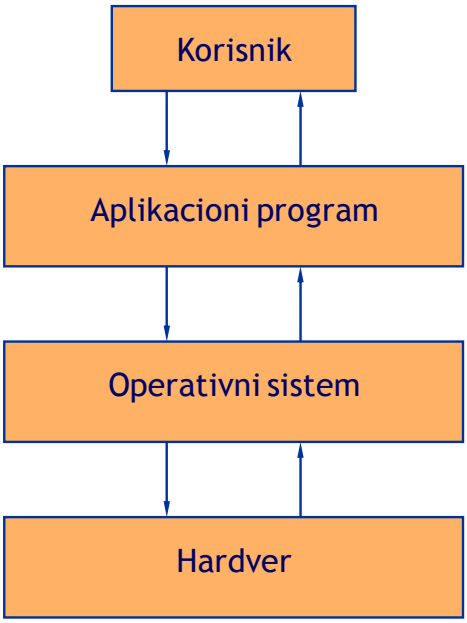
O

- Operativni sistem je skup rutina sistemskog softvera koji je smešten između aplikacionog programa i hardvera.
- Svi drugi softveri rade pod kontrolom OS-a, pristupaju hardveru preko OS-a poštujući pravila postavljena od strane OS-a.
- Pošto OS služi kao hardver/softver interface (posrednik), aplikacioni programeri i korisnici retko moraju komunicirati direktno sa hardverom, čime se pojednostavljuje programiranje.



- Rutine OS-a obavljaju ključne **funkcije podrške**, poput rada sa periferijskim uređajima i prihvatanje i obradu komandi korisnika. Operacije za komunikaciju sa periferijskim uređajima su uobičajene za većinu aplikacija. Pošto sve aplikacije pristupaju hardveru preko OS-a, takva centralna pozicija je idealan povod za pisanje ovakvih deljenih (zajedničkih) sistemskih rutina.
- Još jedna uloga OS-a jeste prevazilaženje problema rada na različitim konfiguracijama računara. Pošto različite konfiguracije zahtevaju različite hardver/softver interfejse, rutine OS-a koje komuniciraju direktno sa hardverom mogu biti vrlo različite, ali rutine koje komuniciraju sa aplikacionim programima predstavljaju konzistentnu platformu.

HARDVERSKE KOMPONENTE

- Računarski hardver se sastoji od:
  - Procesora
  - Glavne memorije
  - Ulazno/izlaznih uređaja

Matične ploče
- Ploča sa štampanim kolima
- Hardverske komponente koje obezbeđuju električno povezivanje između uređaja

PROCESOR
- Deo računara koji manipuliše sa podacima smeštenim u memoriji (RAM i keš) pod kontrolom programa, takođe, smeštenog u memoriji
  - Program - niz instrukcija, od kojih svaka zadaje računaru osnovnu instrukciju: sabiranje, oduzimanje itd. Svaka instrukcija ima kod operacije i jedan ili više operanada (adrese registara u kojima se nalaze podaci).
- 4 bitne komponente:
  - **jedinica za kontrolu instrukcija** (instruction control unit - ICU) ili upravljačka jednica - preuzima instrukcije iz memorije

- **aritmetičko-logička jedinica** - se sastoji kola za sabiranje, oduzimanje, množenje itd., dakle izvršava instrukcije
- **registri** - su privremena skladišta koja čuvaju kontrolne informacije, ključne podatke i međurezultate. Smešteni su na procesoru.
- **sat** - generiše vremenski određene elektronske signale koji sinhronizuju druge komponente. Jedna kompletna oscilacija električnog signala.

MEMORIJSKA HIJERARHIJA

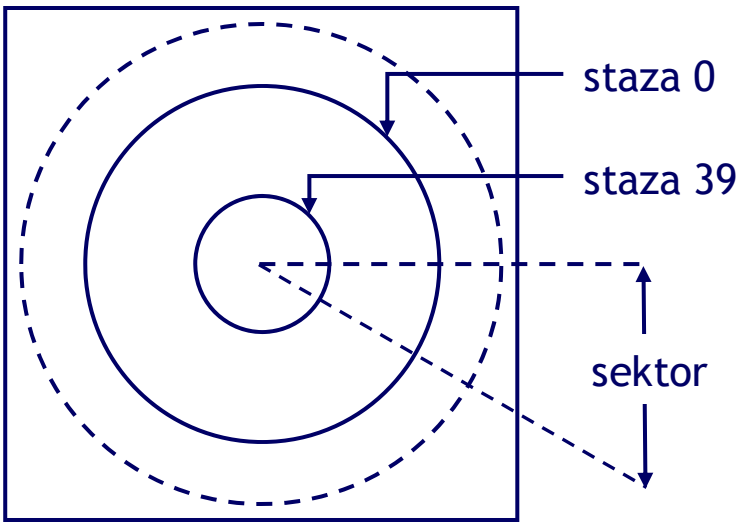
– Najbrža i najskuplja je na vrhu, sporija i manje skupa na dnu
  - Registri
  - L1 Keš
  - L2 Keš
  - Glavna memorija
  - Sekundarna i tercijarna memorija (CD-ovi, DVD-ovi i flopi diskovi)

- MEMORIJA (glavna, realna, fizička)
  - Sadrži trenutno aktivne programe i podatke
  - Adresibilnost memorije - svaka memorija je izgrađena kao skup memorijskih ćelija, gde svaka ćelija ima svoj jedinstven naziv - **adresu**. Podatak koji je smešten u memorijsku ćeliju naziva se sadržajem ćelije.
    RAM, ROM, KEŠ, virtuelne memorije

- SEKUNDARNA MEMORIJA
  - trajno čuvanje podataka, sporija, jeftinija, velikog kapaciteta
  - veće adresibilne jedinice
  - računar **ne može** izvršavati program koji se nalazi u sekundarnom skladištu

- HARD DISK
  - veći kapacitet, konstantna i brža rotacija, veća gustina,
  - skup staza – cilindar
  - najmanja adresibilna jedinica - **klaster**

Pristupanje podacima na čvrstom disku je sporije nego glavnoj memoriji
- Mehaničko kretanje glave za čitanje i pisanje
- Rotaciono kašnjenje
- Vreme prenosa



– Periferni uređaji
– Periferni uređaji je bilo koji uređaj koji processor ne zahteva da bi izvršio softverske instrukcije
– Izmenjiva sekundarna memorija koja olakšava backup i prenos
  - CDs, (CD_R, CD_RW)
    DVDs (DVD-R, DVD+R)
  - Zip diskovi
  - Meki diskovi
  - Fleš memorijske kartice
  - Trake

- Magistrala je skup mikroveza
  – Mikroveze su tanke električne veze koje transportuju informacije između hardverskih uređaja
  – Port je magistrala koja povezuje tačno dva uređaja

Direktan pristup memoriji (Direct Memory Access) Metod prenosa podataka od uređaja do glavne memorije preko kontrolera koji generiše prekid samo kada se prenos završi.

