Univerzitetna založba
Univerze v Mariboru

# Računalništvo in informatika v logistiki

Roman
**GUMZEJ**

Univerza v Mariboru

Fakulteta za logistiko

# Računalništvo in informatika v logistiki

Avtor

**Roman Gumzej**

**Januar 2024**

**Naslov** **Računalništvo in informatika v logistiki**
*Title* *Computing and Informatics in Logistics*

**Avtor** Roman Gumzej
*Author* (Univerza v Mariboru, Fakulteta za logistiko)

**Recenzija** Maja Fošner
*Review* (Univerza v Mariboru, Fakulteta za logistiko)

Bojan Rosi
(Univerza v Mariboru, Fakulteta za logistiko)

Simona Sternad Zabukovšek
(Univerza v Mariboru, Ekonomsko-poslovna fakulteta)

Ajda Fošner
(Univerza na Primorskem, Fakulteta za management)

**Tehnična urednika** Roman Gumzej
*Technical editors* (Univerza v Mariboru, Fakulteta za logistiko)

Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

**Oblikovanje ovitka** Jan Perša
*Cover designer* (Univerza v Mariboru, Univerzitetna založba)

**Grafične priloge** Viri so lastni, razen če ni navedeno drugače.
*Graphic material* Roman Gumzej, 2024

**Grafika na ovitku** Low-angle photography of metal structure, foto: Alina Grubnyak, unsplash.com, CC0, 2018; Satelite, foto: Alexas_Fotos, pixabay.com, CC0, 2024; Person-using-black-laptop-computer, foto: Helena Lopes, unsplash.com, CC0, 2020; Board, foto: blicpixel, pixabay.com, CC0, 2024, A close up of a computer motherboard with wires, foto: Florian Krumm, unsplash.com, CC0, 2019; Antenna tower, foto: Farbsynthesel, pixabay.com, CC0, 2024,
*Cover graphics*

**Založnik** **Univerza v Mariboru**
*Published by* **Univerzitetna založba**
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

**Izdajatelj** **Univerza v Mariboru**
*Issued by* **Fakulteta za logistiko**
Mariborska cesta 7, 3000 Celje, Slovenija
https://fl.um.si, info.fl@um.si

**Izdaja** Prva izdaja
*Edition*

**Vrsta publikacije** E-knjiga
*Publication type*

**Dostopno na** http://press.um.si/index.php/ump/catalog/book/829
*Available at*

**Izdano**
*Published at* Maribor, januar 2024

**Prejšnje izdaje**
*Edition history* Elektronski vir. Celje: Fakulteta za logistiko, 2013



CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004(0.034.2)

GUMZEJ, Roman
Računalništvo in informatika v logistiki [Elektronski vir] / avtor Roman Gumzej. - 1. izd. - E-publikacija. - Maribor : Univerza v Mariboru, Univerzitetna založba, 2024

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/829
ISBN 978-961-286-816-1
doi: 10.18690/um.fl.1.2024
COBISS.SI-ID 179481091

**ISBN** 978-961-286-816-1 (pdf)

**DOI** https://doi.org/10.18690/um.fl.1.2024

**Cena**
*Price* Brezplačni izvod

**Odgovorna oseba založnika**
*For publisher* prof. dr. Zdravko Kačič,
rektor Univerze v Mariboru

**Citiranje**
*Attribution* Gumzej, R.(2024). *Računalništvo in informatika v logistiki*. Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.fl.1.2024

***ZAHVALA***

*gre strokovnim recenzentom za korektno opravljeno delo in koristne napotke.*
*Prisrčna hvala mojim bližnjim za njihovo vzpodbudo pri pisanju.*

*Roman Gumzej*

# Kazalo

# KRATICE

ALU – Arithmetic/Logic Unit

AOA – Activity on Arc

AON – Activity on Node

APN – Access Point Name

ARPA – Advanced Research Projects Agency

ASP – Application Service Providers

BA – Business Analytics

BD – Blue-ray Disc

BI – Business Intelligence

BIOS – Basic Input Output System

CAD – Computer Aided Design

CAM – Computer Aided Manufacturing

CCPM – Critical Chain Project Management

CIM – Computer Integrated Manufacturing

CPM – Critical Path Method

CRM – Customer Relationship Management

CASE – Computer Aided Software Engineering

CBIS – Computer Based Information System

CCD – Charged coupled device

CD – Compact Disc

CDD – Compact Disc Drive

CISC – Complex Instruction Set Computing

CMOS – Complementary Metal Oxide Semiconductor

COTS – Commercial Off-The Shelf

CPU – Central Processing Unit

DM – Data Mining

DMA – Direct Memory Access

DNS – Domain Name System

DOS – Disk Operating System

DSS – Decision Support System

DVD – Digital Video (Versatile) Disc

DW – Dataware

EDI – Electronic Data Interchange

EIS – Executive Information System

ERP – Enterprise Resource Planing

ESS – Enterprise Storage System

FDD – Floppy Disc Drive

FTP – File Transfer Protocol

HD – High Definition/Density

HDD – Hard Disc Drive

HTML – Hyper Text Markup Language

HW – Hardware

GIGO – Garbage In Garbage Out

GPRS – General Packet Radio Service

GSM – Global System for Mobile communications

IM – Instant Messaging

IMAP – Internet Message Access Protocol

ISP – Internet Service Provider

IP – Internet Protocol

ISP – Internet Service Provider

IT – Information Technology

ICT – Information Communication Technology

KMS – Knowledge Management System

LIS – Logistics Information System

LAN – Local Area Networks

LRC – Linear Responsibility Chart

LW – Liveware

MIME – Multipurpose Internet Mail Extension

MIS – Management IS

MMS – Multimedia Messaging System

MRP – Master Resource Planning

MPS – Master Production Scheduling

NAS – Network-Attached Storage

NSP – Network Service Provider

NW – Netware

OW – Orgware

OS – Operating System

P2P – Peer-to-Peer

PC – Program Counter

PERT – Project Evaluation and Review Technique

PIM – Personal Information Manager

POP3 – Post Office Protocol v.3

RFID – Radio Frequency IDentification

RAID – Redudant Arrays of Independent Disks

RAM – Random Access Memory

RAT – Remote Access Tools

ROM – Read-Only Memory

SAN – Storage Area Network

SCM – Supply Chain Management

SIGEN-CA – Slovenian General Certification Authority

SIGOV-CA – Slovenian Governmental Certification Authority

SOP – Sales and Operations Planning

SOAP – Simple Object Access Protocol

SMS – Short Message Service

SMTP – Simple Mail Transfer Protocol

SQL – Structured Query Language

SW – Software

TCP – Transmission Control Protocol

TPS – Transaction Processing System

UDP – User Datagram Protocol

UMTS – Universal Mobile Telecommunication System

URI – Uniform Resource Identifier

URN – Uniform Resource Name

URL – Uniform Resource Locator

USB – Universal Serial Bus

VOD – Video On Demand

XML – Extensible Markup Language

WAN – Wide Area Networks

WAP – Wireless Application Protocol

WBS – Work Breakdown Structure

WLAN – Wireless LAN

WWAN – Wireless WAN

WWW – World Wide Web

Učbeniku na pot...

Na Fakulteti za logistiko Univerze v Mariboru že od začetka uvajamo uspešno in učinkovito prakso podajanja znanja novodobnim generacijam študentov s pomočjo sodobnih metod učenja in poučevanja v celovitem sistemu e-študija. Hkrati se zavedamo, da procesa učenja ni mogoče zožiti zgolj na uporabo interneta in elektronsko posredovane informacije. Zato smo se po vzoru dobrih že preverjenih praks hibridnega študija odločili, da bomo pri posredovanju znanja uporabili tako elektronske kot tudi tradicionalne metode, kar omogoča bolj na študenta osredotočen pristop k študiju. S kombinacijo interaktivnih in statičnih virov študentom omogočamo tudi časovno in prostorsko prilagodljivost. Skladno s to usmeritvijo je nastal tudi ta učbenik, ki ga ob tej priliki priporočam zlasti študentom logistike, pa tudi vsem drugim, ki bi se želeli bolje seznaniti z zanimivim področjem informatike v logistiki.

V Celju, 16. junija 2022 dr. Maja Fošner, dekanica

**Računalništvo in informatika v logistiki**

Učbenik je namenjen predvsem študentom prvih letnikov visokošolskega strokovnega in univerzitetnega študijskega programa Fakultete za logistiko. Je osnovna literatura učnih predmetov Osnove računalništva v logistiki na visokošolskem strokovnem in Računalništvo v logistiki na univerzitetnem študijskem programu. V okviru teh predmetov zasnujemo logistične informacijske sisteme. Predstavimo njihov pomen, vlogo in naloge ter komponente: računalniška podatkovna, strojna, mrežna, programska in organizacijska oprema. Pri praktičnem delu se posvetimo predvsem računalniško podprtemu inženiringu logističnih informacijskih sistemov na univerzitetnem ter poslovni analitiki in podpori odločanju na visokošolskem strokovnem programu. Vsebino sklenemo z metodami in mehanizmi za zagotavljanje varnosti računalniško podprtih logističnih informacijskih sistemov. Učbenik predstavlja osnovo za naknadno obravnavo specializiranih vsebin pri predmetih Poslovni informacijski sistemi v logistiki, Informacijska podpora logističnim sistemom in procesom ter Integracije logističnih informacijskih sistemov.

V Celju, 30. junija 2022 dr. Roman Gumzej

# UVOD

Namen učbenika je študentom prvega letnika dodiplomskega študija logistike podati solidno osnovo s področja informacijske podpore logističnim sistemom in procesom, ki jo bodo lahko uporabili tudi pri drugih študijskih predmetih ter nadgradili pri preostalih predmetih iz te predmetne skupine. V ta namen najprej teoretično in praktično osvežimo in utrdimo osnove računalništva in informatike ter podamo osnovne zakone, ki so jim podvržene informacijske tehnologije. Osredotočimo se na tiste, ki omogočajo uporabo in prenos tehnologij med informatiko in logistiko.

Kot transakcijsko intenzivna panoga lahko logistika s pridom izkoristi zmogljivosti računalniške obdelave. Zato sta računalništvo in informatika zelo prepleteni z logistiko in je tako lahko prišlo do številnih in raznovrstnih prenosov tehnologij – od konceptualnih (npr. projektno vodenje, teorija množične strežbe, optimizacijske metode) do tehničnih (npr. logistični informacijski sistemi in platforme, tehnologije navigacije in identifikacije, ipd.). Vzporedno z obravnavo računalniških vsebin bomo zato s primeri predstavili tudi aplikacije računalništva in informatike v logistiki, kar je tudi namen istoimenskega študijskega predmeta.

Iz definicije informacijskih sistemov izhajamo, ko opisujemo njihove računalniške komponente: strojna oprema (angl. hardware), programska oprema (angl. software), omrežna oprema (angl. netware) in podatkovna oprema (angl. dataware). V učbeniku navedeni performančni kazalci se nanašajo na aktualno stanje informacijske tehnologije (IT) in navedene industrijske standarde, vendar jih je na splošno treba obravnavati kot spremenljive – v smislu stalnih izboljšav performans, večanja kapacitet, itd. Osrednja komponenta vsakega informacijskega sistema (IS) so ljudje (angl. liveware). Njihovo upravljanje je tematika drugih učnih predmetov in jo zato tukaj le omenimo, ko v okviru organizacijske komponente IS (angl. orgware) obravnavamo računalniško podprto planiranje operacij in aktivnosti.

Pred predstavitvijo informacijske infrastrukture IS, ki jo popularno s kratico večkrat imenujemo kar IT, bomo pregledno predstavili strukturo logističnih IS (LIS). LIS služijo različnim nivojem in profilom sodelavcev v organizaciji in jih funkcijsko povezujejo. Po eni strani tu gre za vertikalno povezovanje vodstvenih odločitev z operativnimi aktivnostmi, po drugi pa za horizontalno povezovanje med sodelavci, oddelki, organizacijami in njihovimi strankami. Pri tem opravljajo logistično funkcijo, saj neposredno podpirajo odvijanje poslovnih procesov, omogočajo pa tudi informirane odločitve pri ključnih aktivnostih oz. operacijah. Osrednjo povezovalno vlogo imajo t.i. celoviti upravljavski IS (ERP), ki jih bomo obravnavali v višjih letnikih.

V okviru praktičnega dela študijskega predmeta se bomo osredotočili na operativni nivo LIS in transakcijske IS (TPS) kot vir transakcijskih podatkov, ki jih v LIS zbiramo, hranimo, obdelujemo in posredujemo. TPS so najbližje informacijski infrastrukturi LIS, omogočajo pa nam tudi uporabo IS za upravljanje znanja (KMS) na osnovi poslovne inteligence (BI). Medtem, ko tu obravnavamo transakcijske podatke iz vidika poslovne analitike (BA) in podpore odločanju (DSS), se bomo v višjih letnikih posvetili inženiringu na osnovi znanja (KBE), kot nadgradnji omenjenih metod poslovne inteligence.

Obravnavo računalniško podprtih LIS sklenemo s temo, ki zadeva dve vedno pomembnejši področji uporabe IT – varnost IS in zaupnost njihovih podatkov. Zaradi hitrega razvoja e-poslovanja sta postali nepogrešljiv del vsakega računalniško podprtega poslovnega IS. Ker običajno predstavljata nadgradnjo osnovne funkcionalnosti IS, zadevata pa vse komponente IS, imata tudi tukaj posebno mesto.

# 1 Utemeljitev računalništva in informatike v logistiki

## 1.1 Raziskovanje in obvladovanje realnega sveta

Od nastanka civilizacije človek skuša spoznavati svet okoli sebe s tem, da tvori zapise o svojih vtisih – vsem kar vidi in doživlja. Ločimo več pristopov k opazovanju realnega sveta:

1. opazovanje, ki je primerno predvsem za deterministične fenomene; gre za dogodke, ki se zgodijo vedno na enak način in v določenem časovnem sosledju ali kot reakcija na iste dogodke
2. analiza je primerna predvsem za stohastične fenomene; na podlagi preteklih izkušenj induktivno sklepamo, kaj se bo zgodilo – pri tem nastopa določena verjetnost, da se bo to dejansko zgodilo
3. sinteza uvaja sinergični učinek obeh predhodnih pristopov; z dedukcijo iz zaznanih splošnejših zakonitosti lahko predvidevamo obnašanje v sorodnih situacijah v bodoče

*Pristop opazovanja* in opisovanja uporabljamo pri fenomenih, ki v nespremenjenih okoliščinah dajejo vedno enake rezultate. Tipični primeri takšnih fenomenov so: prosti padec, izmenjava dneva in noči, gibanje teles pod vplivom stalne sile (če zanemarimo trenje), uparjanje tekočin,...

