocy metod sztucznej inteligencji bez udziału człowieka. Posiadamy pewne dane, których część (podzbiór) może być składowymi naszego wektora X . Należy podkreślić, że dużo łatwiej jest w praktyce zaproponować takie dane (niezbędne i tak w modelu matematycznym) niż zaproponować zależność funkcyjną. Można ponadto zaproponować bardzo dużą ilość tych danych i pozwolić aby system sobie wybrał z jakich będzie korzystał, budując zależność funkcyjną. Metodą, która może być tu zastosowana nazywa się Programowanie Genetyczne (PG). Programowanie Genetyczne jest metodą ewolucyjną. Oznacza to, że w pamięci maszyny hoduje się populacje osobników, które odpowiadają funkcjom-modelom. W trakcie działania systemu powstają nowe populacje. Jeżeli przetwarzanie wykonywane jest poprawnie, to wysoce prawdopodobne jest to, że w następnych populacjach będzie występować grupa coraz to lepszych osobników-modeli.

Można to przyrównać do ewolucji opisaną przez Darwina, która zakłada, że w trakcie ewolucji wydadzą potomstwo osobniki najbardziej przystosowane do warunków zewnętrznych. W programowaniu genetycznym wybiera się grupę najlepszych osobników z populacji, które wydadzą potomstwo. Idea ta jest szeroko stosowana w algorytmach genetycznych. Tam jednak

osobniki zazwyczaj reprezentują wartości funkcji, lub jej argumentów, a nie samą funkcję. W programowaniu genetycznym z punktu widzenia systemów ekonomicznych osobnik nie reprezentuje konkretnej wartości ale postać funkcji.

Drzewa oraz ich wartościowanie

Aby przybliżyć idee PG należy przypomnieć co to jest drzewo. Drzewo jest grafem nie zawierającym cykli, którego jeden wierzchołek jest wyróżniony i nazywany jest korzeniem. Na rysunku korzeń to czarna elipsa. Elipsy narysowane linią kropkowaną to liście.

Fragment P to prawe poddrzewo drzewa, fragment L to poddrzewo lewe. Poddrzewo jest również drzewem. P1 jest zatem prawym poddrzewem drzewa L. Drzewo można zatem określić jako korzeń, mający (zależną od drzewa) określoną liczbę poddrzew (być może zerową - pojedynczy węzeł to również drzewo). Drzewo jest zatem strukturą rekurencyjną. Drzewa są bardzo często stosowaną w informatyce strukturą danych. Poddrzewo P składa się z korzenia i dwóch liści, korzeń tego drzewa może reprezentować funkcję, a liście - argumenty tej funkcji, tak jak na rysunku 3. Mamy w tym wypadku wyrażenie $a+b$, nic nie szkodzi aby wyrażenie to potraktować jak funkcję dwuargumentową w postaci:

$$+ (a,b) = a+b$$

Dodawanie jest oczywiście funkcją dwuargumentową, a to czy używamy notacji $a+b$, czy $+(a,b)$ nie ma w zasadzie znaczenia. Ważne jest to, że drzewo odpowiada pewnemu wyrażeniu, które z kolei odpowiada funkcji. Za pomocą drzew możemy reprezentować dowolne funkcje. Dany węzeł ma tyle dzieci ile argumentów ma funkcja, którą on reprezentuje. Liście drzewa (na rysunku 2 węzły rysowane linią kropkowaną) reprezentują argumenty funkcji (lub stałe) reprezentowanej przez całe drzewo.

Użycie drzew w PG pozwala na łatwą reprezentację wyrażeń - funkcji, przy równoczesnej łatwości ich wartościowania i wykonywania na osobnikach operacji, mających na celu wyprodukowanie osobników wchodzących w skład populacji potomnej.