.

Bafer (buffer) je privremena memorijska oblast koja koja drži podatke za vreme I/O prenosa. Spooling je tehnoka baferovanja u kojoj se između procesora i uređaja I sporog U/I uređaja.

KOMPONENTE OPERATIVNOG SISTEMA

Interakcija između aplikacija i operativnog sistema

- **Korisnički inteface** - mehanizam za komunikaciju aplikacionih programa i OS-a i podršku zahtevima OS-a.

- Tipične komponente operativnog sistema uključuju:
  – Raspoređivač procesora (processor management, processor scheduler)
    Omogućava efikasno raspoređivanja procesorskog vremena.
  – Upravljač memorijom (memory management)
    Upravlja glavom memorijom sistema kada računar radi, allociranja memorijskog prostora po potrebi i osiguravanja da se aplikacije ne mešaju.
  – I/O(Input/Output) upravljač (Input/Output Manager)
    Odgovoran je za kontrolu komunikacija za periferijskim uređajima istema.
  – Upravljač međuprocesnim komunikacijama (IPC) (Interprocess Communication)
    Upravlja komunikacijom između procesa.
  – Upravljač sistemom datoteka (File System)
    Sadrži rutine koje omogućavaju korisniku da kreira, briše, menja i, uopšte, radi sa datotekama prema njihovom imenu.

Sistemski poziv (system call)Poziv korisničkog procesa kojim on poziva servise kernela.

Drajveri uređaja (device driver) Softver preko kojeg kernel interaguje sa hardverskim uređajima.

U zavisnosti kako su i gde smešetene komponente OS razlikuju se različite arhitekture OS (Monolitna, Mikrokernel, Slojevita, Klijent/Server)

- Korisnička interakcija sa operativnim sistemom
  – Često, preko određenih aplikacija zvanih *shell* (ljuska).
  – *Kernel* (jezgro)
    - Softver koji sadrži jezgro operativnog sistema

- **Komandni procesor ili shell**
  - Modul OS-a koji prihvata, interpretira i realizuje zadate komande se naziva komandni procesor. Komandni procesor se može predstaviti kao shell (kora) oko OS-a. Programeri i korisnici komuniciraju sa shell-om zadavanjem komandi, a on ih interpretira i prosleđuje OS-u.
- Postoji nekoliko tipova:
  - shell koji radi sa linijskim komandama (shell standardnog MS-DOS-a i UNIX-a)
  - Grafički korisnički interfejs
  - Shell koji omogućava govorno zadavanje komandi.
- Komandni jezik
  - Unutar komandnog procesora postoji veliki broj rutina koje izvršavaju pojedinačne zadatke.
  - Komandni procesor interpretira svaku komandu i predaje kontrolu odgovarajućoj funkcijskoj rutini. Skup datih komandi i njihova sintaksna pravila formiraju **komandni jezik**.
- Batch komande
  - većina OS-a ih podržava
  - skup komandi u jednom fajlu

Processor

- Zaštita i upravljanje memorijom
- Sprečava procese da pristupe memoriji koja im nije dodeljena

- Prekidi i izuzeci

Većina uređaja šalje signal nazvan *prekid* do procesora kada se dogodi neki događaj.

Tajmeri
  - Interval tajmer periodično generiše prekid

Satovi
  - Obezbeđuje merenje kontinuiteta

Početno učitavanje

- Početno učitavanje: puni inicialne komponente OS-a u memoriju
  - Izvršava se od strane računarovog osnovnog ulano/izlaznog sistema (BIOS)
    - Inicijalizuje sistemski hardver
    - Puni instrukcije u glavnu memoriju od regije sekundarne memorije nazvan početni (*boot*) sektor
  - Procesor izvršava početni kod
  - Procesor učitava OS sa diska u glavnu memoriju

Tehnologija uključi i radi (Plug and Play)
  - Omogućava da operativnim sistemima da konfigurišu novo instalirani hardver bez korisnikove interakcije.

Operativni sistemi prema tipu obrade podataka

U prvoj fazi računari nisu imali operativni sistem. Obavljali su poslove za koji su bili namenjeni; u jednom trenutku izvršavan je samo taj (jedan) posao.

U sledećoj fazi poslovi su obavljani uz angažovanje posebnog izvršioca – operatera, koji je aktivirao jedan posao, pa posle završetka tog posla sledeći sa liste čekanja. Aktiviranje posla se sastojalo od učitavanja programa i podataka a potom startovanja. Prisustvo operatera je bilo obavezno.

Paketna obrada. U ulazni uređaj je stavljano više poslova zajedno sa instrukcijama računaru kako da izvrši svaki od ovih poslova. Ove instzrukcije se zovu *komande za upravljanje poslom*. Pisane su jezikom za upravljanje poslom – JCL (Job Control Language), kojiliči na programski jezik. Čuvane su u posebnim datotekama.

Autonomna obrada (off-line). Dalje povećanje iskorišćenosti računara je postignuto smeštanjem bilo ulaznih, bilo izlaznih podataka na brže uređaje, kao što je disk. Sa ovog uređaja su poslovi efikasnije izvršavani u odnosu na paketnu obradu.

Raspoređivanje poslova. Kada su podaci i poslovi uneti autonomno, oni mogu da se izvršavaju u bilo kom redosledu. Operativni system ih izvršava u najboljem redosledu, što znači prema važnosti (hitnosti) posla.

Multi-programiranje. Računar istovremeno obavlja više poslova. Realna memorija sadrži više programa u jednom trenutku. Svaki program će se obrađivati određeni vremenski period. Kada će se i koliko obrađivati neki posao na procesoru je pod kontrolom upravljača procesora.

Neposredna obrada (on-line). Korisnik računara ima direktnu vezu sa računarom. Računaru se komada daje preko JCL. Poslove raspoređuje upravljač procesora.

Interaktivna obrada. Omogućava dijalog (konverzaciju) sa računarom. Na primer, pitanja i odgovori. Zahteva da se obrada izvrši u vrlo kratkom (ograničenom) vremenskom periodu.