Ker so fenomeni, ki jih lahko opišemo zgolj na osnovi opazovanja, bolj izjema kot pravilo, se je že v začetku 16. stol. pojavil analitični pristop. Primeri takšnih fenomenov so: oblaki lahko prinesejo dež, sneg ali točo, vožnja z avtomobilom po isti poti – pri tem lahko dosežemo zelo različne čase potovanja (glede na razmere na cesti),...

*Analitični pristop* je bil prevladujoč od 16. pa vse do začetka 20. stoletja. Je strogo formaliziran in zajema naslednje korake:

1. definicija problema
2. razčlenitev problema na fizične in logične dele
3. razčlenitev vseh vzročno-posledičnih zvez med deli (kavzalni odnos)
4. ustvarjanje verige kavzalnih odnosov
5. rešitev problema (pojasnitev fenomena)

Pod vplivom Alberta Einsteina (1879–1955) v ospredje pride *sistemski pristop*. Sistemski – deduktivni – pristop nas uči, da najbolj zanesljivo in celovito spoznavamo realni svet s sinergičnim učinkom, ko svet okoli sebe opazujemo kot sistem, ki deluje po določenih "naravnih" zakonitostih. Einstein se je v okviru svojih raziskav ukvarjal s fenomeni, ki jih z analitičnim pristopom ni bilo mogoče rešiti in jim pripisal sinergični učinek. Le-ta se pojavlja, ko pomena nekega fenomena ne moremo razložiti zgolj na podlagi lastnosti, niti ne moremo nanj sklepati na podlagi kavzalnih odnosov. Primer sinergičnega učinka je živi organizem – ima funkcijo preživetja, medtem ko posamezne celice, iz katerih je sestavljen, te funkcije nimajo. Ker metoda indukcije v tem primeru ne da ustreznih rezultatov, uporabimo metodo dedukcije, ki pojasnjuje njihov sinergični učinek. Deduktivno sklepanje deluje v smeri od splošnega k posebnemu. Lastnosti celote so rezultat interakcije med vsemi ali njenimi izbranimi (ne vedno istimi) deli.

*Sistemski pristop* temelji na naslednjih tezah:

1. vse je sistem
2. vsak sistem sestavlja en ali več podsistemov
3. sistem pogosto ima lastnosti, ki jih nima nobeden od njegovih delov
4. vsak sistem ima: elemente (E), strukturo (R) in smoter/funkcijo (F)

***Sistemska teorija*** *je pristop k reševanju problemov, ki predpostavlja, da je vsak problem možno obravnavati kot sistem.*

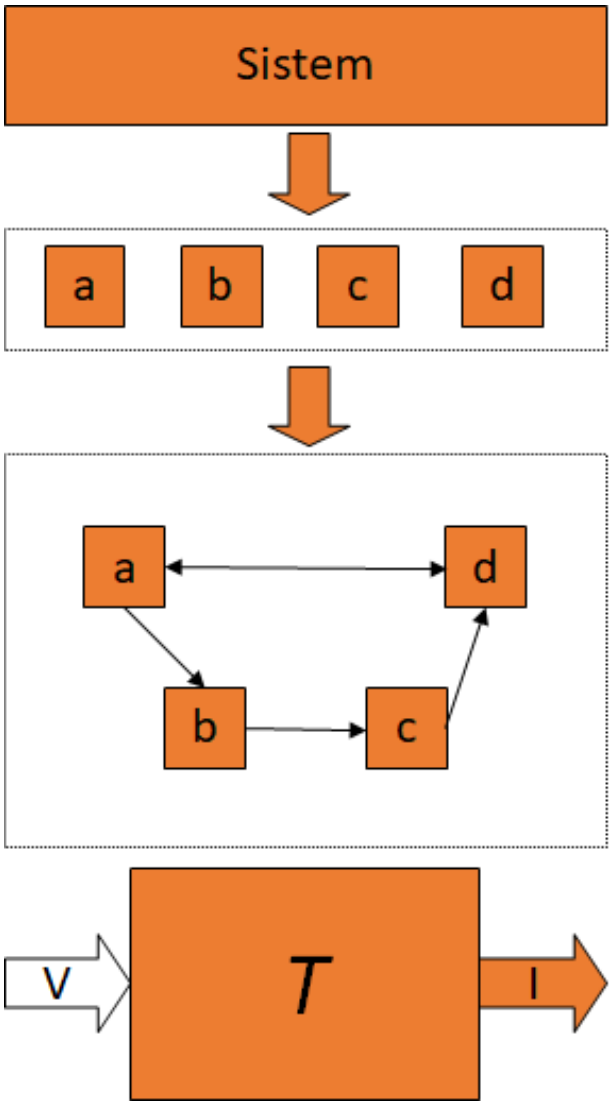
*Sistemska teorija* je sestavljena veda in zajema: splošno teorijo sistemov, kibernetiko (veda o obnašanju tehničnih in družbenih sistemov), semiotiko (veda o obliki simbolov in sistemih simbolov), semantiko (veda o pomenu simbolov), informatiko, matematično teorijo sistemov, itd. Od tod izhaja definicija sistema:

***Sistem*** *je strukturni sklop povezanih elementov, ki z medsebojnim delovanjem in sodelovanjem z okolico izpolnjujejo nek namen ali funkcijo.*

*Sistemski pristop* k reševanju problemov zajema določanje:

1. komponent sistema (elemente, podsisteme)
2. mej sistema (vse izven teh mej je okolica sistema)
3. strukture sistema (ta predstavlja način povezanosti komponent)
4. okolice sistema (tudi to na nek način sistematično opišemo)
5. povezav sistema z okolico (vhodi in izhodi sistema)
6. cilja sistema (smoter sistema)
7. funkcij sistema (način, na katerega sistem transformira vhode v izhode)
8. procesov v sistemu (način in vrstni red uporabe funkcij sistema)
9. zakonitosti, ki jim je sistem podvržen

Poenostavljeno strukturo sistema nam podaja Slika 1, ki predstavlja sistem kot sestav povezanih členov, ki vsak zase in skupaj predstavljajo transformacijo nekih vhodov (V) v izhode (I) na podlagi transformacijske funkcije ($T$), ki predstavlja namen oz. smoter sistema. V primeru informacijskega sistema gre za transformacijo vhodnih podatkov v rezultate – nove podatke. Ker predstavljajo novo znanje, govorimo o informacijah, ampak o tem več kasneje.

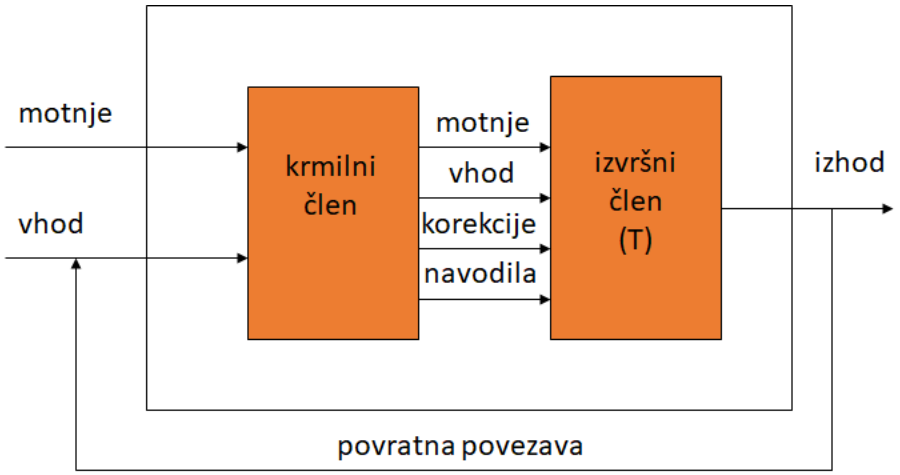



Slika 1: *Funkcionalna shema sistema*
Vir: lasten

Sistemski pristop se je uveljavil v tridesetih letih prejšnjega stoletja in temelji na sistemski teoriji. Sistemska teorija je bila temelj znanstvenega pogleda na svet, vesolje in stvarstvo, ki se je začel širiti v času francoske revolucije in so ga postavili nasproti takrat uveljavljenemu teološkemu vidiku. Ta favorizira tehnicistični pristop k spoznavanju sveta, po katerem je vse določljivo in doumljivo iz znanstvenega vidika. Razen že omenjenega A. Einsteina so bili njegovi prvi protagonisti: Karl Marx (1817–1883), Herbert Spencer (1820–1903), Vilfredo Pareto (1848–1923), Émile Durkheim (1858–1917), Alexander Bogdanov (1873–1928), Nicolai Hartmann (1882–1950), Stafford Beer (1926–2002), Robert Maynard Hutchins (1929–1951) in drugi (vir: Wikipedia).

### 1.1.1 Kibernetika in informatika

Procesi predstavljajo način spreminjanja stanj posameznih komponent in sistema v celoti. Med odvijanjem procesov sistem izpolnjuje svoj smoter. Z usmerjenim delovanjem na spremenljivke sistema le-ta prehaja v želeno stanje – upravljanje sistema. Od tu pojem *krmiljenje.* Norbert Wiener (1894–1964) je 1948. leta utemeljil novo znanstveno disciplino, ki jo je poimenoval *kibernetika* (gr. Kibernetes – krmar). *Kibernetika* se ukvarja z načeli upravljanja dinamičnih sistemov. Wiener splošne koncepte kibernetike razloži v knjigi „Cybernetics or Control and Communication in the Animal and Machines“ – obstajajo splošne zakonitosti upravljanja sistemov ne glede na vrsto: „Vsak sistem, ki ga namensko ne usmerjamo, izgublja svojo naravno urejenost in možnost izpolnjevanja svojega smotra – uresničevanja cilja“. Pravimo, da v njem narašča neurejenost sistema – entropija ali nered. Stopnja entropije je odvisna od verjetnostnih porazdelitev možnih stanj sistema. Upravljanje sistemov temelji na aktivnostih nadzora (kontrole) in izmenjavi informacij (komunikaciji) s sistemom. Wiener je svoje raziskave povzel v *krmilnem sistemu* kot nadgradnji osnovnega funkcionalnega modela sistema, ki ga ponazarja *regulacijska zanka* (Slika 2).




Slika 2: *Regulacijska zanka*
Vir: lasten

Iz kibernetike kot dela sistemske teorije izvirajo naslednje znanstvene discipline, ki so neposredno vplivale na nastanek informatike:

1. teorija komunikacije (izmenjava informacij)
   (a) teorija informacije (nastanek, prenos, sprejem in uporaba informacij)
   (b) teorija kodiranja (načini prikaza informacij in pretvorbe med njimi)
   (c) pomenoslovje (semiotika - definira znake, simbole in signale s katerimi prikazujemo informacije ter način njihovega povezovanja v smiselne celote)
2. teorija odločanja
   (a) racionalno (odločanje ob znanih ali predvidljivih posledicah)
   (b) intuitivno (odločanje ob nezadostnih informacijah o vseh opcijah)
   (c) hevristično (hitrejše odločanje s kombinacijo racionalnega in intuitivnega principa)

3. teorija programiranja
   (a) teorija algoritmov (sistematičen prikaz metodološkega znanja v smislu določenega števila nedvoumnih korakov, ki vodijo do gotove rešitve problema, če le ta obstaja)
   (b) teorija avtomatov (definicija tehniških sistemov, ki lahko opravljajo fizične in logične operacije, ki so bile predhodno algoritmično definirane)
4. teorija povratnih zvez
   (a) tehniških (v tehniških sistemih)
   (b) družbenih (v družbenih skupinah in skupnostih)
5. splošna teorija sistemov (združevanje znanstvenih področij v izogib prekrivanju/podvajanju, neracionalnemu trošenju sredstev in nepotrebnemu kopičenju znanstveno-raziskovalnih rezultatov)

Vse naštete znanstvene discipline so vplivale na nastanek informatike, katere namen je obvladati naraščajočo količino podatkov, ki izvira iz naraščanja kompleksnosti problemov drugih znanstvenih disciplin, in pri tem ločiti bistveno od manj pomembnega. *Informatika* je bila zasnovana z nalogo izvršiti sistematizacijo podatkov z namenom pridobivanja novih informacij.

Izraz "informatika" izvira iz Francoščine:

*L'informatique =L'information+L'automatique*
(informatika=informacija+avtomatika)

Sedaj, ko smo dodobra opisali ozadje in okoliščine nastanka te "mlade" znanstvene discipline (okoli 1950), že lahko navedemo definicijo informatike:

***Informatika*** *je znanstvena disciplina, ki se ukvarja z zajemanjem, hrambo, obdelavo in uporabo informacij.*

Iz definicije informatike izhaja, da gre pri informatiki za avtomatsko obdelavo podatkov (praviloma s pomočjo računalnikov). Od tod tudi povezava z drugo "novo" vedo – računalništvom.

***Računalništvo*** *predstavlja znanstveno disciplino, ki se ukvarja s proučevanjem računalnikov in postopkov v njih.*

Eden od teh postopkov je tudi obdelava podatkov, vendar je pri računalništvu poudarek na podatkih kot predstavitvi informacij in postopkih za njihovo obdelavo, medtem ko je pri informatiki bistvena klasifikacija podatkov, ki predstavljajo informacije. To predstavlja glavno razliko med obema disciplinama. Zaradi večplastne povezanosti obeh znanstvenih disciplin običajno govorimo o enotnem znanstveno-raziskovalnem področju – *računalništvo in informatika* (angl. *computer science*).

Pojem, ki se prav tako pojavlja v zvezi z računalništvom in informatiko in predstavlja rezultat sinergičnega delovanja obeh znanstvenih disciplin, so *informacijske tehnologije (angl. Information Technology, IT)*. Tu gre za sklop vseh tehnologij, ki podpirajo zbiranje, hranjenje, prenos in obdelavo informacij. Če želimo poudariti podporo telekomunikacij pri naštetih operacijah, lahko uporabimo tudi pojem *informacijsko-komunikacijske tehnologije (angl. Information Communication Technology, ICT/IKT),* vendar se oba pojma običajno uporabljata kot sinonima.