Wartościowanie ma na celu przypisanie osobnikowi-drzewu liczby określającej jego stopień przystosowania. Wartościowanie odbywa się rekurencyjnie, tzn. zakładając, że węzeł, który właśnie wartościujemy reprezentuje funkcję $+$, to wartościowanie tego węzła polega na obliczeniu sumy wartościowań jego dwóch dzieci. A więc jeśli dany węzeł reprezentuje stałą, to wartościowaniem tego węzła jest po prostu wartość tej stałej. Jeżeli węzeł reprezentuje funkcję,

to wartościowanie polega na wartościowaniu poddrzew reprezentujących argumenty tej funkcji, a następnie obliczeniu wartości funkcji dla tak obliczonych argumentów. Przyjmijmy że D jest całym drzewem, L jest lewym poddrzewem D, P jest prawym poddrzewem D, a f jest funkcją (dwuargumentową) zawartą w korzeniu D. Wartościowanie drzewa D wygląda następująco: $\text{Var}(D) = f(\text{Var}(L), \text{Var}(P))$

Wartościowanie poddrzew przeprowadzamy w ten sam sposób.

Przykład:

Dla prostego drzewa z rysunku 3 wartościowanie dla $a=1$ i $b=2$ przebiegać będzie następująco:

$$\text{Var}(D) = +(\text{Var}(a), \text{Var}(b))$$

czyli:

$$\text{Var}(D) = \text{Var}(a) + \text{Var}(b) = a + b$$

dla powyższych wartości a oraz b
mamy:

$$\text{Var}(D) = a + b = 1 + 2 = 3$$

Dla drzewa z rysunku 4 i następujących wartości $a=1$, $b=2$, $c = P/2$ mamy:

$$\text{Var}(D) = +(\text{Var}(a), \text{Var}(P)) = a + \text{Var}(P)$$

$$\text{Var}(P) = *(b, \text{Var}(P1)) = b * \text{Var}(P1)$$

$$\text{Var}(P1) = \sin(\text{Var}(c)) = \sin(c)$$

Składając wyrażenia "od dołu" w jedno otrzymujemy:

$$\text{Var}(P) = b * \sin(c)$$

$$\text{Var}(D) = a + b * \sin(c)$$

Przyjmując wartości jak powyżej otrzymamy:

$$\text{Var}(D) = 1 + 2 * \sin(P/2) = 3$$

gdzie:

D całe drzewo

P prawe poddrzewo drzewa D

P1 prawe poddrzewo drzewa P

Wartościowanie drzewa odpowiada zatem obliczeniu wartości funkcji reprezentowanej przez całe drzewo, dla danych wartości argumentów tej funkcji.

Przykład:

Jeżeli drzewo reprezentuje funkcję

$f(a,b,c) = a+b*c$, wtedy wartościowanie dla $a=1$, $b=2$, $c=3$ odpowiada obliczeniu wartości funkcji $f(1,2,3) = 7$.

Wartościowanie drzew jest niezbędne do określenia przystosowania osobnika (reprezentowanego przez to drzewo). Na podstawie przystosowania poszczególnych osobników w populacji wybiera się osobniki, które będą miały potomstwo.

Dlaczego wartościujemy - czyli o uczeniu

W wielu metodach sztucznej inteligencji, w tym w PG, potrzebne jest uczenie. Załóżmy, że chce się przewidywać wartości akcji spółki Y, uzależniając cenę akcji wyłącznie od jednej zmiennej generowanej przez system. Niech zmienna ta będzie umownym numerem notowania. Dysponujemy ciągiem 10 poprzednich notowań, tak jak pokazano w tabeli 1. Weźmy przykładową populację drzew w systemie (tutaj zakładamy populację 5-ciu drzew, ale w praktyce przetwarza się populacje złożone z dziesiątek tysięcy drzew), które odpowiadają funkcjom przedstawionym w tabeli. Aby określić przystosowanie poszczególnych osobników reprezentowanych przez drzewa należy dokonać wartościowania tych drzew dla wszystkich punktów uczących.