Višestruki pristup. Na računar je povezano više terminala. Svi mogu da se koriste u isto vreme. Svaki od njih može da se uključi u neposrednu obradu. Ova kombinacija se često naziva *sistem za neposrednu obradu sa višestrukim pristupom*.

Obrada u realnom vremenu. Realno vreme je vreme koje nam je poznato. Kod prethodnih tipova obrada, procesor je obrađivao poslove u skladu sa najboljim iskorišćenjem vremena, nije bilo bitno koji je datum odnosno doba dana. Obrada u realnom vremenu zahteva da se procesor ni jednog trenutka ne zaustavi u odnosu na posao, odnosno da se ne može dodeliti drugom poslu. Najuobičajenija primena je u sistemima za upravljanje. (na primer, kontrola leta).

Distribuirana obrada. Skup udaljenih nezavisnih računara koji komuniciraju preko mreže i kooperativno izvršavaju određeni posao.

Procesi i niti

Process je program u izvršavanju.

Svaki proces ima svoj adresni prostor, koji se sastoji od tekst regije, regije podataka i stack regije.Stack regija sadrži instrukcije i vrednosti za pozive procedura. Proces prolazi kroz seriju diskretnih stanja (blokiran, spreman u izvršavanju itd.). Kada se process kreira OS mu dodelju7je identifikacioni broj I kreira blok za kontrolu procesa (Process Control Block).

Os obezbeđuje mehanizam za međuprocesnu komunikaciju (IPC Inter Process Comunication). (Na primer omogućavanje Web pretraživaču da pristupi podacima na serveru). Komunikacija se obavlja preko signala ili poruka.

- Nit je proces niže kategorije. Dele adresni prostor i druge globalne informacije sa svojim procesom
- Nitima može da upravlja operativni sistem ili korisnikova aplikacija

Razlozi za uvođenje niti:
- Izrada softvera

  - Prirodnije izražava nerazdvojive paralelne poslove
- Performanse
  - Bolja skalabilnost na višeprocesorske sisteme
- Saradnja
  - Deljivi adresni prostor troši manje resursa od IPC.

Životni ciklus niti (stanje, rođen, blokira, izvršavanje, itd.).

Niti kao i procesi imaju operacije (kreiranje, zaustavi, nastavi i td).

- Tri najpopularnija modela niti

  1. Niti na nivou korisnika
  2. Niti na nivou kernela
  3. Kombinacija niti na nivou korisnika i niti na nivou kernela

Konkurentno programiranje

Konkurentno izvođenje
Više od jedne niti postoji u sistemu odjednom.
Može da se izvodi nezavisno ili u saradnji.

Uzajamno isključivanje
Problem pristupanja dve niti podacima istovremeno.
Postoje različite tehnike za rešenje problema uzajamnog isključivanja.
(Algoritmi Dekker, Peterson, Lamport, hardverska rešenja, semafori, monitori).
Objektni jezici imaju rešenmje za konkurentno programiranje.

Zastoj (*deadlock*)
Proces ili nit čekaju na određeni događaj koji neće da se dogodi.
Većina zastoja se razvila zbog prirodnih sukoba za rezervisanjem resursa.
Kružno čekanje je karakteristika sistema sa zastojem.
Postoje algoritmi za sprečavanje zastoja (*Dijkstra Banker*).
Otkrivanje zastoja.
Oporavak od zastoja.

Vremensko planiranje procesora

Politika vremenske raspodele procesora
Određuje koji se procesi odvijaju u nekom trenutku.
Ciljevi vremenske raspodele (maksimizacija iskorišćenja procesora, završavanje procesa do krajnjeg datuma, maksimiranje propusne moći i td).
Kriterijum vremenskog planiranja (FIFO, Round-Robin (RR), Najkraći-Proces-Prvi i td).

Organizacija i upravljanje realnom memorijom

Upravljanje memorijom
Strategije za izbor optimalnih performansi memorije
Izvode se od strane upravljača memorije
Koji će procesi ostati u memoriji?
Kolikoj memoriji će moći da pristupi svaki proces?
U koji deo memorije će otići svaki proces?

Operativni sistem ima svoje zahteve za memorijom (na primer, Windows XP Professional minimalno 128 MB, preporučeno 256 MB).
Operativni sistem se štiti (granični registar - sadrži adrese gde počinje memorijski prostor za programe).
Kontinulano i nekontinualnom dodeljivanju memorije.
Preklapanje.

Programiranje u konačnim i promenjivim particijama.
- konačne particije: svaki aktivni proces prima konačno veliki blok memorije.
- promenljive particije: particije su velike koliko je potrebno.

Organizacija virtuelne memorije

Virtuelna memorija
- Nalazi se na sekundarnom memorijskom uređaju, najčešće disku.
- Rešava problem ograničenog memorijskog prostora.
- Stvara iluziju da postoji više memorije nego što je raspoloživo u sistemu
- Postoje dva tipa adresa u sistemima virtuelne memorije
  - Virtuelne adrese
    - Referencirane od strane procesa.
  - Fizičke adrese
    - Opisuju lokaciju u glavnoj memoriji.
- Jedinica za upravljanje memorijom (MMU)
  - Prevodi virtuelne adrese u fizičke adrese.

Mapiranje bloka
- Stranični blokovi fiksne veličine
- Segmenti blokovi varijabilne veličine
- Mapiranje stranica, segmenata i kombinovano mapiranje.

Upravljanje virtuelnom memorijom

Fetch strategija
- Određuje kada bi strane ili segmenti trebalo da se učitaju u glavnu memoriju.

Strategija zamene
- Tehnika koju sistem angažuje radi selekcije strana za zamenu kada je memorija puna.
- Određuje gde će u glavnoj memoriji da smesti stranu koja dolazi ili segment.
- postoji više strategija zamenea strane (FIFO, strana koja je provela, najviše vremena u memoriji bez pozivanja , strana koja je najmanje pozivana itd.)
- Postoji broblem definisanja veličine stranice (Intel/AMD Pentium 4 Athlon XP).

Optimizacija performansi diska

Merenje peformansi
- Vreme rotiranja diska
  - Vreme rotiranja podataka od tekuće pozicije do pozicije glave za čitanje i pisanje.
- Vreme traženja (Vreme pristupa cilindru)
  - Vreme kretanja glave za čitanje i pisanje do novog cilindra.
- Vreme prenosa
  - Vreme za koje glava za čitanje-pisanje učita/upiše sve željene podatke.
  - Vremena se mere u ms.