### 1.1.2 Kratka zgodovina računalništva

Zgodovino računalništva in informatike bi lahko na kratko povzeli v izjavi: "od preprostih konceptov do zapletenih izračunov." Zgodovinsko gledano je razvoj računalniške obdelave podatkov šel vedno vzporedno z razvojem družbe in bil vedno povezan s količino podatkov, ki jih je bilo potrebno obdelati – bolj kot je bila družba napredna, več je bilo podatkov, ki jih je bilo potrebno obdelati. To je potegnilo za sabo nov razvojni ciklus računskih naprav, ki so bile sposobne

takšne obdelave, kar je spet posredno vplivalo na razvoj družbe in razvoj novih potreb po še hitrejši/obsežnejši obdelavi podatkov. Zato nekateri govorijo tudi o "sklenjenem ciklu potreb po hitri obdelavi podatkov".

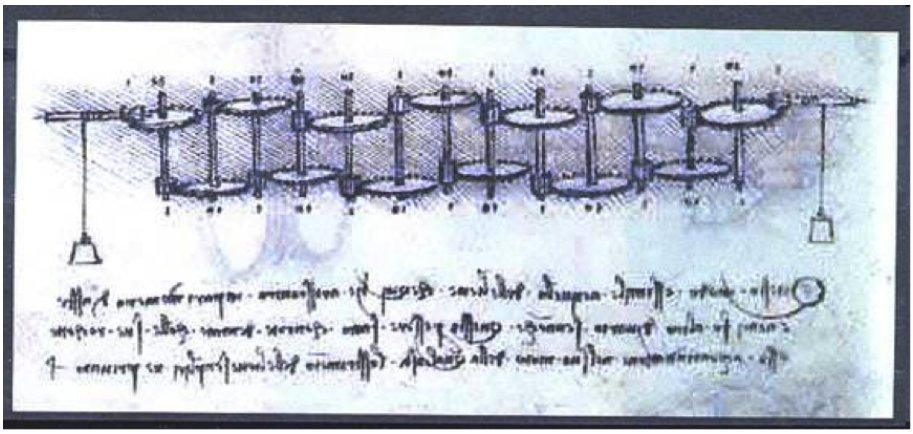
V grobem ločimo naslednje faze v razvoju obdelave podatkov:

1. Ročna obdelava podatkov
2. Mehanska obdelava podatkov
3. Elektro-mehanska obdelava podatkov
4. Elektronska obdelava podatkov

Pri ročni obdelavi podatkov gre za najstarejšo vrsto obdelave podatkov, kjer so ljudje uporabljali le roke in um. To je tudi prva zavestna obdelava podatkov, s katero se sreča vsak posameznik (npr. "računanje na prste", "daljši/krajši meseci", ipd.). Zaradi pomnjenja vmesnih rezultatov, prav tako pa tudi zaradi hitrejših in/ali bolj zapletenih izračunov, so ljudje kmalu začeli izumljati računska pomagala. Prvi kalkulator Abak je bil iznajden 2600 let pr.n.št. in je dobil svoje ime po osnovnih gradnikih – kamenčki (lat. calculus). Kljub svoji preprostosti se ta naprava predvsem na Kitajskem, kjer so jo iznašli, uporablja še danes. Kot zanimivost lahko navedemo, da so bili baje vsi izračuni za prvo kitajsko jedrsko bombo opravljeni z njim.

Ljudje so kmalu spoznali, da lahko učinkovito obdelujejo podatke, če se držijo določenih postopkov – *algoritmov*. Beseda algoritem izvira iz latinskega prevoda imena arabskega matematika Mohammeda al-Khowarizmija, ki ga imajo za "očeta" matematične algebre. Algoritmi so bistveno vplivali na razvoj računskih strojev (računalnikov), ki slonijo na algoritmičnem principu in s tem povečujejo hitrost in točnost izračunov.

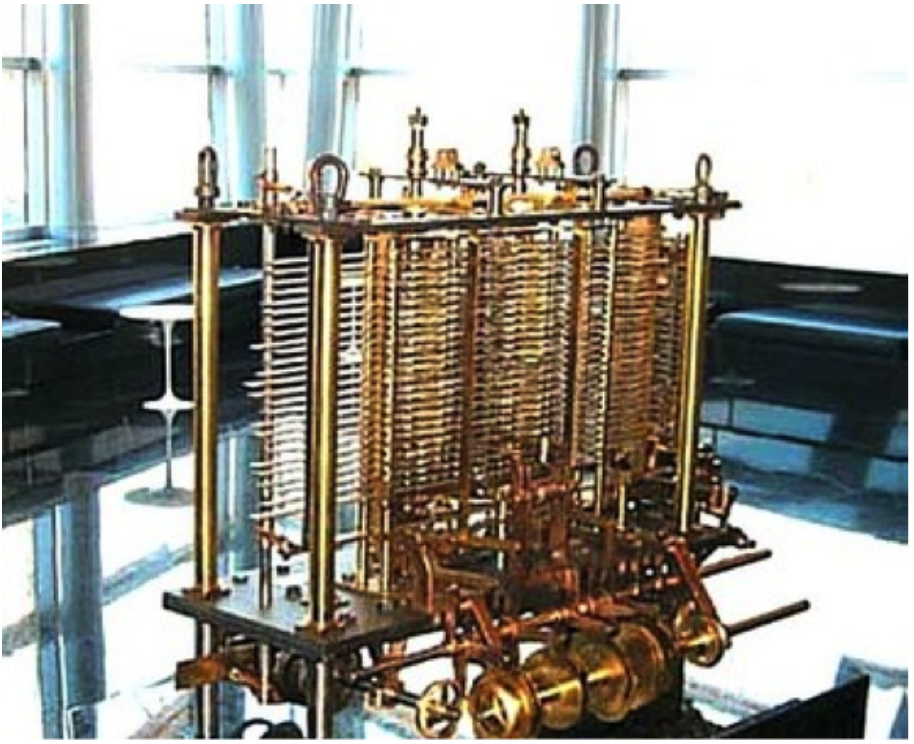
**Algorit*em*** *je natančno definiran seznam dejavnosti, ki jim je potrebno dobesedno slediti, da bi dosegli želeni rezultat – rešili določen problem.*



Slika 3: *Da Vincijev računski stroj*
Vir: Da Vinci, Codex Madrid I, str. 36

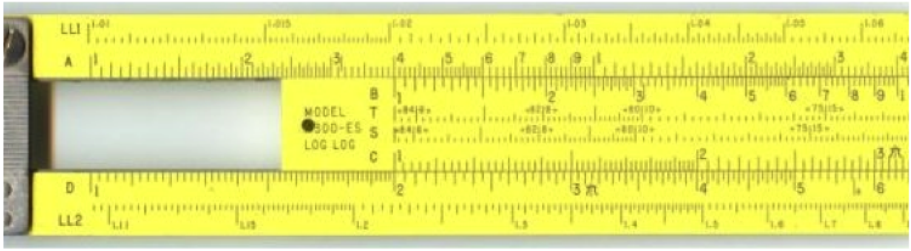
Prvi mehanski računski stroj (Slika 3) je že okoli 1500 n.št. zasnoval univerzalni genij Leonardo da Vinci. Tej ideji so sledile vedno naprednejše neodvisno razvite praktične izvedbe kot sta na primer okoli leta 1642 razviti Pascalov aritmetični stroj, ki je omogočal le seštevanje in odštevanje, in leta 1673 razviti Leibnizov računski stroj, ki je razen seštevanja in odštevanja omogočal tudi množenje in deljenje ter je že uporabljal binarni sistem, ki ga sodobni računalniki uporabljajo še danes. Oba sta bila sestavljena iz ročno krmiljenih mehanskih sklopov – sistem, ki so ga še v prejšnjem stoletju uporabljali v trgovinskih blagajnah za zaračunavanje blaga.

Prvi ne-ročno gnani mehanski računski stroj je bil Babbage-ov diferenčni stroj (1820, Slika 4). Gnal ga je parni stroj in je bil predhodnik prvega univerzalnega računskega stroja – analitičnega stroja (1837), ki bi naj že imel ustroj sodobnih računalnikov in izvajal ukaze, shranjene na luknjanih karticah. Ti zgodnji "stroji za obdelavo števil" so imeli dve pomembni skupni lastnosti: (1) sestavljeni so bili iz mehanskih sklopov in (2) bili so "po meri človeka" – njihovi deli so bili dovolj veliki, da jih je bilo mogoče ročno sestaviti/popraviti.



Slika 4: *Babbageov diferenčni stroj*
Vir: http://www-03.ibm.com/ibm/history/exhibits/attic2/attic2_182.html

V omenjeno skupino spada še ena iznajdba – škotski matematik Johan Napier je leta 1617 napravil logaritemske tablice. Le-te je kopiral na slonokoščene palice, z njihovim premikanjem pa dobil želeni zmnožek. Iz te naprave so kasneje razvili logaritmično računalo. Od odkritja logaritmičnega drsnega računala (“šiberja”, Slika 5) 1621, je to matematično orodje prevladovalo v tehniških okoljih do sedemdesetih let prejšnjega stoletja, ko se je pojavil prvi elektronski kalkulator. Izum logaritmičnega računala je poenostavil in pohitril operacije množenja, deljenja in potenciranja ter olajšal tudi izdelavo elektromehanskih in elektronskih računskih naprav, ki te operacije izvršijo avtomatsko.




Slika 5: *Logaritmično računalo*
Vir: Wikipedia, 2005

Za razvoj sodobnih računalnikov je bil pomemben koncept Babbageovega analitičnega stroja (1837), ki že ima njihov ustroj:

1. enota, kjer vnašamo podatke
2. enota, kjer potekajo izračuni
3. enota, ki diktira računalniku, kaj naj dela
4. enota, ki pomni podatke
5. enota, ki prikazuje rezultate

Bil je razvit pred svojim časom in takrat tehnično ni bil izvedljiv, so pa model ohranili in uporabili kasneje pri izgradnji elektronskih in polprevodniških računalnikov. Še dve iznajdbi sta bistveno vplivali na razvoj računskih naprav, čeprav to ni bil njun prvotni namen. Obe sta se kasneje razvili v enote za vnos podatkov kot tudi enote za prikaz in hranjenje rezultatov. John Mill je leta 1714 predstavil svoj izum – pisalni stroj; leta 1868 so razvili model, s katerim je bilo možno pisati hitreje kot ročno (Christoper Latham Sholes). Joseph Marie Jacquard pa je leta 1801 izumil mehanične statve, ki so tkale vzorce, shranjene na luknjanih karticah kot prvem pomnilniku podatkov.

Mejnik med mehansko in elektromehansko obdelavo podatkov predstavlja iznajdba Hermana Holleritha iz leta 1890 (Slika 6), s katero je zmagal na natečaju za avtomatsko obdelavo podatkov iz popisa prebivalcev, ki ga vlada ZDA od leta 1851 sistematično izvaja vsakih 10 let. Nosilec podatkov je bila Jacquardova luknjana kartica, stroj pa je že gnala elektrika. Za razliko od popisa 1880, ki je trajal 8 let, so na ta način prve neuradne rezultate dobili že po šestih tednih, popis pa zaključili v enem letu. Zaradi velike praktične vrednosti, so se odločili začeti serijsko proizvodnjo Hollerithovega izuma – podviga se je lotilo podjetje *Tabulating Machines Company*, katerega ustanovitelj je bil Hollerith. To je kasneje 1911 združeno s sorodnimi podjetji v *Computing Tabulating Recording Corporation* leta 1924 preraslo v *International Business Machines (IBM ali "Big Blue")*, ki je še danes eden od največjih in najpomembnejših proizvajalcev računalnikov na svetu.

Tehnološki mejnik med elektromehansko in elektronsko obdelavo podatkov predstavlja iznajdba elektronke (vakuumske diode). Več znanstvenikov je raziskovalo to, kar je Thomas Edison 1884 patentiral kot "Edisonov efekt". Elektronko, kot osnovo tega fenomena, je sprva označil kot neuporabno, zato iznajdbo pripisujejo bolj Flemingu, ki je eksperimentiral z Edisonovimi elektronkami in leta 1904 izgotovil napravo, ki jo je sam poimenoval "oscilacijski ventil", ker je bila sposobna usmerjanja izmenične napetosti in detekcije radijskih valov. 1906 je Robert von Lieben patentiral triodo, ki je bila sposobna še ojačitve napetosti in je vsebovala elektromagnet za usmerjanje električnega toka. Nastale so še številne inačice elektronk, vendar osnovnega problema – prekomernega segrevanja med delovanjem – niso uspele v celoti zajeziti (vir: Wikipedia).

Druga svetovna vojna je usodno vplivala na razvoj elektronskih računalnikov – sodobna orodja in orožja so za svojo konstrukcijo zahtevala veliko izračunov. Neodvisno je nastalo več elektronskih računalnikov. Elektronke so sestavljale "srce" računalnika z imenom Colossus (Slika 7), ki so ga Britanci med 2. svetovno vojno uporabljali za dešifriranje nemških z Enigmo kodiranih sporočil.

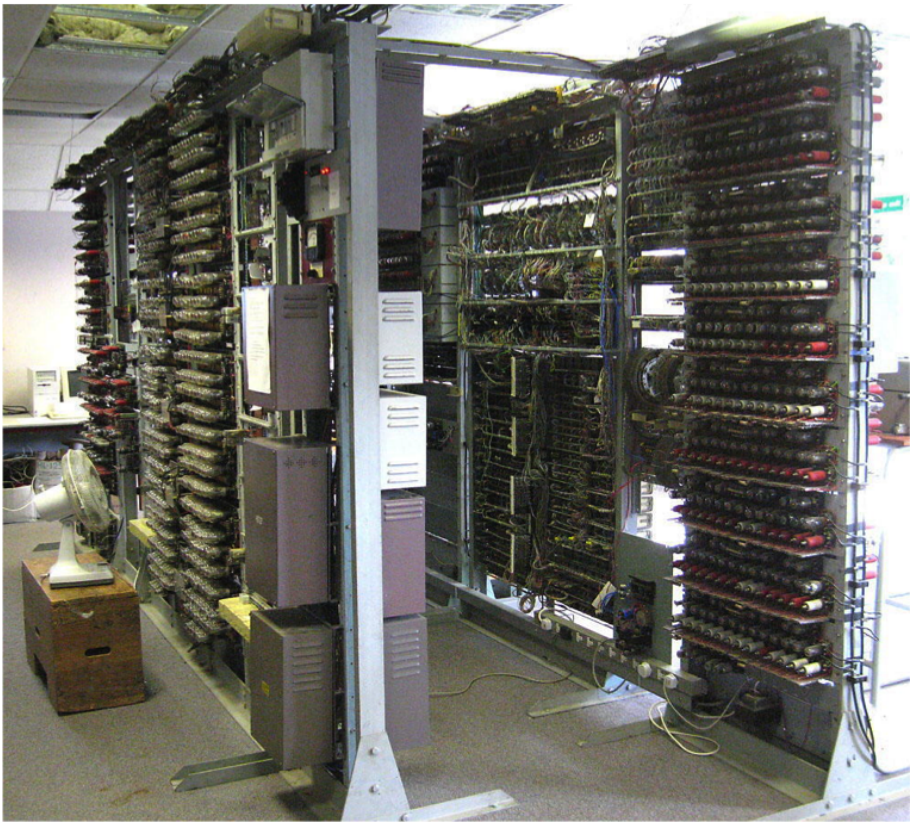
Slika 6: *Hollerithov računski stroj*
Vir: en.wikipedia.org, Adam Schuster, CC-BY, 2015

Tudi Nemci so sestavili splošno-namenski računalnik, ki pa je bil med bombardiranjem uničen. Med leti 1943-45 so Američani izdelali ENIAC, katerega primarna raba je bil izračun trajektorij artilerijskih izstrelkov. Mere teh računalnikov so bile resnično "kolosalne". ENIAC je bil cca. 30m dolg, 2,4m visok in 30T težek, vseboval je 17.468 elektronk, ki so vsako uro potrošile 200 kW električne energije. Zato so za obratovanje običajno potrebovali lastno elektrarno. Skupna slabost elektronskih računalnikov so bile pogoste okvare, ki so bile v glavnem pogojene s kratko življenjsko dobo elektronk.