Tabela 2 Drzewa w przykładowej populacji i odpowiadające im funkcje
DRZEWO

Funkcja

D1 $f(x)=101$

D2 $f(x)=100+0,1*x$

D3 $f(x)=1$

D4 $f(x)=100+(x-4,5)^2$

D5

$f(x) = 100+x^2$

Dla wszystkich drzew z populacji dla każdego punktu uczącego (notowania) należy obliczyć różnicę pomiędzy wartościowaniem w tym punkcie a rzeczywistą wartością notowania akcji w tym punkcie tj. błąd, który jest określony następująco.

$B_p = |W_p - \text{Varp}(D)|$

gdzie:

B_p błąd w punkcie uczącym p ,

W_p rzeczywista wartość ceny akcji w punkcie p ,

$\text{Varp}(D)$ wartościowanie drzewa

D w punkcie p ,

p punkt uczący, czyli notowanie.

Na rysunku 5 przedstawiono błędy dla punktów uczących (kolejnych notowań) dla drzew z populacji; widać, że najmniejsze błędy są dla drzew D1 i D2, nieco większe błędy są dla drzewa D4. Biorąc pod uwagę sumy błędów dla wszystkich punktów uczących możemy uszeregować drzewa wg rosnących błędów:

D1, D2, D4, D5, D3 Można ogólnie powiedzieć, że jest bardzo mało prawdopodobne aby drzewo D3 miało potomka, jak również jest wysoce prawdopodobne, aby drzewa D1 i D2 miały potomków, aczkolwiek istnieją różne metody wyboru drzew do reprodukcji. Niezbędne jest zawsze jednak wartościowanie drzew. Przystosowanie drzew formalnie określa funkcja przystosowania zwana funkcją fitness. Często jest to suma wartości błędów dla wszystkich punktów uczących.

Więcej o PG, czyli jak to działa

Budując lub konfigurując system określamy jak, a raczej z czego będą budowane drzewa. W programowaniu genetycznym określa się zbiór funkcji F oraz zbiór symboli terminalnych T . Funkcje ze zbioru F będą węzłami w budowanych drzewach, a symbole terminalne ze zbioru T , odpowiadające argumentom funkcji reprezentowanej przez całe drzewo lub stałym, będą liśćmi drzew.

Przykład:

Jeżeli w systemie zbory F i T będą określone następująco:

$F=\{+, *, \sin\}$

$T=\{a, b, c\}$

możliwe jest wygenerowanie drzewa jak na rysunku 4. Nic oczywiście nie stoi na przeszkodzie aby zbiory te zawierały więcej funkcji i symboli terminalnych tzn.:

$F=\{+, -, *, \sin, \cos, \log\}$

$T=\{a, b, c, d, e, f, g\}$.

gdzie:

+ oznacza dwuargumentową funkcję, która zwraca sumę swoich argumentów

- oznacza dwuargumentową funkcję, która zwraca różnicę swoich argumentów

* oznacza dwuargumentową funkcję, która zwraca iloczyn swoich argumentów

sin oznacza jednoargumentową funkcję sinus

cos oznacza jednoargumentową funkcję cosinus

log oznacza jednoargumentową funkcję logarytm dziesiętny

a,b,c są argumentami funkcji reprezentowanej przez całe drzewo

Siła programowania genetycznego polega na tym, że nawet z niezbyt licznych zbiorów F oraz T

wygenerować można bardzo dużo różnych drzew. Liczba możliwych do wygenerowania drzew wzrasta bardzo szybko wraz ze wzrostem liczności zbiorów F i T . Przy definiowaniu funkcji należy zachować ostrożność. Jak wiadomo, w matematyce, argumentem funkcji logarytm może być tylko liczba dodatnia. Wartość logarytmu dla liczby zero lub mniejszej od zera jest nieokreślona. W PG należy nieco "ulepszyć" definicję funkcji logarytm w stosunku do tej znanej z matematyki.