Postoje različite optimizacione strategije (Servisira zahteve u redosledu kako su dolazili, Servisira zahtev koji rezultira najkraćom razdaljinom traženja i td.).

Ostale tehnike za performanse diska (defragmentacija, kompresija, višestruke kopije podataka kojima se često pristupa i td.).

Redudantni skup nezavisnih diskova (Redundant Arrays of Independent Disks- RAID)
- Istovremen pristup više diskova radi povećanja propusne moći.

Sistem datoteka

Datoteke
- Imenovana kolekcija podataka kojom se manipuliše kao sa jedinicom.
- Smeštene na sekundarnim memorijskim uređajima.

Hijerarhija podataka

Informacije su memorisane u računarima u zavisnosti od hijerarhije podataka
Najniži nivo hijerahije podataka je sastavljen od bitova.
Šabloni bitova predstavljaju sve podatke od interesa u računarskom sistemu.
Sledeći nivo u hijerarhiji podataka su šabloni fiksne dužine bitova kao što su bajtovi, karakteri i reči.

Operacije nad datotekom kao celinom: otvaranje, zatvaranje, kreiranje, uništavanje, kopiranje, preimenovanje, listanje.

Pojedinačnim stavkama podataka u okviru datoteka može da se manipuliše sa operacijama kao što su: čitanje, pisanje, ažuriranje, brisanje.

Sistemi datoteka
Organizuju datoteke i upravljaju pristupom podacima.

Direktorijumi
Datoteke koje sadrže nazive i lokacije drugih datoteka, radi organizovanja i bržeg pronalaženja datoteka.

Pojedinačni nivo (ili flat) sistem datoteka i hijerarhijski sistem datoteka.
Pojedinačni nivo: smešta sve datoteke korišćenjem jednog direktorijuma.
Hijerarhijski sistem datoteka: koren označava gde na memorijskom uređaju počinje glavni direktorij.
Radni direktorijum: uprošćava navigaciju korišćenjem imena putanja.

Organizacija datoteka
Organizacija datoteka: način na koji su slogovi datoteke raspoređeni na sekundarnoj memoriji.
Najčešće organizacije su sekvencijane i indeksne.

Datoteke na sekundarnom uređaju mogu da budu raspoređene kontinualno i nekontinualno.

Zaštita integriteta podataka kod datoteka.
Rešava se odreživanjem prava pristupa.
U slučaju lomova koriste se rezervne kopije, dnevnik transakcija i RAID. RAID je redudantni skup nezavisnih diskova (**R**edundant **A**rray of **I**ndependent (or Inexpensive) **D**isks).

Serveri datoteka (klijenti šalju sve zahteve centralizovanom serveru) i distribuirani sistemi (datoteke smeštene na različitim računarima).

Performanse procesora

Ocenjivanje performansi je korisno kada se:
- Razvija sistem
- Planiraju kupovina ili nadgradnja
- Podešava sistem

Dva tipa merenja
- Merenje apsolutnih peformansi (na primer, propusna moć)
- Merenje relativnih performansi (na primer, lakoća korišćenja)

Tehnike ocenjivanja performansi

- Različite tehnike za različite namene
- Neke ocenjuju sistem kao celinu
- Neke izoluju performanse individualnih podsistema, komponenata, funkcija ili instrukcija

Neke od tehnika za merenje performansi

Trasiranje: registrovanje aktivnosti sistema, obično log korisnika.
Profili: registruju aktivnosti sistema u kernel modu.
Tajming: merenje izvornih performansi (na primer, broj ciklusa u sekundi).
Mikrobenčmark: merenje vremena koje je potrebno za izvršavanje specifičnih operacija operativnog sistema (na primer, kreiranje procesa).

Multiprocesorski sistemi

Multiprosesorski sistem
- Računar koji sadrži više od jednog procesora
- Koristi
- Povećava snagu obrade
- Odmerava korišćenje resursa prema zahtevima aplikacije

Multiprosesorski sistem se sastoji od čvorova i veza..
Postoje različite šeme spajanja procesora: deljiva magistrala, ukrštena matrica, oknasta mreža i td.

Multiprosesorski sistemi mogu da budu čvrsto spojeni preko zajedničke memorije ili labavo spojeni kada komuniciraju preko poruka.

Multiprosesorski sistemi prema odgovornosti za posao: jedan procesor jeglavni, odvojeni nezavisni procesori (labavo spregnuti), OS upravlčja grupom identičnih procesora.

Rad u mreži

Mrežna topologija
- Opisuje vezu između različitih host-ova
- Mogu biti: magistralne, prstenaste, oknaste, zvezdaste i td.
- Mogu biti lokalne (**L**ocal **A**rea **N**etwork - LAN) i velike (**W**ide **A**rea **N**etwork - WAN)

TCP/IP (**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol) - sastoji se od četiri logičkih nivoa nazvanih slojevima: aplikativni, transportni, mrežni sloj i sloj veze.

Distribuirani sistemi

Distribuirani sistemi
- Udaljeni računari rade zajedno preko mreže kao da se radi o jednoj mašini
- Korisnici imaju utisak da interaguju samo sa jednom mašinom
- Raspoređuju komunikacije i memorisanja kroz celu mrežu računara
- Aplikacije su sposobne da izvršavaju kod na lokalnim i udaljenim računarima i da dele podatke, datoteke i druge resurse između ovih računara

Pružaju niz prednosti: performanse, sigurnost, pouzdanost, skalabilnost.

Mrežni OS
- Pristupaju resursima na udaljenim računarima koji rade na nezavisnim operativnim sistemima

Distribuirani OS
- Upravljaju resursima lociranim na više umreženih računara.

Obezbeđuju interoperabilnost, tj. dozvoljava softverskim komponentama da i interaguju između različitih
- Hardeverskih i softverskih platformi
- Programskih jezika
- Komunikacionih protokola

Poziv udaljene procedure (**R**emote **P**rocedure **C**all - RPC) - omogućava procesu koji se izvodi na jednom računaru da poziva procedure u procesima koji se izvode na drugom računaru.

Poziv udaljenog metoda (**R**emote **M**ethod **I**nvocation – RMI) - omogućava Java procesu koji se zvodi na jednom računaru da pozove metod nekog objekta na udaljenom računaru korišćenjem iste sintakse kao poziv lokalnog metoda.

U distribuiranim sistemima postoji problem sinhronizacije događaja koji se rešava posebnim pravilima. Takođe postoji problem međusobnog isključivanja koji se rešava različitim sinhronizacionim metodama.