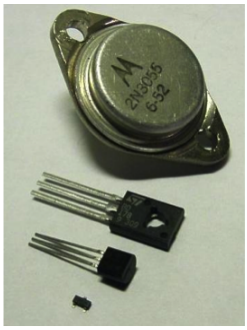
V času elektronskih računalnikov se je pojavil tudi izraz "bug" (računalniški hrošč), ki se je v računalniškem slengu udomačil za karakterizacijo anomalij, ki so vzrok napačnega delovanja ali odpovedi. Leta 1945 je odpoved vojaškega računalnika povzročil molj, ki se je ujel med releji in povzročil kratek stik. Napako je našla in odpravila častnica Grace Hopper ter dokaz (molja) zalepila v dnevnik. Čeprav gre tukaj za prvi dokumentirani primer, pa je potrebno pripomniti, da je bil izraz v rabi že prej tudi pri mehanskih strojih.

Naslednji tehnološki mejnik v razvoju računskih strojev predstavlja iznajdba tranzistorja (1926). Polprevodniški tranzistor, ki pa je bil patentiran šele 1947, so iznašli pri *Bell Labs*. Njegova funkcija je v osnovi enaka funkciji elektronke – usmerjanje električnega toka – vendar je bil "velikosti graha" in mnogo manj "požrešen". S komercialno proizvodnjo tranzistorjev so začeli pri *Texas Instruments* (TI) leta 1954, International Business Machines (IBM) pa jih je vgradil v prvi komercialni računalnik z oznako *650*.

Ko so na polprevodniški element namestili več miniaturnih aktivnih elementov (npr. tranzistorjev, diod) in pasivnih elementov (npr. kondenzatorjev, uporov), je nastalo *integrirano vezje (angl. integrated circuit, IC, chip)*. Čez 4 leta Slika 9) je TI predstavil prvo integrirano vezje, sestavljeno iz 5 elementov na 15cm dolgi germanijevi plošči. Miniaturizacija elementov (Slika 8), njihovo povezovanje v ohišja in izgradnja komponent ter povezovanje le-teh v komple-



Slika 7: *Colossus*
Vir: jbo.wikipedia.org, MaltaGC, CC-BY-SA, 2008




Slika 8: *Tranzistor*
Vir: commons.wikimedia.org, Daniel Ryde, CC-BY-SA, 2004

ksnejša vezja je sprožilo trend razvoja računalnikov, kot jih poznamo danes. Vedno manjši računalniki so "rasli" v računski moči, kompleksnosti, vendar so bili zaradi uporabniške neprijaznosti v začetku rezervirani za elito.

Prvi osebni računalnik Altair 8800, ki ni imel niti zaslona niti tipkovnice (le stikala in lučke), je vseboval Intel 8080 procesor in 256 byteov spomina ter je stal $367. Računalniška zanesenjaka Steven Jobs in Steve Wozniak sta prva sestavila "PC" (angl. Personal Computer) in ga poimenovala Apple. Sledili so izdelki podjetij Commodore, Spectrum in podobnih. Preboj na področju osebnih računalnikov je storil IBM. Odločili so se, da bo IBM PC sestavljen iz standardnih komponent različnih proizvajalcev in s tem omogočili konkurenco; zadržali so le licenco za svoj BIOS – skupno logiko za krmiljenje teh standardnih komponent (vir: Wikipedia).


Slika 9: *Čip (angl. chip)*
Vir: commons.wikimedia.org, Raimond Spekking, CC-BY-SA, 2020

Začetnim trendom zviševanja števila tranzistorjev in posledičnemu zmanjševanju razdalje med njimi ter višanju taktnih frekvenc procesorjev, kar je sčasoma povzročilo nesorazmerno rast potrošnje energije ter pretirano gretje procesorjev, sta sledila dva koncepta, ki sta zaznamovala prihodnost sodobnih računalnikov:

1. več-jedrni procesorji (več procesorjev v enem, povezanih preko skupnega internega vodila na zunanje vodilo; osebni računalniki)

2. več-procesorski sistemi (več ločenih procesorjev, ki vsak zase dostopajo do skupnega vodila; veliki centralni računalniki)

Logika za tema konceptoma je – med delovanjem računalnika je vedno zaseden le del procesorske kapacitete. Če lahko opravila "razbijemo" na več neodvisnih procesov, ki tečejo vzporedno, lahko bolje izkoristimo procesor(je) ("hyperthreading") in s tem dosežemo hitrejše izvajanje operacij. Aktualni trendi pri obdelavi podatkov se nanašajo predvsem na:

- energetsko varčno procesiranje, kar dosegamo predvsem z nižanjem taktne frekvence
- uravnoteženo procesiranje glede na procesno breme ob selektivni uporabi večjega/manjšega števila procesorjev/jeder, kar prav tako vpliva na porabo energije

Glede na tehnologijo, način programiranja in vrsto aplikacij, ki jih poganjajo, ločimo več generacij računalnikov, kot jih povzema Tabela 1.

Tabela 1: *Generacije računalnikov*

| **Generacija** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **Osnovni gradnik** | elektronka | tranzistor | čip | mikro - procesor | mikro - procesor |
| **Programska oprema** | strojni jezik | višji programski jeziki | operacijski sistem, aplik. programi | podatkovne baze | umetna inteligenca |
| **Obdobje / Začetek** | 1946 - 1959 | 1959 - 1964 | 1965-1971 | 1970 | 1980 |

### 1.1.3 Informacijska družba

Od 1954. leta naprej govorimo ”informacijski družbi” ali družbi 4.0. Narava družbe je običajno vezana na prevladujočo gospodarsko dejavnost, s katero je tako ali drugače povezano delovanje aktivne populacije. Če smo v letih 1850-1906 govorili o poljedelski družbi in nato 1907-1953 o industrijski družbi, je delovna aktivnost po letu 1954 tako ali drugače povezana z zbiranjem, obdelavo, posredovanjem ali uporabo informacij – od tod pojem informacijska družba. Za razvite družbe smatramo tiste, kjer je več kot 50% prebivalstva zaposleno v informacijskem sektorju.

Za informacijsko družbo je značilno zbiranje in posredovanje koristnih informacij. Seveda vse informacije, ki jih prejmemo, niso takšne. Zato je ob prejemu potrebno izločanje škodljivih informacij. Pojem filtriranja informacij je zelo širok in od primera do primera različen, lahko pa na splošno rečemo, da so škodljive vse informacije, ki ne povečujejo našega znanja o določeni temi, ker so npr. neresnične, napačnega formata ali nekonsistentne z ostalimi že zbranimi informacijami. Ključna beseda tukaj je *znanje oz. vedenje* in informacijska družba ima za cilj povečevanje le-tega. Od tod tudi pojem "na znanju temelječe družbe". Značilno za takšno družbo je takšno vodenje posameznikov, ki ima za posledico povečanje njihove uspešnosti zaradi povečanja njihovega znanja in ne kot rezultat stroge kontrole njihove dejavnosti. Da bi bilo znanje široko uporabno, mora biti kakovostno in dosegljivo v pravilnem obsegu in formatu, zato je standardizacija v vseh fazah obdelave znanja ključnega pomena. Standardizacija nam v svetovnem merilu tudi zagotavlja enotnost dostopa do znanja, ki je tudi enotno organizirano za učinkovito hrambo in obdelavo ter enako dostopno vsem sodelujočim partnerjem.

V globalnem svetu, kjer se ekonomija vse bolj decentralizira, podjetništvo doživlja nov razcvet zaradi možnosti dostopa do širšega trga ter dodatnih možnosti mednarodnega sodelovanja in poslovanja. Zaradi globalne dostopnosti informacij, se pridelava lažje prilagaja potrebam tržišča in lažje pravočasno identificira deficite in suficite blaga. V odprtem tržišču, kjer je blago globalno dostopno, se povečuje tudi zahtevnost trga, kar stimulira industrijsko in trgovsko tekmovalnost. Zato upravljanje vse pogosteje prevzemajo specializirani strokovnjaki, ki so pripravljeni na sodelovanje na globalnem trgu, kjer je organizacija kolektiva nujno fleksibilna.

Tudi bralci teh vrstic se bodo slednjim kmalu pridružili, zato je pomembno, da se zavedajo navedenih zahtev za sodelujoče na globalnem trgu. Na tržišču znanja, kjer znanstvena informacija postaja vodilni vir pridelave, pri čemer so visoke tehnologije samo materialna podlaga le-te, vse bolj v ospredje stopa potreba po vseživljenjskem-učenju in izpopolnjevanju na vseh nivojih. Prav tako individualne spretnosti in originalne ideje pridobivajo na veljavi in nadomeščajo oponašanje velikih vzorov, kar je bilo značilno za industrijsko družbo.

Za informacijsko družbo so značilni trije modeli:

1. *inovativna*: v državah, ki veliko vlagajo v razvoj in raziskave, razvijajo informacijsko infrastrukturo ter pri tem spodbujajo osebno in kolektivno kreativnost
2. *imitativna*: v državah, kjer v celoti sprejemajo tehnologije in znanja razvitih družb
3. *neinventivna*: v državah, kjer je osebna in kolektivna kreativnost omejena s številnimi administrativnimi in infrastrukturnimi omejitvami

Ključne spremembe, ki jih v vsakodnevno življenje uvaja informacijska družba so:

1. mnoga dela opravljajo roboti
2. informacije so enako dostopne skoraj vsem članom kolektiva, vsi pa so informirani o pomembnejših dogodkih
3. delo je manj fizično in rutinsko
4. nov odnos do dela, ki prinaša zadovoljstvo
5. spoštovani so vsi člani kolektiva
6. nagrajuje se uspešno opravljeno delo
7. možnosti izpopolnjevanja pri delu
8. možnosti realizirati tudi lastne zamisli – ne samo naloge
9. fleksibilen delovni čas
10. delavci so delničarji
11. kooperativen odnos pri delu

12. nova vloga managementa – ne vodenje, ampak spodbujanje talentov sodelavcev

Od tod lahko sklepamo, da v teh pogojih ni več potrebe po nadzoru sodelavcev. Le-ti delajo po najboljših zmožnostih, kjer in kadar so za to podani pogoji in potrebe in so za dobro delo tudi ustrezno stimulirani. Posledično organiziranost ni več pretežno hierarhična, ampak se veže na pretok informacij, ki so potrebne za sodelovanje pri delu. Odločilni faktor uspeha ni več le kapital, energija ali material, temveč v ospredje stopajo informiranost, znanje in usposobljenost.

Nahajamo se na prehodu med dvema različnima ekonomskima modeloma:

1. ekonomiji masovne proizvodnje in potrošnje
2. informacijsko-orientirani ekonomiji

Novi ekonomski model (2) teži k uporabi informacij s ciljem ustvarjanja *inteligentnega proizvoda* (boljša kakovost, uporabnost, življenjska doba, okoljska vzdržnost,...). Vloga informacij pri takšnem proizvodu naj bo zmanjšanje potrošnje energije, zmanjšanje količine potrebnega materiala in obsega vloženega dela, lažje vzdrževanje – skratka *boljša logistika*. Od tu s sinergičnim učinkom področji informatike in logistike vodita *informacijsko družbo* v *inovacijsko družbo*.

## 1.2 Osnovni zakoni informacijske tehnologije

Preden se posvetimo osnovnim zakonom, ki veljajo za računske sisteme in so delno matematično-logične narave, po drugi strani pa zanje, ker gre za električne naprave, seveda veljajo s tem povezane fizikalne zakonitosti, se spomnimo da ima informatika izvor v sistemski znanosti. V skladu s sistemsko

teorijo velja, da vsak sistem vrši transformacijo vhodnih podatkov (vhod) v izhodne rezultate (izhod) s pomočjo ustreznih postopkov (obdelava) – enako seveda velja za vsak računski sistem (Slika 1).

V vsakem realnem (fizičnem) sistemu lahko identificiramo materialne, energetske in informacijske tokove. Da bi lahko deloval v skladu s pričakovanji, sistem potrebuje informacije, ki jih delno lahko pridobi iz okolja, delno pa jih lahko proizvede sam – s pomočjo računalnikov. Vemo, da brez zadostnih informacij v vsakem sistemu narašča entropija. Le-ta ima v sistemu enak pomen kot informacija, le nasproten učinek.

Pravilna in pravočasna informacija, v pravem obsegu in formatu je tista, ki jo vsak sistem potrebuje da izpolnjuje svoj smoter. Od tod lahko sklepamo tudi, da ima informatika v logistiki sinergični učinek na logistični sistem, kjer izhodne informacije sistema po količini in kakovosti prekašajo vhodne.