Nasz logarytm musi "akceptować" także wartość 0 i wartości mniejsze od zera. Można to zapewnić definiując funkcję logarytm tak aby dla argumentu równego 0 zwracała 0, a dla innych logarytm wartości bezwzględnej argumentu. Postępowanie takie ma na celu zapewnienie domkniętości zbioru funkcji F . Domkniętość polega na tym, że każda funkcja z F akceptuje dowolną wartość uzyskaną w wyniku wartościowania dowolnego drzewa zbudowanego z węzłów określonych w zbiorach F i T . Ponadto zbiór F musi posiadać własność dostateczności, która zapewnia, że rozwiązanie może być wyrażone za pomocą funkcji z F . Teoretyczna własność domkniętości niestety nie wystarcza. W praktyce rozwiązanie musi być łatwe do wyrażenia przy pomocy funkcji ze zbioru F .

Przykład:

Jeżeli $F=\{*, /\}$ oraz $T=\{a,b\}$ to nie jest możliwe wygenerowanie drzewa odpowiadającego wyrażeniu $a+b$.

Jeżeli $F=\{+, *\}$ oraz $T=\{a,b\}$ to wygenerowanie drzewa odpowiadającego wyrażeniu $(a+b)^2$ jest możliwe ale mniej prawdopodobne niż w przypadku kiedy $F=\{+, *, \text{pow}\}$, jeżeli pow oznacza jednoargumentową funkcję zwracającą wartość argumentu. Można to sobie wyobrazić rozrysowując wyrażenie $(a+b)^2$ w postaci drzewa (zapisane oczywiście jako $(a+b) * (a+b)$) dla $F=\{+, *\}$, a następnie dla $F=\{+, *, \text{pow}\}$.

Teoretycznie można powiedzieć, że im bardziej "bogaty" jest zbiór funkcji tym lepiej. W trakcie przeprowadzonych eksperymentów okazało się jednak, że obecność niektórych funkcji w zbiorze F wyraźnie pogarsza wyniki.

PG posiada bardzo wiele różnych parametrów konfiguracyjnych mających wpływ na uzyskiwane wyniki. Poza określeniem zbiorów F oraz T , bardzo istotny jest rozmiar populacji, oraz określenie jak długo będzie wykonywane przetwarzanie, czyli mówiąc bardziej poprawnie, jakie jest kryterium zakończenia przetwarzania.

Początek przetwarzania, czyli wygenerowanie generacji początkowej

Przetwarzanie w PG rozpoczyna się wygenerowaniem generacji początkowej. Drzewa generacji początkowej generowane są automatycznie przez system. Zasadniczo istnieją trzy metody generowania drzew populacji początkowej:

full - metoda ta polega na generowaniu drzew o jednakowej, założonej z góry wysokości. Dopóki wysokość drzewa jest mniejsza od założonej bierze się symbole wyłącznie ze zbioru F, jeżeli wysokość drzewa jest równa założonej bierze się symbole ze zbioru T.

grow - metoda ta polega na generowaniu drzew o wysokości nie większej niż założona. Dopóki wysokość drzewa jest mniejsza niż założona bierze się symbole zarówno ze zbioru F oraz T. Jeżeli wysokość drzewa jest równa założonej bierze się symbole ze zbioru T.

ramped half and half - polega na zastosowaniu obu powyższych metod, każdej z prawdopodobieństwem 0.5.

Proces generowania drzewa rozpoczyna się od wybrania losowej funkcji ze zbioru F. Liczba poddrzew drzewa, którego korzeń reprezentuje tę funkcję jest równa liczbie argumentów tej funkcji. Następnie proces jest powtarzany rekurencyjnie (czyli dla każdego poddrzewa itd.) zgodnie z jedną z powyższych metod.