Klijent /server model

Troslojni sistem
- Nudi jasnije razdvajanje aplikacione logike od korisničkog interfejsa i podataka

Idealno, logika se nalazi na svom sopstvenom sloju
- Moguće na posebnim računarima
- Nezavisna od klijenta i podataka

Povećava fleksibilnost i proširljivost

Mrežni i distribuirani sistem datoteka

- Mrežni sistemi datoteka
  - Omogućavaju klijentima da pristupaju datotekama koje su smeštene na udaljenim računarima
- Distribuirani sistemi datoteka
  - Specijalni primeri mrežnih sistema datoteka koji omogućavaju transparentni pristup do udaljenih datoteka

- Distribuirani sistemi datoteka stvaraju iluziju o transparentnosti
  - Kompletna transparentnost lociranja datoteka znači da je korisnik nesvestan fižičkih lokacija datoteka u okviru distribuiranog sistema datoteka
  - Korisnik vidi samo globalni sistem datoteka

- Briga o zaštiti u distribuiranim sistemima
  - Garantovanje zaštite komunikacija
  - Kontrola pristupa

Klastering

- Klastering
  - Međusobno povezivanje čvorova (računari sa jednim procesorom ili sa više procesora) u okviru veoma brzog LAN-a, koji funkcionišu kao jedan paralelni računar.
  - Ciljevi mogu biti performanse, pouzdanost, uravnoteženje opterećenja.

Peer-to-Peer računarstvo

- Peer
  - Pojedinačni računar u P2P sistemu
  - Svaki izvršava i klijentske i serverske funkcije.

Grid Computing

- Grid Computing
  Povezuju računarske resurse koji su distribuirani u velikoj mreži (kao što su računari, memorije i naučni uređaji) radi rešavanja složenih problema.

Web servisi

- Web servisi
  - Zaokruženi skup odgovarajućih standarda koji mogu da omoguće računarskim aplikacijama da komuniciraju i razmenjuju podatke preko Interneta
  - Tehnologija i platforma nezavisni

.NET inicijativa

- Sadrži Visual Studio.NET integrisano razvojno okruženje
- Omogućava programerima da razviju Web servise u različitim jezicima, uključujući:
  - C++
  - C#
  - Visual Basic .NET

Kompjuterska zaštita

- Obuhvata pitanja sprečavanja neautorizovanog pristupa resursima i informacijama koje održavaju računari.
- Kriptografija: kodiranje i dekodiranje podataka tako da mogu da ih interpretiraju samo namenjeni primaoci.

Autentifikacija

Identifikovanje korisnika i akcija koje su dozvoljene za izvršavanje

Kontrola pristupa

- Ograničava ili limitira akcije koje mogu da budu izvršene nad resursima.
- Tehnike za kontrolu pristupa: matrice i liste.

Napadi na bezbednost: kripto-analitički napadi, virusi i crvi, napadi na odbijanje servisa, eksploatacija propusta u softveru, upadi u sistem.

Sprečavanje napada: zaštitni zidovi, sistemi za otkrivanje upada, anti-virusni softver, bezbednosne zakrpe, bezbedni sistem datoteka.

Programski jezici

Mašinski jezik

Računar razume samo svoj mašinski jezik, prema tome on je zavistan od računara. Mašinski jezik se sastoji od naziva i brojeva koji su predstavljeni isključivo preko jedinica i nula.

Asemblerski jezik

Uvodi instrukcije pomoću kratkih jednostavnih reči (na primer, ADD vred1, vred2). Instrukcije su bliže programeru (čoveku) ali su nerazumljive računaru. Asembler prevodi instrukcije u asemblerskom jeziku u mašinski jezik.

Viši programski jezici

- Lakši za korišćenje, instrukcije su slične engleskom jeziku.
- Zahtevaju kompajler za prevođenje višeg programskog jezika u mašinski jezik (prevodi celi program).
- Interpreteri prevode svaki red programa i odmah ga izvršavaju . prevođenje u jezik niskog nivoa koji nije mašinski jezik (COBOL, FORTRAN, BASIC, ALGOL).

- Program se piše u posebnoj datoteci koja se naziva izvorna datoteka, a sam kod izvorni kod (source code). Kompiliranje (compile) prevodi izvorni kod u mašinski kod. Produkt je objektni modul.
- Linkovanje (linking) je proces integracije programskog objektnog modula i različitih drugih modula (biblioteka) u pojedinačnu izvršnu datoteku.

Objektno orijentisano programiranje

Objektno orijentisano programiranje (OOP - Object-oriented programming) je razvijeno kao odgovor na sve složenije softverske zahteve. Koncept za rešavanje ovih zahteva se bazira na nezavisnim jedinicama programske logike i njihovoj ponovnoj upotrebi u softveru.

OOP može da se shvati kao skup povezanih opjekata koji zajedno rade. Svaki objekat je sposoban da prima i predaje poruke od/do drugih objekata i da obražuje podatke. Svaki objekat ima svoju ulogu i odgovornost.

Osnovni koncepti

Klasa je apstrakcija nekog objekta iz realnog sistema i uključuje njegove karakteristike (osobine, atribute) i ponašanje (metodi, operacije). Klasa je nacrt (šablon) koji opisuje prirodu nekih stvari (tvorevina). Ove stvari mogu biti fizički objekti ili neki konceti odnosno apstrakcije (Na primer, klase Student, Nastavnik, Dobri studenti).

Objekat je pojedinačna instanca (primerak) klase. Na primer, objekat Ana Nikolić je jedna instanca u klsi Studenti. vrednosti atributa u jednom objektu se nazivaju stanja. Objekat je nosilac ponašanja u svojoj klasi. Prema tome, klasa se može definisati kao skup objekata koji imaju iste atribute i ponašanje.

Metod predstavlja sposobnost objekta. U okviru programa samo jedan objekat izvršava metod.

Predaja poruka je proces u kome jedan objekat šalje podatke drugom objektu ili poziva drugi objekat da pozove metod.

Nasleđivanje je specijalizovana verzija klase koja nasleđuje atribute i ponašanje svoje nad klase (klase roditelja), ali ima i svoje sopstvene atrubute i ponašanja. Na primer, u klasi Studenti studenti svih smerova pripadaju toj klasi, ali smeru Informatika pripadaju samo studenti koji su upisani na odgovarajući program. Višestruko nasleđivanje je kada klasa nasleđuje osobine I ponašanja od više klasa.

Enkapsulacija je sakrivanje funkcionalnih detalja klase od objekata koji mu šalju poruke. Enkapsulacija se postiže specificiranjem koje klase mogu da koriste članice datog objekta. Svaki objekat izlaže nekoj drugoj klasi interfejs - one članice koje su dostupne toj klasi.