### 1.2.1 Kompleksnost

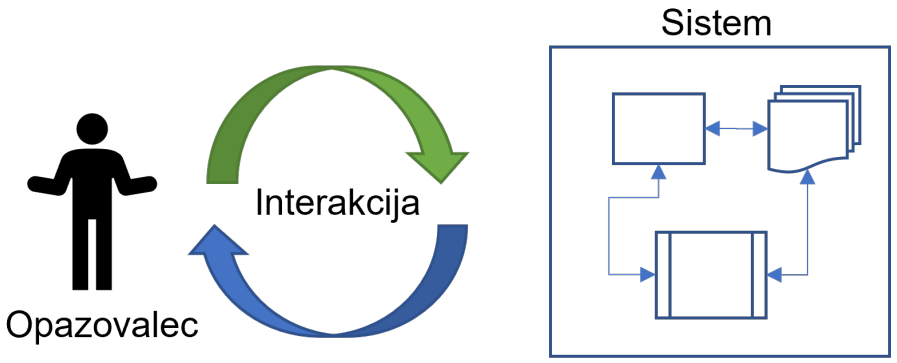
Delovanje komponent ocenjujemo skozi prizmo doseganja cilja celotnega sistema. Več kot jih je, več je tudi možnih povezav med njimi in težavneje je uskladiti njihovo delovanje. Od tod pojem kompleksnosti.

***Kompleksnost*** *zajema vse, kar je težko uvideti (Flood, 1987) – sistem to lastnost pridobi ob opazovanju ali delovanju opazovalca na sistem.*

V duhu sistemske teorije kompleksnost nastopa kot posledica interakcije opazovalca in sistema (Slika 10).

Glede na izvor ločimo:

1. sistemsko kompleksnost (“strukturna kompleksnost”) – izvira iz sistema samega; gre za kompleksnost v omenjenem smislu




Slika 10: *Interakcija med opazovalcem/uporabnikom in sistemom*
Vir: lasten

2. projektantsko kompleksnost – izvira iz analize/opazovanja sistema; pogosto smo v situaciji, ko sami nismo načrtovalci določenega sistema in ga spoznavamo kot opazovalci
3. kompleksnost delovanja – povratno delovanje opazovalca na sistem; ob preizkušanju sistema, če to možnost seveda imamo, lahko do določene mere ugotovimo njegovo delovanje
4. interakcijsko kompleksnost (“kompleksnost obnašanja”) – glede na tip interakcije; kompleksnost sistema je odvisna tudi od načina interakcije z njim (npr. najpreprostejša je preko gumbov na tipkovnici, najkompleksnejša pa je avdiovizualna interakcija)

Seveda težimo k enostavnosti sistemov, čeprav glede na enega od ”očetov” računalništva E.W. Dijkstra velja: ”rešitve naj bodo kar se da enostavne, vendar ne enostavnejše”. Kaj je torej enostavnost? Najpreprosteje jo razložimo kot nasprotje kompleksnosti. Pomeni, da sistem:

1. ima malo komponent in povezav med njimi
2. je lahko razčlenljiv na osnovne komponente (na osnovi obnašanja komponent enostavno sklepamo na obnašanje celote)

3. ima centralizirano odločanje (malo centrov odločanja ter maloštevilne ali neobstoječe povratne povezave med odločanjem in delovanjem)...

Od tod vidimo, da tudi razložiti kompleksnost sistema ni enostavno. Glede na prej predstavljene izvore kompleksnosti, bomo morali nekoliko razširiti podano splošno definicijo – najbolj celovito lahko kompleksnost definiramo na naslednji način:

***Kompleksnost*** = *strukturna kompleksnost* + *kompleksnost obnašanja,*
kjer
*kompleksnost obnašanja* = *opazovalčeva kompleksnost* + *interakcijska kompleksnost*

Kako spoznamo kompleksen sistem? Nasploh lahko rečemo, da gre tu za vsak sistem, ki se obnaša "nepredvidljivo" (asimetrija, nelinearnost, aperiodičnost) oz. je njegovo obnašanje težko uvideti. Če imamo vpogled v sistem, relativno enostavno objektivno ocenimo njegovo strukturno kompleksnost – število komponent in povezav med njimi. Na drugi strani pa je interakcijska kompleksnost "v očeh opazovalca" oz. jo le-ta ocenjuje iz svojega vidika – je torej odvisna od nivoja interakcije (npr. voznik avtomobila in inženir bosta kompleksnost nekega avtomobila ocenjevala različno, čeprav gre za isti sistem). Nasploh lahko trdimo:

***Kompleksen sistem*** *je težko razčlenljiv sistem, ki ima veliko medsebojno povezanih komponent, sistem odločanja v sistemu pa je decentraliziran, kar ima za posledico negotovo obnašanje, ki izvira iz nepredvidljivosti in nerazložljivosti.*

Obvladovanje kompleksnosti predstavlja izboljšanje naše sposobnosti reševanja nalog vezanih na kompleksne sisteme. V zvezi s tem lahko postavimo dve tezi:

1. "vsaka rešitev, ki ne vodi v katastrofo in daje nek odgovor na zastavljen problem, je uspešna", ker nam povečuje znanje o sistemu

2. "bolje je ne dobiti rešitve kompleksnega problema kot dobiti napačno rešitev", ker nas pri spoznavanju sistema lahko zavaja

Pristopi k razreševanju/obvladovanju kompleksnosti so različni:

1. metode agregacije (združevanja):
   (a) primerne v primeru strukturne ali organizacijske kompleksnosti
   (b) elemente sistema združujemo v podsisteme na osnovi sorodnega obnašanja ali prostorske razporeditve
2. metode abstrakcije (posploševanja):
   (a) primerne v primeru nezadostne deterministočnosti sistema
   (b) zmanjšujemo nivo podrobnosti (npr. uporaba intervalov namesto posameznih vrednosti, uporaba opisnih namesto kvantitativnih vrednosti, ipd.)

Ena od znanih metod zmanjševanja kompleksnosti je kvalitativno modeliranje. Kvalitativno modeliranje je postopek modeliranja na osnovi simboličnih vrednosti. Na drugi strani imamo kvantitativno modeliranje. Kvantitativno modeliranje je postopek modeliranja na osnovi matematično-logičnih povezav med komponentami. Nasploh lahko rečemo, da kvalitativno modeliranje praviloma zmanjšuje natančnost, povečuje pa uporabnost modela sistema, ki smo ga oblikovali, da bi uvideli njegov smoter, strukturo in obnašanje.

**Primer**: kvalitativno modeliranje – spreminjanje vrednostnega prostora:

1. najmanjše podrobnosti (+/-): "vklop/izklop", "je/ni",...
2. triadni sistem je bolj natančen (0/+/-): uvaja nevtralno stanje
3. štirivalentna logika (0/+/-/?): uvaja nedoločenost "?"

4. višjo stopnjo podrobnosti uvaja t.i. ”meglena” (angl. “fuzzy”) logika: “visoka/nizka sobna temperatura”, “suho/deževno/soparno/ugodno vreme”,...

Pravimo, da nam tehnika kvalitativnega modeliranja s pomočjo usmerjanja naših misli pomaga soočiti se s kompleksnostjo sistema in jo obvladati. Pristopi so različni:

1. strukturno modeliranje (kognitivne mreže)
2. kvalitativna fizika temelji na znanstveno-izkustvenih pristopih na osnovi: komponent, procesov in/ali omejitev...

### 1.2.2 Informacija

“Oče” teorije informacij je po splošnem prepričanju Claude Shannon, ki je v svojem delu “A Mathematical Theory of Communication” (1948), kasneje objavljenem v knjigi (Shannon & Weaver, 1963), obdelal problem prenosa informacije po kanalu s šumom.

Poglavitna izsledka te raziskave sta dva teorema:

1. *Kodiranje vira*: “število bitov, potrebnih za predstavitev naključnega dogodka, je odvisno od njegove *entropije*”.
2. *Kanal s šumom*: “zanesljiva komunikacija po kanalu s šumom je mogoča, če je hitrost komunikacije manjša od kapacitete kanala”.

Oba teorema predstavljata osnovo teorije kodiranja, ki v praksi ugotavlja kapaciteto komunikacijskega kanala z ustreznim kodiranjem in dekodiranjem podatkov.

Shannon je *informacijo* opredelil kot *znanje, ki zmanjša negotovost, povezano s pojavom določenega dogodka iz končne množice možnih dogodkov*. Gre za spoznavanje sistema na podlagi interakcije z njim – spoznavamo ga glede na odziv sistema.

Informacijo, ki jo pridobimo s tem, ko izvemo, da se je pripetil določen dogodek, lahko izračunamo po formuli:

$$I = -log_2 p(x)[bit]$$

kjer je:

1. podatek x dejstvo, da se je zgodil dogodek x (kjer je X slučajna spremenljivka, ki ima določeno verjetnostno porazdelitveno funkcijo in verjetnost p(x), da zavzame vrednost x iz množice vrednosti X)
2. znanje I količina informacije, ki jo prejmemo, ob nastopu dogodka x iz množice X
3. *bit* (**bi**nary digi**t**) je merska enota informacije

**Primer**: 1 bit (b) informacije dobimo z odgovorom na vprašanje pri katerem sta možna natanko dva enako verjetna odgovora. Če mečemo tri kovance bi naj torej dobili 3 bite informacije. Imamo 8 možnih enako verjetnih izidov. Pa preverimo:

$$I = -log_2 \frac{1}{n} = \frac{log8}{log2} = 3$$

Najmanjši naslovljivi podatek v računalniškem spominu je zlog (angl. byte), ki je dolg 8 bitov. Pa preverimo še, koliko različnih podatkov x iz množice X lahko predstavimo z enim zlogom informacije:

$$n = 10^{I \cdot log2} = 10^{8 \cdot log2} = 256$$

Sedaj, ko vemo, kaj predstavlja in kako jo merimo, že lahko postavimo tudi bolj operativni definiciji informacije in podatka.

***Informacija*** *je vsako novo (spo)znanje, ki prinaša v sistem dodano vrednost.*

*in*

***Podatek*** *je nosilec informacije.*

V računalniškem svetu imajo podatki binarno predstavitev in so *fizično* shranjeni v pomnilniku, *logično* pa so povezani v podatkovne zbirke/baze/skladišča.

Kapaciteto kanala (hitrost prenosa podatkov po kanalu) s šumom lahko izračunamo po formuli:

$$C = B \cdot log_2\left(1 + \frac{S}{N}\right)\left[\frac{bit}{s}\right]$$

kjer je

**B** pasovna širina kanala [Hz]

**S** povprečna jakost sprejetega signala [W]

**N** povprečna jakost šuma [W]

$S/N$ v elektroniki pogosto označujemo tudi kot razmerje signal/šum.

**Primer**: WiFi omrežje deluje na frekvenci 2,5 GHz. Recimo, da imamo ugodno razmerje $S/N = 2$, kar pomeni, da je signal enkrat močnejši od šuma. Kakšna je kapaciteta takšnega kanala?

$$C = 2{,}5 \cdot log_2(1+2) = 3{,}96\left[\frac{Gbit}{s}\right]$$

### 1.2.3 Paralelno izvajanje

V računalništvu, pa tudi v logistiki, nas pogosto zanima pohitritev sistema, ki mu dodajamo resurse, da bi povečali njegovo zmogljivost. Le-ta seveda ni nujno linearna v vsakem primeru oz. za poljuben problem. Iz različnih razlogov je ponavadi samo del bremena ali sistema takšen, da je takšna pohitritev možna.

Po Amdahlovem zakonu lahko pohitritev izvajanja s paralelno obdelavo izračunamo na podlagi naslednje formule:

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

kjer je P delež obdelave (1 = 100%), ki ga lahko pohitrimo, N pa je faktor pohitritve tega dela.

**Primer**: naš problem nam omogoča vzporedno izvajanje 12% operacij, preostalih 88 % pa zahteva zaporedno obdelavo. Za koliko torej lahko teoretično pohitrimo naš sistem?

$$S(\infty) = \frac{1}{1-0,12} = 1,136$$

saj ob tem predpostavimo, da je N, ki ga ne poznamo, neskončno velik.

Od tod vidimo, da našega sistema ne moremo bistveno pohitriti oz. da je njegova pohitritev skoraj enaka deležu, ki ga lahko pohitrimo, tudi če imamo na voljo neskončne resurse.

**Primer**: obratno nas lahko zanima tudi ali se nam investicija splača, oz. kolikšen delež bremena mora minimalno nositi dodatna oprema, da bomo dosegli zaželeno pohitritev.

Predpostavke:

- N=10 (nova enota je 10 krat hitrejša od obstoječe)
- S(N)=2 (doseči želimo 2 kratno oz. 100% pohitritev obdelave)

Zanima nas P (delež, ki ga lahko pohitrimo):

iz

$$S = \frac{N}{N(1-P)+P}$$

sledi

$$P = \frac{N(S-1)}{S(N-1)} = \frac{10(2-1)}{2(10-1)} = 0,5556$$

Hitrejša enota mora torej prevzeti vsaj 56% vhodnega bremena, da bi lahko s paralelnim izvajanjem dosegli 2-kratno skupno pohitritev izvajanja.

Naš problem lahko seveda razširimo na več deležev bremena, ki jim zaradi različno hitrih namenskih procesnih enot, ki jih bodo izvajale, pripišemo ločene faktorje pohitritve.

**Primer**: naša obdelava ima 4 dele, med katerimi 11% ne moremo paralelizirati, s paralelizacijo ostalih delov pa lahko dosežemo različne faktorje pohitritve (npr. z dodajanjem enakega števila paralelnih procesorjev ali dodajanjem le-teh samo določen delež časa delovnega cikla):

Predpostavke:

- P1=11% (ne moremo pohitriti) N1=1
- P2=18% (lahko pohitrimo 5 krat) N2=5
- P3=23% (lahko pohitrimo 20 krat) N2=20
- P4=48% (lahko pohitrimo 1,6 krat) N2=1,6

Vsota vseh deležev P mora biti 100 % – tako vemo, da smo upoštevali celotno vhodno breme. Nov skupen čas obdelave bo glede na dosežene pohitritve ustrezno krajši:

$$T' = \frac{P1}{N1} + \frac{P2}{N2} + \frac{P3}{N3} + \frac{P4}{N4} = \frac{0,11}{1} + \frac{0,18}{5} + \frac{0,23}{20} + \frac{0,48}{1,6} = 0,4575$$

Od tod lahko zaključimo, da je maksimalna možna skupna pohitritev:

$$S = \frac{1}{T'} = 2,1858$$

### 1.2.4 Razvojni trend računalnikov

Moorov zakon opisuje dolgoročen trend v razvoju računalnikov: “od iznajdbe tranzistorja se število le-teh, ki jih je možno cenovno učinkovito vgraditi v integrirano vezje, povečuje eksponentno - podvoji vsaki dve leti”. Ta trend, ki se kljub drugačnim navedbam strokovne javnosti še nadaljuje, je med prvimi opazoval in opisal eden od soustanoviteljev Intel-a Gordon E. Moore ter svoje prve izsledke (Slika 11) objavil že leta 1965. Moorov zakon je pomemben za razvoj elektronskih komponent, iz katerih so zgrajene računalniške naprave, saj razen na povečanje njihove računske moči vpliva tudi na kapacitete pomnilnih medijev, natančnost senzorjev (npr. število slikovnih pik v senzorjih digitalnih fotoaparatov), ločljivost matrik televizijskih ekranov, ipd.