Właściwe przetwarzanie, czyli ewolucja

Po wygenerowaniu przez system populacji początkowej rozpoczyna się właściwe przetwarzanie. Trzeba pamiętać, że w PG przekształceniu podlega w większym stopniu cała populacja niż pojedynczy osobnik. Operacja reprodukcji polega na wybraniu z generacji osobnika, który zostanie następnie skopiowany bez żadnych zmian do generacji potomnej i poddany działaniu operatorów genetycznych. Do wybierania osobników do operacji reprodukcji używa się różnych metod selekcji. Metody selekcji opierają się na funkcji przystosowania osobnika. Wyróżniamy następujące metody selekcji:

proporcjonalna - prawdopodobieństwo reprodukcji danego osobnika jest równe stosunkowi przystosowania tego osobnika do sumy przystosowań wszystkich osobników w populacji.

rankingowa - polega na posortowaniu osobników wg wartości funkcji przystosowania, a następnie wyborze najlepszego.

turniejowa - polega na losowym wybraniu pewnej, określonej liczby osobników, a następnie wybraniu spośród nich najlepszego.

Najważniejszym operatorem genetycznym w PG jest krzyżowanie. Aby wyprodukować potomstwo potrzeba dwojga rodziców. Obydwoje rodzice są (najczęściej) wybierani wg metod selekcji omówionych powyżej. Produkcja potomstwa polega na wzięciu części z jednego rodzica i części z drugiego. U każdego z rodziców wybiera się losowo punkt i wymienia się poddrzewa wychodzące z tych punktów. W ten sposób powstaje dwóch potomków. Operator mutacji polegający na wprowadzeniu losowych zmian w strukturze drzewa ma w PG małe zastosowanie. Przetwarzanie jest oczywiście tak prowadzone, aby wielkość populacji w poszczególnych generacjach nie ulegała zmianie. Można przetwarzać przez zadaną z góry liczbę generacji (np. 20, 40, lub 100), albo dopóki nie osiągnięty zostanie zadowalający wynik. Wynikiem przetwarzania w sensie znajdowania zależności funkcyjnej jest na ogół najbardziej przystosowany osobnik, który kiedykolwiek się pojawił. Można zapytać czy zastosowanie PG daje lepsze wyniki niż ślepe przeszukiwanie. Trzeba zdać sobie sprawę, że przestrzeń przeszukiwania składająca się ze wszystkich możliwych do wygenerowania drzew z założonych zbiorów F oraz T jest bardzo duża i znacznie większa niż ilość wszystkich

szacowań funkcji fitness podczas całego przetwarzania (równa, co oczywiste iloczynowi liczby generacji i ilości osobników w populacji). Zauważmy, że:

Rozwiązanie zostaje znalezione przy pomocy PG w n-tej, a nie w początkowej populacji. W trakcie przetwarzania wzrasta średnie przystosowanie populacji w kolejnych generacjach. W literaturze przedmiotu porównywano wyniki uzyskane przy użyciu PG oraz metod losowych. Dla tej samej (sumarycznej) liczby osobników przy pomocy ślepego przeszukiwania nie znaleziono nawet przybliżonego rozwiązania, podczas gdy przy użyciu PG problem udało się rozwiązać.

Zależność funkcyjna (czyli model) została znaleziona - co dalej?

Po znalezieniu modelu, czyli zależności funkcyjnej należy go przetestować. Testowanie polega na porównaniu wyników (np. przewidywania) uzyskanych przy pomocy modelu ze znanymi rzeczywistymi danymi. Zazwyczaj do uczenia wykorzystuje się podciąg uczący o długości $2/3$ ciągu danych a do testowania ciąg testowy o długości $1/3$ ciągu z danymi. Przykładowo jeśli do uczenia wykorzystano ciąg taki jak w tabeli 1 o długości 10, to do testowania modelu należy użyć następnych (nie ujętych w tej tabeli) pięciu notowań.