Apstrakcija je pojednostavljenje složene realnosti modelovanjem klasa koje odgovaraju problemu. To je kontrolisano izostavljanje karakteristika realnog objekta koje nisu bitne za rešavanje određenog problema.

Polimorfizam omogućava da se izvedene članice klase tretiraju kao članice njihove klase roditelj. To je sposobnost objekata koji pripadaju različitim tipovima podataka da odgovore na pozive metoda

Baze podataka

Po pravilu podaci se čuvaju u datotekama ili bazama podataka. Svaki od ovih načina ima određene prednosti od kojih je većina na strani baza podataka. Postoji više tipova baza podataka. Danas su dominantne relacione baze podataka zbog mnogih prednosti kao što su: povezivanje podataka, eliminisanje redudanse, nezavisnost logičke i fizičke strukture, višekorisnički rad, zaštita podataka i pojednostavljeno programiranje.

Baza podataka je skup međusobno povezanih objekata. Sama baza podataka je statički model nekog realnog sistema. Ovi objekti su predstavljeni u bazi podataka kao relacije (tabele). Relacija se sastoji od kolona (column) i vrsta (rows). Kolone označavaju atribute objekta, a u vrstama su smeštene njihove vrednosti. Postoji skup pravila koja treba svaka relacija da zadovolji, na primer da ne postoje dve identične vrste u istoj relaciji.

Jedna vrsta u relaciji može da se identifikuje pomoću ključa relacije. Ključ relacije čini jedan i ili više atributa koji jednoznačno određuju svaku vrstu. Ključevi imaju ulogu povezivanja relacija u relacionoj bazi podataka.

Baza podataka mora u svakom trenutku da ostane u ispravnom stanju. Ovaj zahtev je realizovan skupom ograničenja koja nazivamo pravilima integriteta. To znači da u svakoj koloni može da se unese isti tip podataka, da se ne mogu uneti dve kolone sa istom vrednošću ključa, kao i da u povezanim tabelama mora da budu konzistentna stanja.

Baze odataka raspolažu posebnim jezikom koji se naziva SQL (Structured Query Language). SQL nije proceduralan jezik. Raspolaže moćnim konceptima za sve operacije na podacima u bazi podataka. Za razliku od datoteka čiji je operand slog, kod relacionih baza podataka to je cela tabela.

Pri konstrukciji baze podataka se primenjuju različite metode koje treba da obezbede validnost izvršavanja SQL naredbi. Pod validnošću se podrazumeva eliminisanje anomalija u operacijama primenjenim nad bazom podataka. Ovaj postupak se naziva normalizacija.

U osnovi SQL naredbi stoji više operacija koje su sadržane u matematičkim oblastima (relaciona algebra i predikatski račun). Primena ovih operacija, pa samim tim i SQL je uslovljena određenim nivoom normalizacije relacija.

Naredbe u SQL su podeljene u više grupa kao što su naredbe za manipulaciju podataka (DML – Data Manipulation Language), naredbe za kreiranje i izmenu objekata baze podataka (DDL - Data Definition Language)), naredbe za definisanje prava pristupa, upravljanje transakcijama i td.

DML naredbe služe za upite nad bazom podataka, upisivanje novih podataka, izmene postojećih i brisanje podataka.

Korišćenje SQL je moguće preko odgovarajućih programa u okviru sistema za upravljanje bazama podataka ili u okviru programskih jezika (jezika domaćina, na primer, Java, C++, C# i td).

Baze podataka – nastavak

Pristup bazama podataka iz objektnih jezika se ostvaruje uz pomoć odgovarajućeg interfejsa (API). Poznati interfejsi su JDBC(Java Database Connectivity) i ADO.NET za .NET platformu.
Pristup bazama podataka je moguć i direktno korišćenjem odgovarajućih servisa sistema za upravljanje bazama podataka – SUBP (Data Base Management System). Primeri ovih servisa su Oracle SQL Plus i Microsoft Management Studio.

DDL naredbe se koriste za konstrukciju baze podataka i objekata baze podataka, ako što su tabele , trigeri, procedure. Procedura predstavlja određenu programsku celinu u kojoj pored SQL naredbi postoje i drugi tipovi naredbi (rad u petlji, grananja, aritmetičke operacije i td.). Trigeri su objekti koji pokreću određenu proceduru kada nastupi neki događaj. Na primer upisivanje vrste.

Pošto baza podataka podrazumeva istovremeni rad više korisnika, koji mogu da koriste iste podatke, potrebno je održati koegzistentno stanje u bazi podataka u svakom trenutku. To se postiže upravljanjem transakcija. Transakcije su jedinice posla (Job, Task) koje predstavljaju celinu. Transakcija može biti samo jedna naredba ili skup naredbi. Upravljanje transakcijama može biti implicitno (ostvaruje ga sistem za upravljanje bazama podataka) i eksplicitno (ostvaruje ga programer). Koegzistentno stanje je omogućeno izvođenjem paralelnih transakcija tako da se postignu isti efekti kao i da se ovo stanje dobija serijskim izvršavanjem transakcija (kao da se obavljaju jedna po jedna).

Za eksplicitno upravljanje transakcijama postoje naredbe za zaključavanje objekata ili delova objekata u BP, potvrđivanje transakcija (upisivanje iz bafera u glavnoj memoriji na disk), poništavanje transakcija i td.

Naredbe za upravljanje pristupom bazi podataka omogućavaju delegiranje određenih prava svakom korisniku BP nad objektima BP . Pod pravima pristupa se podrazumeva određivanje šta koji korisnik može da radi nad određenim objektom.

Sistem za upravljanje BP je složeni softverski proizvod koji omogućava:
- istovremeni rad više korisnika nad BP,
- oporavak BP u slučaju nepredviđenih prekida rada (pada sistema)
- koegzistentnost stanja u BP.

Pored toga SUBP raspolaže sa nizom servisa kao što su nadgledanje rada BP (praćenje performansi), servisi za uvoz i izvoz podataka, oporavak BP u slučaju pada sistema (na primer u slučaju prestanka napajanja računara).

Za upravljanje radom SUBP i kreiranje njene fizičke strukture je nadležno posebno lice – administrator BP. Fizičku strukturu BP čini više datoteka različitog tipa. Logičku strukturu BP čini njena šema.

Postoji više tipova baza podataka kao što su multimedijalne BP, distribuirane BP, Internet BP, XML BP i td.