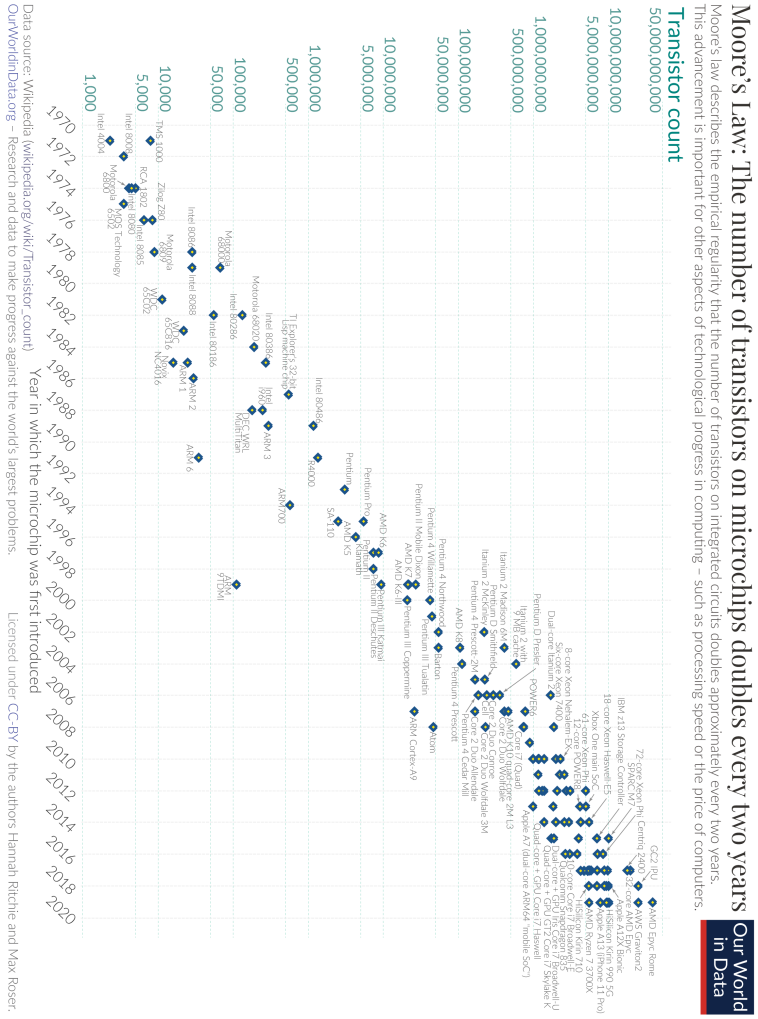
Zaradi segrevanja, interference in potrebe po zmanjšani porabi energije se je omenjeni trend že nekajkrat upočasnil, vendar se ni ustavil. Nadaljnji razvoj omogoča hiperskaliranje računalniških komponent in uporaba novih materialov, se pa povečujejo stroški razvoja takšnih komponent, o čemer govori drugi Moorov zakon. Ob tem je treba pripomniti, da ne gre za naravni zakon, ampak za predvidevanje, ki temelji na opazovanju.

### 1.2.5 Zanesljivost računalniške opreme

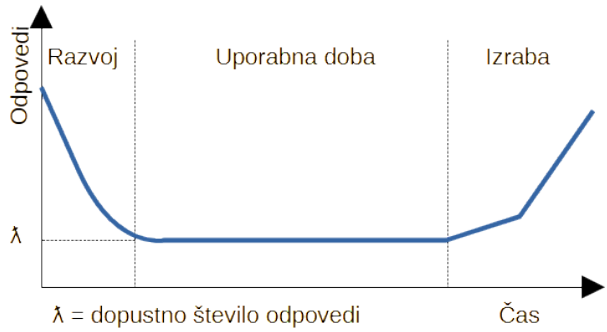
Računalniška oprema je tako kot vsak drug fizikalen sistem podvržena staranju. Ker pa gre za kompleksne sisteme, katerih deli se ne starajo na enak način in tudi ne enako hitro, je dobro da se tega zavedamo, saj je od tega odvisna zanesljivost računalniških sistemov.

V grobem delimo računalniško opremo na strojno (angl. hardware, HW) in programsko (angl. software, SW). Zanesljivost strojne opreme se nanaša na pojav fizičnih napak, medtem ko se zanesljivost programske opreme nanaša na postopkovne napake v delovanju, ki jih je težje vizualizirati, klasificirati in korigirati. V obeh primerih mislimo samo na napake, ki vodijo v odpoved oz. napačno delovanje sistema. Medtem, ko pri odpovedih strojne opreme običajno lažje ugotovimo ne-delovanje in z zamenjavo ustrezne(ga) (dela) opreme relativno enostavno odpravimo, je to pri programski opremi težje.

Značilno za računalniško strojno opremo je, da so v začetni fazi njenega razvoja odpovedi pogoste (t.i. otroške bolezni). Ko tehnologija dozori, pa so napake redke in se običajno ustalijo pri določenem majhnem številu, ki ne presega normalnega industrijskega izmeta. Zaradi končne kakovostne kontrole takšni proizvodi tudi zelo redko sploh pridejo na trg. Ko strojna oprema deluje določen čas in se bliža zaključku predvidene ”življenjske dobe”, pa so napake spet pogostejše in kmalu končajo z dokončno odpovedjo (dela) opreme.



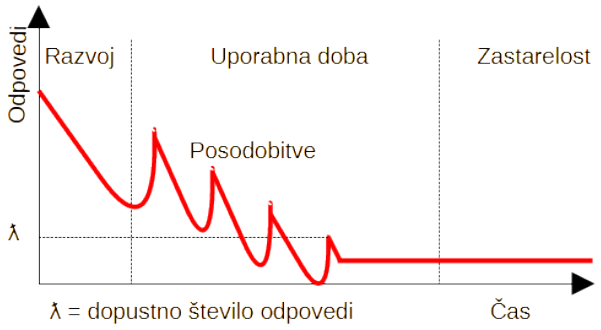

Slika 11: *Moorov zakon*
Vir: commons.wikimedia.org, Max Roser, Hannah Ritchie, CC-BY, 2020




Slika 12: *Zanesljivost računalniške strojne opreme*
Vir: lasten

Odpovedi računalniške programske opreme izvirajo iz napak v načrtovanju, programerskih napak, lahko pa so tudi posledica napačne uporabe ali vnosa napačnih podatkov (princip GIGO – ”smeti noter, smeti ven”). Do svoje zrele dobe – stabilnega delovanja – so odpovedi programske opreme (podobno kot pri strojni opremi) pogoste. Za razliko od strojne opreme v zreli dobi in po njenem poteku odpovedi praktično ni ali pa so le-te omejene na napake zaradi napačnega vnosa podatkov. Programska oprema ni podvržena staranju, saj ne gre za oprijemljivo (fizično) računalniško komponento, ampak za logiko po kateri deluje računalniška strojna oprema.

Zanesljivost računalniške strojne (Slika 12) in programske opreme (Slika 13) skozi njuno življenjsko dobo ponazarjata krivulji, ki sta zaradi značilne oblike dobili ime po “kopalni kadi” (Keene, 1994).




Slika 13: *Zanesljivost računalniške programske opreme*
Vir: lasten

### 1.2.6 Nedoumljivost

Informacijske tehnologije so zaradi svoje raznovrstnosti in prepletenosti težko doumljive nestrokovnjakom, saj tudi strokovnjaki lahko podrobno obvladajo samo posamezne segmente. Zaradi kompleksnosti so jim zato v začetku nekateri pripisovali nadnaravne sposobnosti. V slengu računalniški strokovnjaki v zvezi z računalniki ali računalniško opremo še dandanes pogosto uporabljajo izraze in fraze, ki izvirajo iz vsakdanjega življenja in s tehnologijo nimajo neposredne zveze:

- ”Čaronalnik.” (kot oznaka za računalnik, ki ”avtomagično” nekaj stori)
- ”Pletenje programov.” in ”Špageti koda.” (kot oznaka za programsko kodo)
- ”Če v računalniškem programu nismo našli nobene napake, smo najbrž kaj spregledali.” in končno
- “Če lahko kaj gre narobe, bo tudi šlo.” ali iz vsakdanjega življenja
- “Kruh ponavadi pade na namazano stran.”

Te izjave jasno kažejo na zahtevnost dela računalniških strokovnjakov, prepletenost njihovega dela z realnim življenjem, pa tudi na dejstvo, da je večino problemov najlažje prenesti, če že ne rešiti, s humorjem. Enega najbolj nazornih primerov "religije" računalniških znanosti, ki kažejo na izredno kompleksnost računalniško podprtih informacijskih sistemov, predstavljajo t.i. Murphyjevi zakoni.

# 2 Informacijski sistemi v logistiki

Informacijski sistemi so obstajali in še obstajajo seveda tudi brez računalnikov, vendar v tem primeru za predstavitev informacij moramo uporabiti drug medij, ki nam prav tako omogoča zbiranje, hranjenje, obdelavo ter predstavitev podatkov in rezultatov. Ker gre pri tej "ročni obdelavi" običajno za papir, od tod izvira tudi pojem "papirna vojna".

*Informacijski sistemi (angl. Information Systems, IS)* morajo zagotoviti izvedbo naslednjih nalog:

1. Zbiranje podatkov
2. Obdelavo podatkov
3. Hrambo podatkov in informacij
4. Dostavo podatkov uporabnikom

Ključno za logistične IS (LIS) je izpolnjevanje ciljev logistike:

***Logistika*** *je del oskrbovalne verige, ki načrtuje, izvršuje in upravlja zmogljiv in učinkovit pretok in skladiščenje blaga, storitev ter ustreznih informacij od mesta izvora do mesta potrošnje, da bi zadovoljila potrebe odjemalca.*

Njihov namen izvira iz definicije logistike kot akterja, ki zagotavlja učinkovit pretok in skladiščenje blaga in informacij med ponudniki in odjemalci:

***Logistični informacijski sistemi (LIS)*** *so vsi informacijski sistemi, katerih namen je zbiranje, hranjenje, obdelava in posredovanje podatkov o informacijskem toku ljudi, blaga in storitev znotraj in med organizacijami.*

Vloga logističnega informacijskega sistema (LIS) je priskrbeti *(1) prave informacije, (2) pravim ljudem, (3) pravočasno, v (4) pravem obsegu in formatu* ob sprejemljivih stroških ter ohranjanju ali povečevanju konkurenčne prednosti podjetja. Gre za znani logistični *4P princip*.

## 2.1 Računalniško podprti informacijski sistemi

*Računalniško podprti IS (angl. Computer Based Information Systems, CBIS)* imajo naslednje komponente:

- *računalniška strojna oprema (angl. Hardware, HW)*
- *računalniška komunikacijska oprema (angl. Netware, NW)*
- *računalniška programska oprema (angl. Software, SW)*
- *računalniška podatkovna oprema (angl. Dataware, DW)*
- *računalniška organizacijska oprema (angl. Orgware, OW)*
- *ljudje (angl. Liveware, LW)*

Sodobni računalniško podprti IS imajo razen že naštetih še dodatno nalogo, ki ji pravimo *podatkovno rudarjenje (angl. data mining, DM)* in predstavlja raziskovanje povezav med podatki in pridobljenimi informacijami z namenom generiranja novih informacij (t.j. podatkov z dodano vrednostjo). Gre za funkcionalnost, ki jo s pridom izkorišča *poslovna inteligenca (angl. business intelligence, BI)*. V organizacijah je okoli njihovih informacijskih sistemov zasnovana poslovna strategija, katere namen je ohranjanje/povečevanje konkurenčne prednosti – informacijski sitem ni nikoli samemu sebi namen!

Računalniško podprti informacijski sistemi imajo veliko prednosti:

1. Izvršijo veliko število numeričnih operacij v kratkem času
2. Omogočajo zanesljivo komunikacijo znotraj in med organizacijami
3. Hranijo velike količine podatkov in informacij, do katerih dostopamo hitro in preprosto ter jih učinkovito hranimo
4. Omogočajo enostaven, hiter in cenovno učinkovit pristop do informacij z različnih fizičnih lokacij

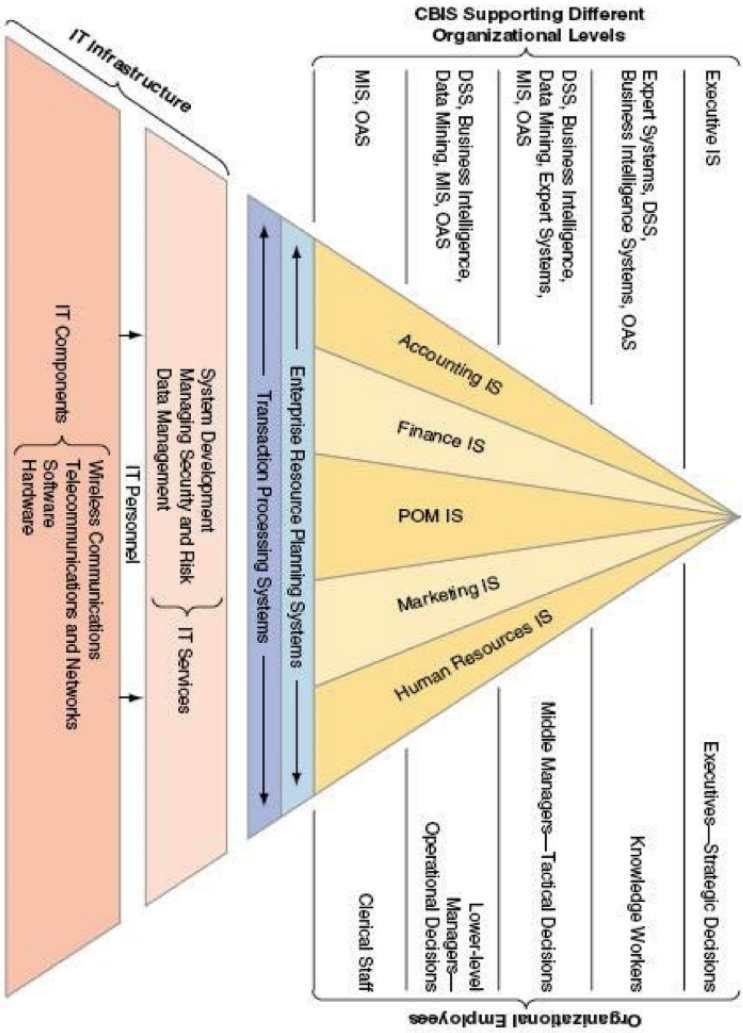
5. Izvajajo interpretacijo velike količine podatkov
6. Povečujejo učinkovitost sodelovanja delavcev na različnih fizičnih lokacijah
7. Avtomatizirajo večino poslovnih procesov

## 2.2 Struktura logističnih informacijskih sistemov

Znotraj vsake organizacije računalniško podprti LIS podpirajo *informacijske procese*, ki avtomatizirajo in pohitrijo cikel odločanja, vodenja in nadzora. Ti povezujejo temeljne procese, ki se nanašajo na njeno osnovno dejavnost, z upravljavskimi procesi, ki jih koordinirajo in oskrbujejo z vsemi potrebnimi viri. Vsak poslovni sistem definirajo omenjeni trije osnovni procesi: *temeljni, upravljavski in informacijski* (Slika 14). Temeljni predstavlja osnovno dejavnost poslovnega sistema, upravljavski vodenje in nadzor tega sistema, informacijski pa vso potrebno izmenjavo informacij pri tem. Vloga računalniško podprtega logističnega informacijskega sistema (LIS) je olajšati, pohitriti in narediti pretok informacij zanesljivejši kot bi bil zgolj pretok dokumentov in ustnih navodil.

LIS morajo torej poskrbeti predvsem za usklajenost poslovnih procesov ter blagovnih, denarnih in informacijskih tokov znotraj in med organizacijami. Sodobne logistike si brez podpore računalniško podprtih LIS ne znamo več predstavljati. Omogočajo nam logistično (strateško, taktično in operativno) odločanje ter upravljanje oskrbovalne verige. Kot smo že nakazali v uvodu, LIS služijo predvsem različnim nivojem managementa pri opravljanju njegove upravljavske funkcije. Zato je s tem usklajen tudi njihov ustroj, ki je v skladu z najbolj razširjeno mednarodno klasifikacijo celovito prikazan na Sliki 15.


Slika 14: *Poslovni sistem*
Vir: lasten




Slika 15: *Informacijski sistemi znotraj organizacije*
Vir: (Rainer et al., 2006)

S tem, ko se vrhovni management ukvarja predvsem z vprašanjem ”kaj narediti” (npr. z razvojem novih produktov, reorganizacijo poslovanja, uvajanjem novih tehnologij ali raziskavami novih tržišč) so njegove potrebe skoncentrirane predvsem na zmožnost odločanja o tem, kaj to pomeni zanje in za njihov poslovni sistem. Sprejemajo predvsem dolgoročne strateške odločitve. Za ta namen so bili razviti t.i. *izvršilni informacijski sistemi (angl. Executive Information System, EIS)*.

Srednji management je skoncentriran na vprašanje ”kako narediti” in sprejema predvsem srednjeročne taktične odločitve. Te so pogojene s sposobnostjo sodelavcev, tehnično opremljenostjo in podkovanostjo, pogoji poslovanja, specifičnostjo poslovnega procesa ipd. Pri poslovodenju so jim v podporo t.i. *upravljavski informacijski sistemi (angl. management information system, MIS)*. Uporabljajo programska okolja za celovito upravljanje virov poslovanja *(Enterprise Resource Planing, ERP)* ter projektantska programska okolja za *proizvodni in logistični inženiring (Computer Aided Engineering, CAE)*.

Operativni management sprejema odločitve, ki predstavljajo odgovor na vprašanje ”s pomočjo česa narediti”. Vodje sprejemajo kratkoročne, operativne odločitve. Pri tem črpajo informacije iz ERP sistemov, pri delu pa jih podpirajo *transakcijski informacijski sistemi (angl. Transaction Processing System, TPS)*, ki povezujejo programsko-aparaturne sklope za vodenje proizvodnje, kot so na primer proizvodne celice *računalniško podprte proizvodnje (angl. Computer Aided Manufacturing, CAM)* ali celo popolnoma *integrirane robotske proizvodnje (angl. Computer Integrated Manufacturing, CIM)*, in podsisteme upravljanja z oskrbovalno verigo (angl. Supply Chain Management, SCM) – *upravljanje naročil (angl. Order Management System, OMS)*, *upravljanje zalog (angl. Warehouse Management System, WMS)* in *upravljanje transportov (angl. Transport Management System, TMS)*, z upravljavskim nivojem podjetja.

Strokovni sodelavci podpirajo različne nivoje managementa pri njihovem odločanju preko *sistemov upravljanja znanja (angl. Knowledge Management System, KMS)* kot nadgradnje *sistemov za podporo odločanju (angl. Decision*

*Support System, DSS)* oz. *ekspertnih sistemov (angl. Expert System, ES)*. Ti na osnovi tehnologij *poslovne inteligence (angl. Business Intelligence, BI)* združujejo funkcije *poslovne analitike (angl. Business Analytics, BA)* kot nadgradnje *podatkovnega rudarjenja (angl. Data Mining, DM)* in *inženirstva na osnovi znanja (angl. Knowledge Based Engineering, KBE)* kot nadgradnje *računalniško podprtega inženirstva (angl. Computer Aided Engineering, CAE).* V ta namen zbirajo, hranijo, obdelujejo in interpretirajo planske in transakcijske podatke ter na ta način podpirajo odločanje in inženiring na vseh nivojih upravljanja.

S stališča opazovalca so možni različni vidiki informacijskega sistema. *Konceptualni vidik* nam razkriva strukturo in povezave sistema. Pri tem lahko vidimo, da nivoji LIS odsevajo strukturo poslovnega sistema, ki je povezana s poslovnimi pravili, zakoni ter objekti in subjekti, katerih podatke obdelujemo in hranimo v okviru *baze znanja (angl. Knowledge Base)* informacijskega sistema. Ta predstavlja osnovo KMS. Konceptualnemu vidiku sta komplementarna *notranji in zunanji uporabniški vidik*, ki sta odvisna od okolja iz katerega izvira uporabnik. Medtem ko delokrog večine uporabnikov LIS zajema funkcije zunanjega uporabniškega vidika, je notranji *tehnični vidik* predvsem v domeni razvijalcev in vzdrževalcev le-tega ter naslavlja *informacijsko infrastrukturo (IT)*, s katero se bomo podrobneje seznanili v naslednjih poglavjih.

LIS imajo torej povezovalno vlogo v poslovnem sistemu ter zagotavljajo lažjo in boljšo komunikacijo med oddelki, poslovnimi enotami in posameznimi zaposlenimi, omogočajo pa tudi medorganizacijsko povezovanje in sodelovanje s poslovnimi partnerji.

## 2.3 Funkcije logističnih informacijskih sistemov

Funkcije informacijskega sistema oz. uporabniški vidik LIS lahko najbolje predstavimo, če se najprej ozremo na strukturo zaposlenih v organizaciji:

1. uslužbenci (angl. Clerical workers) – podpirajo delo managerjev

2. operativni vodje (angl. Low-level managers) – koordinirajo dnevna rutinska opravila (razporeditev dela, manjša naročila ipd.)

3. srednji in vrhovni management (angl. Middle and Top-level managers) – sprejemajo taktične in strateške odločitve, ki se nanašajo na kratkoročno oz. srednje/dolgoročno planiranje, organizacijo in upravljanje

4. strokovnjaki (Knowledge workers) – specializirani sodelavci, kot so npr. finančni analitiki, inženirji, pravniki,... so svetovalci srednjemu in vrhovnemu managementu

Omenjenim kategorijam zaposlenih so namenjeni naslednji nivoji LIS:

1. Izvršilni informacijski sistemi (angl. Executive Information System, EIS)
   (a) namenjeni izključno izvršilnemu managementu
   (b) strukturirano prikazujejo zbrane in časovno urejene podatke o stanju poslovanja (npr. za določen proizvod, določeno časovno obdobje)

2. Upravljavski informacijski sistemi (angl. Management Information System, MIS)
   (a) namenjeni (predvsem) srednjemu managementu
   (b) ERP (Enterprise Resource Planing) IS zbira, hrani in omogoča upravljanje s podatki, ki jih organizacija potrebuje za učinkovito poslovanje

3. Sistemi znanja in razvoja (angl. Knowledge Management System, KMS)
   (a) namenjeni strokovnim sodelavcem (za posamezna področja)
   (b) sistemi za podporo odločanju (DSS) in ekspertni sistemi (ES)
   (c) poslovna analitika (BA) kot nadgradnja podatkovnega rudarjenja (DM)

(d) inženirstvo na osnovi znanja (KBE) kot nadgradnja računalniško podprtega inženirstva CAE

(e) zbirajo, hranijo, obdelujejo in interpretirajo planske in transakcijske podatke ter podpirajo odločanje in inženiring

4. Transakcijski informacijski sistemi (angl. Transaction Processing System, TPS)

   (a) namenjeni vodjem oddelkov in uslužbencem

   (b) upravljanje z oskrbovalno verigo (angl. Supply Chain Management, SCM) sestavljajo

      i. upravljanje naročil (angl. Order Management System, OMS)

      ii. upravljanje zalog (angl. Warehouse Management System, WMS)

      iii. upravljanje transportov (angl. Transport Management System, TMS)

   (c) upravljanje odnosov z dobavitelji in strankami (angl. Supplier/Customer Relationship Management, SRM/CRM)

   (d) vodenje proizvodnje (angl. Manufacturng Execution System, MES), podprto s procesno krmilnimi in nadzornimi sistemi – PCS (Process Control System) in SCADA (Supervisory Control and Data Acquisition)

Razen po nivojih lahko funkcije LIS razdelimo tudi na tiste, ki podpirajo poslovanje organizacije navznoter in navzven. Poslovanje organizacije navznoter podpirajo oddelčni IS: računovodski, finančni, človeški viri, proizvodnja, marketing. Poslovanje organizacije navzven pa podpirata IS za upravljanje z oskrbovalno verigo (angl. Supply Chain Management) in upravljanje odnosov s partnerji (angl. Supplier/Customer Relationship Management).

V nadaljevanju bomo posvetili našo pozornost predvsem omenjeni informacijski infrastrukturi. Drugim nivojem LIS se bomo podrobneje posvetili pri predmetih višjih letnikov, zainteresirani bralec pa lahko več informacij najde tudi v knjigah (Šuhel & Murovec, 2003) in (Grant et al., 2006).

# 3 Računalniška strojna oprema (HW)

Potem, ko smo že nekaj povedali o kronologiji računalniške obdelave podatkov ter v grobem našteli glavne tehnološke mejnike in generacije računalnikov, se bomo v tem poglavju skoncentrirali na računalniško strojno opremo oz. računalniške sisteme kot takšne in njihove komponente.

Identificirali bomo osnovne strojne komponente računalnika ter predstavili von Neumannovo arhitekturo računalnika, ki je skupna splošno namenskim računalnikom. Potegnili bomo mejo ločnico med računalniško periferno opremo in centralno procesno enoto (CPE) in jih podrobneje predstavili. Prav tako bomo vzpostavili tudi povezavo med ustrojem procesorja in zmogljivostjo računalnikov. Podrobneje bomo predstavili hierarhijo pomnilniških enot v računalniku ter izpostavili razliko med enotami za hrambo podatkov znotraj računalnika in tistimi za hrambo podatkov na nivoju organizacije. Na kratko bomo predstavili tudi multimedijske sisteme in njihove komponente ter ob tem diskutirali trend razvoja računalniške tehnologije. Zaključili bomo z diskusijo o vplivu strojne računalniške opreme na poslovno strategijo.

Računalniki se glede na namembnost razlikujejo med sabo – obstaja nekaj arhetipov, ki so vredni omembe, čeprav aktualna razmejitev med njimi ni več tako stroga. Omenili bomo tudi nekaj novih hibridnih računalniških platform, ki nastopajo v logističnih aplikacijah. Ker bi jih težko umestili v osnovno tipizacijo računskih naprav, po drugi strani pa njihova raznolika raba ne omogoča specifikacije novega tipa računalniških naprav, jih bomo tukaj identificirali z aplikacijami, kjer nastopajo.

## 3.1 Osnovna delitev računalniške strojne opreme

Računalniška strojna oprema predstavlja vidne, otipljive komponente računalnika, ki so namenjene:

1. vnosu podatkov – vhodne (periferne) naprave; preko njih računalnik sprejema podatke (npr. tipkovnica, miška, optični čitalnik, čitalnik črtne kode, kamera,...)

2. obdelavi podatkov – procesor (centralna procesna enota) – vsak računalnik ima vsaj enega; CPE opravlja vse potrebne aritmetične in logične izračune za delovanje računalnika

3. shranjevanju podatkov – pomnilne naprave; med delovanjem računalnik shranjuje podatke v delovni pomnilnik (RAM), ki ga imenujemo tudi primarni pomnilnik, vmesne in končne rezultate pa shranjuje v trajnejši sekundarni pomnilnik (npr. magnetni disk, optične enote, tračne enote,...)

4. za izpis in posredovanje podatkov – izhodne (periferne) naprave (npr. računalniški zaslon, tiskalnik, risalnik,...)