Informacioni sistemi i softverski inženjering

Informacioni sistem je sistem u kome se veze između objekata u BP, kao i veze sistema sa okolinom ostvaruju razmenom informacija. Sistem deluje na okolinu preko svojih izlaza, a okolina na sistem preko ulaza. Informacioni sistem predstavlja model realnog sistema. Model realnog sistema treba da obuhvati objekte sistema sa njihovim vezama i atributima, kao i procesima koje objekti sistema realizuju.

Glavni problem modelirana informacionog sistema (IS) je savladavanje njegove složenosti. Jedan od načina prevazilaženja problema je apstrakcija. Apstrakcija je kontrolisano uključivanje detalja. Postoje različiti tipovi apstrakcija. Modeliranje realnog sistema obuhvata modeliranje podataka i modeliranje procesa. Proces predstavlja transformaciju bar jednog ulaznog toka u bar jedan izlazni tok.

Drugi način za savladavanje složenosti realnog sistema koji se modelira je dekompozicija. Dekompozicija predstavlja razčlanjavanje složenog sistema na određeni broj manje složenih podsistema.

Zadatak sofverskog inženjeringa je da obezbedi postupak izrade sofverskih proizvoda koji će obezbediti odgovarajući nivo kvaliteta. U razvoju softverskog proizvoda postoje sledeće faze:
- Planiranje,
- Analiza specifikacije zahteva,
- Projektovanje
- Implementacija i
- Raspoređivanje.

Planiranje podrazumeva utvrđivanje ciljeva IS.
Analiza specifikacije zahteva obuhvata skup zahteva koje sistem treba da zadovolji.
Projektovanje podrazumeva modeliranje podataka i procesa.
Implementacija podrezumeva programiranje.
Raspoređivanje podrazumeva raspoređivanje softverskih komponenti na računarskoj mreži.

Ove faze mogu da se realizuju na različite načine: klasični životni ciklus, prototipski razvoj ili operacioni razvoj.

Softver

Softver su (1) Instrukcije (kompjuterski programi), koji kada se izvrse obezbedjuju zeljene funkcije i performance, (2) structure podataka koje omogucavaju programima da na odgovarajuci način rukuju informacijama, (3) dokumente koji opisuju operacije i korisćenje programa.

Softverske aplikacije

Sistemski softver Je kolekcija programa napisana da servisira druge programe. Uglavnom usmereni na interakciju sa hardverom (operativni sistemi, komunikacioni softver, compajleri, editori, baze podataka).

Softver u realnom vremenu je softver koji monitors/analyzes/controls real-world events as they occur is called *real time.*

Poslovni softver sadrzi sirok spektar aplikacija u razlicitim oblastima poslovanja. Pored toga ukljucuje sisteme za upravljanje (menadzment informacioni sistemi) i interaktivne sisteme.

Inzenjerski i naucni softver. Na primer, softver u meteorologiji, astronomiji, biologiji (bio informatika), inzenjerskoj konstrukciji (CASE alati).

Ugradjeni softver. Primenjen u tz. "inteligentnim" proizvodima. Na primer, u automobilima, proizvodnim masinama i td).

Softver licnih racunara. Izuzetno velik napredak poslednjih godina. Na primer, grafika,multimedija, mreze, baze podataka i td.

Softver baziran na WEB-u. Web stranice preko pretrazivaca incorporiraju instrukcije softvera širokog opsega i podatke od hiperteksta do audio i video formata.

Upravljanje projektima

Upravljanje softverskim projektima je umetnost balansiranja suprotstavljenih ciljeva, upravljanja rizikom i prevazilaženja ograničenja radi uspešne isporuke proizvoda koji zadovoljava potrebe kupaca (onih koji plaćaju) i krajnjih korisnika.

- Svrha upravljanja projektima je da:
  - pruži okvir za upravljanje softverskim projektima;
  - pruži praktična uputstva za planiranje, izbor osoblja, izvršenje i monitoring projekata;
  - pruži okvir za upravljanje rizikom.

Analize pokazuju da oko dve trećine projekata prekorači budžet i rokove za više od 125%.

Razlozi neuspeha softverskih proizvoda

- Mada postoji mnogo razloga, jedan od najvažnijih je neodgovarajuće upravljanje projektom.
- Na primer, glavni razlozi za projekte koji su van kontrole su nejasni ciljevi, loše planiranje, nove tehnologije, nedostatak metodologije za upravljanje projektima i nedovoljno ljudstvo.

Projekat

- Projekat je obično jednokratna aktivnost sa dobro definisanim skupom želja i rezultata. Projektne aktivnosti se razlikuju od funkcionalnih aktivnosti.
- Projekat se definiše kao jedinstveni vremenski poduhvat da se proizvede softverski proizvod prema specificiranim kriterijumima performansi kupca.
- Svaki projekat je jedinstven. Svaki projekat je karakterisan nekim stepenom prilagodljivosti kupcu.
- Interesne grupe (klijent ili kupac, matična organizacija, projektni tim i javnost) definišu uspeh ili neuspeh na različite načine.

- Interesne grupe su obično različite za svaki projekat.

Upravljanje projektom

- Upravljanje projektom je poseban pristup za upravljanje poslom.
- Tradicionalno upravljanje podrazumeva planiranje, organizaciju, vođene i kontrolu poslovnih procesa.
- Upravljanje projektom pored toga uključuje i uvođenje i zaključivanje projekta.

WBS

- *Work Breakdown Structure* (WBS) je struktuiran način dekompozicije projekta na osnovne komponente – softver, hardver, komunikaciona mreža, servisi dokumentacija, rad, testiranje, implementacija, instalacija i održavanje. WBS definiše domen projekta i opisuje neophodne aktivnosti za ostvarivanje projektnih ciljeva.
- WBS je formalizovan način svođenja projekta na niže nivoe sa više detalja.

Mrežna analiza

- Project evaluation review technique (PERT)
- Precedence diagram method (PDM).

Software Development Life Cycle – SDLC i Project Life Cycle – PLC se prepliću i treba da se koriste zajedno za uspešan razvoj projekta.
Faze PLC-a:

- Konceptualna faza
- Faza planiranja
- Faza realizacije
- Faza završetka
- Održavanje

Važne stavke: Zahtevi korisnika i upravljanje rizikom.
Projekat zahteva kontinuiranu proveru. Provera se vrši preko odgovarajućih kriterijuma.
Nosioci projekta su rukovodilac projekta (Project Manager) i projektni tim.

Kvalitet softvera

Validacija softverskog proizvoda se sprovodi testiranjem. Testiranjem softvera se otkrivaju greške i procenjuje njegov kvalitet. Softverski proizvod treba da zadovolji definisani i prihvaćeni skup zahteva, prema proceni sa definisanim i prihvaćenim merama i kriterijumima.
Testiranje softvera obuhvata:

- Verifikovanje interakcija komponenata
- Verifikuje odgovarajuću integraciju komponenata
- Verifikuje da su svi zahtevi korektno implementirani
- Utvrđuje i garantuje da su sve otkrivene greške locirane pre nego što je softver razvijen.

Testiranje se sprovodi od početka do kraja razvojnog procesa i u svim fazama životnog ciklusa softverskog proizvoda. Na taj način se obezbeđuje što ranije otkrivanje grešaka. Testiranje je iterativni proces i obezbeđuje povratnu spregu na početak faze u kojoj je greška otkrivena.
Za procenu kvaliteta softvera su potrebni različiti načini testiranja svaki sa različitim fokusom. Dimenzije ovakvog testiranja su: kvalitet, nivoi testiranja i tip testa.

Kvalitet
Kvalitet se procenjuje preko sledećih kriterijuma:

- Pouzdanost – softver je otporan na greške prilikom izvršavanja
- Funkcionalnost – softver izvršava zahtevane slučajeve korišćenja ili željeno ponašanje
- Performanse – izvršavanje i reagovanje su blagovremeni i u skladu sa operacionim karakteristikama realnog sistema. Performanse treba da obezbede funkcionalnost.

Svaka od dimenzija kvaliteta, jednog ili više tipova treba da se izvršava za jedan ili više nivoa. Postoje i druge, subjektivnije procene: sposobnost održavanja, proširivost, prenosivost i fleksibilnost.

Nivoi testiranja

Testiranje nije pojedinačna aktivnost koja se izvodi samo jednom. Testiranje se sprovodi u različitim fazama, od testiranja malih elemenata sistema kao što su komponente (jedinično testiranja) do testiranja kompletnog sistema (sistemsko testiranje).
Postoje različiti nivoi testiranja kao što su: jedinični test, integracioni test, test sistema, test prihvatljivosti.

Vrste testiranja

Postoji više tipova testiranja koji su usmereni na različite ciljeve testa i testiranje samo jedne karakteristike ili atributa softvera. Ona mogu da se odnose na različite nivoe testiranja u različitim fazama životnog ciklusa.
Najčešći tipovi testiranja su: *Benchmark* test, *Konfiguracioni test, Test funkcije, Instalacioni test, Test performansi.*

Inteligentni poslovni sistemi (Business Intelligence)

Data Warehouse, Data Mart

Principi DW:

- Integrisanost
- Orijentacija prema temama
- Zavisnost od jedinice vremena
- Relativna neopromenljivost

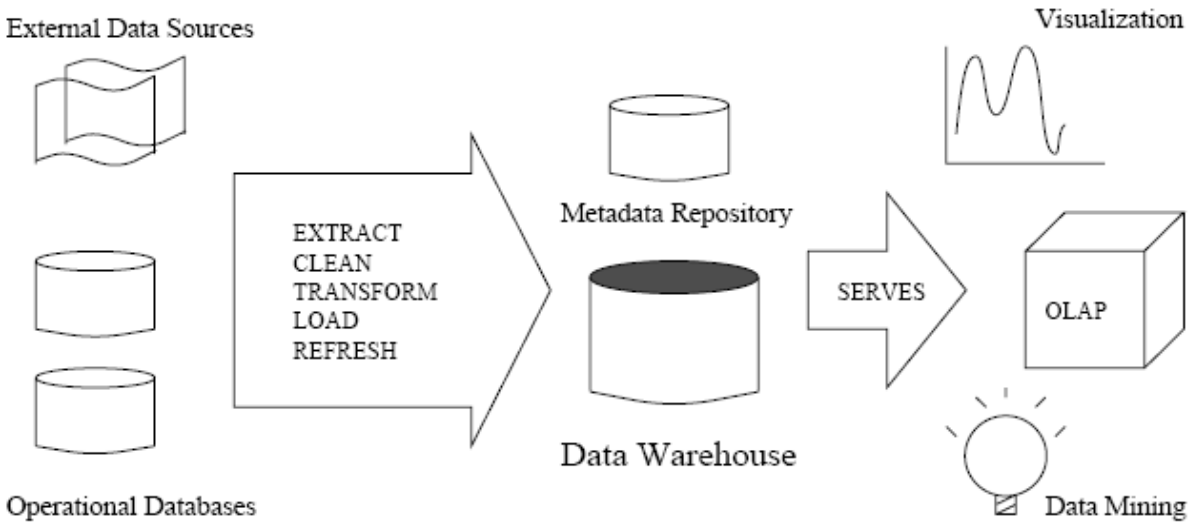
Izgradnja DW:

- relacione BP
- specijalne višedimenzionalne BP

Data Mart je DW koji zadovoljava potrebe jedne organizacione celine.

Data Mining

Cilj analize pri donošenju poslovnih odluka je da se otkriju nepoznate karakteristike podataka, nepoznate veze, zavisnoti ili tendencije.



Primena DW i DM je u oblasti poslovnih sistema, zdravstvu, biologiji, obrazovanju.

Inteligentni sistemi

Istraživanja u oblasti veštačke inteligencije (Artificial Intelligence - AI) su usmerena na izgradnju "mašina koje misle" i povećanje našeg razumevanja inteligencije. Napori su usmereni na dostizanje ljudskih mentalnih sposobnosti uključujući rezonovanje, razumevanje, maštu, prepoznavanje, kreativnost i emocije.
Alati u oblasti AI:

- Sistemi zasnovani na bazi znanja (Knowledge- based sistems)
- Računarska inteligencija
- Hibridni sistemi

Sistemi zasnovani na bazi znanja uključuju ekspertne sisteme i sisteme bazirane na pravilima, objekto orijentisane, frame-based sistemei i inteligentne agente.
Računarska inteligencija uključuje: neuronske mreže, genetičke algotitme i druge optimizacione algoritme.
Tehnike za upravljanje neizvesnošću kao što je fazi logika spadaju u obe kategorije.
Glavne komponente sistema baziranih na znanju su baza znanja, mehanizam za zaključivanje, modul za sticanje znanja, modul za objašnjenje i intefejs prema spoljašnjem svetu.
Ekspertni sistemi su tip sistema baziranih na znanju kreirani da ostvare ekspertizu u pojedinačnom specijalizovanom domenu.