Vsi sodobni komercialni računalniki imajo enak ustroj in jih sestavljajo zgoraj omenjene komponente. Kot lahko zasledimo že pri Babbageovem analitičnem stroju (1837), ki je bil takrat še tehnološko neizvedljiv, je John von Neumann (1946) oblikoval arhitekturo sodobnega računalnika z naslednjimi funkcionalnimi enotami (glej sliko 16):

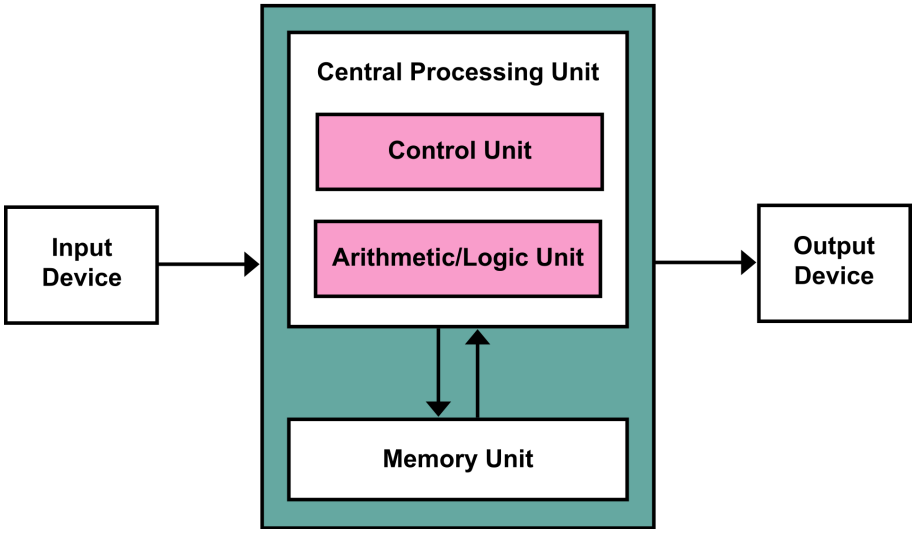
1. krmilna enota – upravljavska enota, ki koordinira delovanje vseh ostalih enot

2. aritmetično/logična enota – računska enota, ki izvaja aritmetične in logične operacije

3. glavni pomnilnik – pomnilna enota, kjer so shranjeni vsi podatki, kakor tudi programi, ki krmilijo delovanje celotnega računalnika

4. vhodne enote – služijo za vnos podatkov iz zunanjega sveta v računalnik

5. izhodne enote – služijo za posredovanje rezultatov računalniških obdelav zunanjemu svetu




Slika 16: *Von Neumannova arhitektura računalnika*
Vir: Wikipedia, Kapooht, CC-BY-SA, 2013

Od tod pojem *von Neumannova arhitektura*. Pomembno je poudariti, da to ni edina računalniška arhitektura (npr. Harvardska, matrična, vektorska,... arhitektura), je pa najbolj razširjena. V osnovi vse uporabljajo enake komponente, le da jih organizirajo in kombinirajo na različne načine zaradi učinkovitosti obdelave v različnih aplikacijah.

Želja von Neumanna je bila ustvariti arhitekturo računalnika, ki bo simulirala delovanje človeških možganov, omogočala večnivojsko sklepanje ter paralelno obdelavo. Oblikoval je osnutek takšne arhitekture, vendar le-te zaradi tehnoloških omejitev še ni bilo mogoče realizirati.

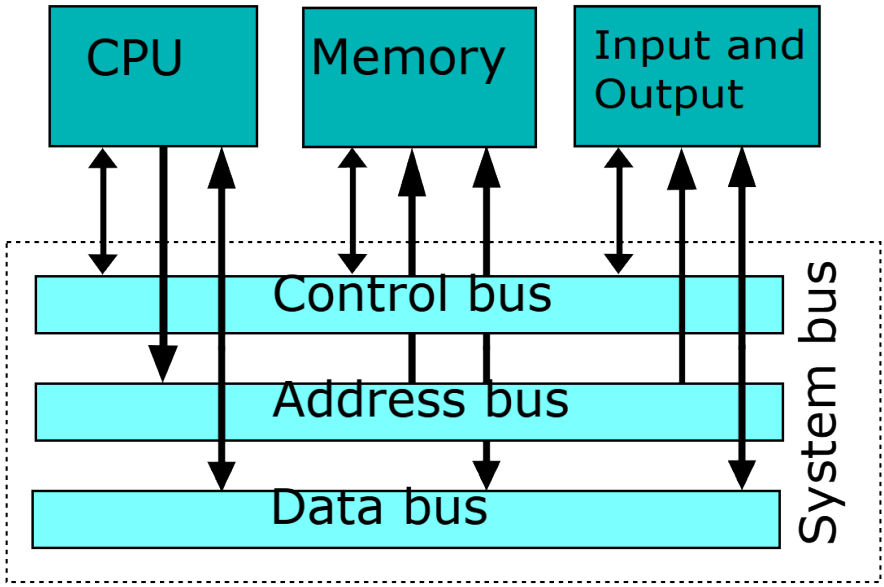
Značilna za von Neumannovo Arhitekturo je zaporedna obdelava ukazov. Leti so shranjeni v *glavnem pomnilniku (angl. Main Memory)*, ki hrani programe (zbirke ukazov) in podatke (števila, znaki, besedilo, slike, zvok, ...). Pomnilnik mora biti zato organiziran dokaj univerzalno – v obliki pomnilnih celic enake velikosti, od katerih ima vsaka svoj naslov, preko katerega dostopamo

do njene vsebine. Celice hranijo ukaze programa, prav tako pa tudi vmesne in končne rezultate obdelav. Osnovni cilj von Neumannove arhitekture je bil, da je struktura računalnika neodvisna od problema, ki ga le-ta rešuje. Program naj bo možno vstaviti od zunaj – ga naložiti v glavni pomnilnik in ga od tam zagnati. Za prenos podatkov med glavnim pomnilnikom in centralno procesno enoto skrbi *krmilna enota (angl. Control Unit, CU)*, ki potrebuje neposredno povezavo z glavnim pomnilnikom. *Centralna procesna enota (angl. Central Processing Unit, CPU)* v izvršilnem ciklu izvaja ukaze, kot mu jih dostavlja krmilna enota. Taktna frekvenca procesorja diktira hitrost zaporedja izvršilnih ciklov in s tem tudi hitrost obdelave. Za samo obdelavo ukazov je zadolžena *aritmetično-logična enota (angl. Arithmetic/Logic Unit, ALU)*, ki ukaze dekodira, zahteva morebitne potrebne parametre in dela z registri procesorja (angl. Registers), kamor shranjuje vmesne rezultate, končne pa (s posredovanjem krmilne enote) shranjuje nazaj v glavni pomnilnik.

Omenjeni *izvršilni cikel (angl. execution cycle)* "pridobi-dekodiraj-izvrši" *(angl. fetch-decode-execute)* je sestavljen iz naslednjih korakov:

1. pridobi naslednji ukaz iz pomnilniške celice, kamor kaže programski števec*
2. dekodiraj ukaz v obliko, ki je razumljiva aritmetično-logični enoti
3. iz pomnilnika prečitaj še morebitne potrebne podatke (operande) in jih shrani v registre procesorja
4. izvrši ukaz
5. vmesne rezultate shrani v registre procesorja, končni pa v glavni pomnilnik
6. programski števec procesorja premakni na naslednji ukaz za izvedbo (nov cikel)

*Programski števec* (angl. Program Counter, PC)* je register procesorja, katerega vsebina vedno hrani lokacijo ukaza za izvedbo v naslednjem izvršilnem ciklu.

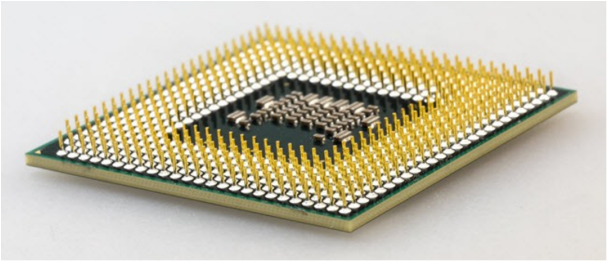



Slika 17: *Razširjena von Neumannova arhitektura računalnika*
Vir: Wikipedia, W Nowicki, CC-BY-SA, 2011

Ker moramo ukaze in podatke na nek način prenesti v računalnik, potrebujemo še povezavo do vhodno-izhodnih naprav (angl. Input/Output System). Prav zaradi lažje izvedbe povezav med omenjenimi komponentami računalnika so uvedli t.i. vodila (angl. Bus), ki jih povezujejo med sabo (Slika 17), nad njimi pa bdi krmilna enota.

Kot kaže Slika 17, imajo vodila namenske linije za določene tipe podatkov, ki jih prenašajo. Zato običajno govorimo kar o treh vodilih:

1. naslovno vodilo (angl. Address Bus) prenaša naslove podatkov za obdelavo



Slika 18: *Procesorski čip*
Vir: commons.wikimedia.org, Fx Mehdi, CC-BY-SA, 2019

2. podatkovno vodilo (angl. Data Bus) prenaša podatke med glavnim pomnilnikom in procesorjem

3. krmilno vodilo (angl. Control Bus) prenaša krmilne signale, ki določajo kateri ukaz se bo naslednji izvajal in katere komponente bodo pri tem sodelovale

## 3.2 Centralna procesna enota

*Centralna procesna enota (CPE)* ali *procesor* izvaja ukaze v *strojnem jeziku* (angl. assembly language, assembler). Ukazi in podatki, ki jih obdeluje, so zakodirani v binarnem sistemu in shranjeni v pomnilniku.

Zmogljivost procesorja je določena s hitrostjo izvajanja operacij in velikostjo podatka (dolžino besede), ki ga procesor lahko obdela naenkrat (v enem izvršilnem ciklu). Krajši kot je izvršilni cikel, več jih lahko izvedemo v časovni enoti in hitrejše je lahko izvajanje. Enostavni ukazi zahtevajo le en izvršilni cikel, da se v celoti izvedejo, kompleksnejši pa zahtevajo več ciklov (npr. vsako pridobivanje dodatnega operanda iz glavnega pomnilnika lahko zahteva dodaten cikel, prav tako pa tudi vsako shranjevanje rezultata v glavni pomnilnik). Skupno število ciklov v sekundi predstavlja hitrost oz. takt procesorja (npr.

računalnik s taktno frekvenco 1GHz izvaja $10^9$ osnovnih operacij v sekundi, kar pomeni, da en cikel traja 1 ns). Sodobni procesorji imajo takt do 3.8 GHz (kar je trenutna tehnološka meja, saj nad 4GHz prihaja do prekomernega segrevanja).

Kot smo že omenili, tudi dolžina procesorske besede pogojuje hitrost procesiranja. Le-ta določa velikost podatkov oz. kompleksnost ukazov, ki jih procesor lahko naenkrat obdela. Glede na število bitov v besedi lahko vanjo zakodiramo cela števila in racionalna števila različnega obsega – daljša kot je beseda, večji je ta obseg oz. natančnost predstavitve in izračuna (s tem se tudi lahko izognemo prenašanju večjih podatkov v več ciklih, kar lahko nastopi, če je njihova dolžina večja kot jih lahko obdelamo naenkrat). Sodobni procesorji uporabljajo kompleksen nabor ukazov (CISC – Complex Instruction Set Computing), ki že na strojnem nivoju omogoča izvedbo raznovrstnih operacij. Za razliko od RISC (Reduced Instruction Set Computing) procesorjev le-ti potrebujejo tudi večjo dolžino besede za lažje in hitrejše kodiranje/dekodiranje ukazov. Sodobni procesorji so 32- ali 64-bitni. Dolžina besede je pogojena z notranjo arhitekturo procesorja (velikostjo registrov, širino notranjega vodila) in zunanjimi napravami (širina zunanjega vodila, dolžina pomnilniške besede).

Veliko sodobnih procesorjev ima tudi interni *hitri predpomnilnik* (angl. *cache*), ki jim omogoča hitrejši dostop do podatkov in izvrševanje ukazov, shranjenih tam. Gre za dodatni interni pomnilnik, ki procesorju omogoča skoraj enako hiter dostop do večjega obsega podatkov, kot jih lahko shrani v registrih procesorja. V hitrem predpomnilniku shranjujemo podatke in ukaze, ki jih krmilna enota ”na mah” prečita iz glavnega pomnilnika. S tem procesorju pri prevzemanju in izvajanju ukazov ter nalaganju dodatnih operandov pogosto ni več treba dostopati do glavnega pomnilnika, ampak te podatke hitreje pridobi iz hitrega predpomnilnika. Enako kot prevzemanje tudi shranjevanje podatkov iz hitrega predpomnilnika nazaj v glavni pomnilnik poteka ”na mah” – periodično ali na zahtevo (ob zahtevi po dostopu do lokacije, ki se ne nahaja v predpomnilniku). Od vrste in količine internega pomnilnika je odvisna arhitektura

procesorja. Nasploh velja – več kot ima internega pomnilnika, manj časa bo procesor porabil za nalaganje podatkov iz delovnega (RAM) ali sekundarnega pomnilnika in hitrejše bo lahko njegovo delovanje.

Hitrost procesorja običajno izražamo z:

1. MIPS (Mega Instructions Per Second) – je enota za hitrost, ki predstavlja milijon izvedenih ukazov v sekundi
2. MFLOPS (Mega FLoating point Operations Per Second) – je enota za hitrost, ki predstavlja milijon operacij z racionalnimi števili oz. števili s plavajočo vejico (angl. floating point)

Glede na izvedbo ločimo več vrst procesorjev. Mikroprocesor je procesor na enem čipu (Slika 18). Mikrokrmilniki so sorodni mikroprocesorjem, vendar so preprostejše zgradbe in namenjeni manj zahtevnim aplikacijam (manjša dolžina besede, običajno ne vsebujejo hitrega predpomnilnika,...). Pogosto so vgrajeni v proizvode, ki v svoji osnovni funkciji niso računalniki (npr. gospodinjski aparati, avtomobili,...). Trend v razvoju mikroprocesorjev so t.i. več-jedrni (angl. multi-core) čipi z več procesorji. Namenjeni so za najzahtevnejše (multimedijske) aplikacije (npr. igre, HD-video,...), ki zahtevajo večjo procesorsko moč oz. hitrost procesiranja.

Pri načrtovanju računalniškega sistema je izbira določenega procesorja ključna, saj ta določa tako nabor kompatibilnih komponent, ki jih lahko na skupnem vezju (npr. matični plošči) povežemo s procesorjem, kot tudi strojni jezik aplikacijskih programov, ki jih bomo na njem lahko izvajali. Posebna kategorija so t.i. sistemi na čipu (angl. System on Chip, SoC), ki (za razliko od sistemov na matični plošči) na istem čipu združujejo vse osnovne komponente računalniškega sistema – procesor, pomnilnik, vhodno/izhodne vmesnike in sekundarni pomnilnik – skupaj s pomožnimi komponentami, kot so modemi, grafični procesorji (angl. graphics processing unit, GPU), ipd. V kolikor gre za namenske procesorje, ti običajno vključujejo tudi komponente za obdelavo digitalnih/analognih signalov.

## 3.3 Matična plošča

*Matična plošča* (angl. Motherboard; Slika 19) je *tiskano vezje* (angl. Printed Circuit Board, PCB), ki povezuje vse komponente računalnika (procesor, pomnilnik, vhodno/izhodne vmesnike, vodila in čipovje, ki jih krmili) v celoto. Če procesorju pogosto rečemo "možgani" računalnika, je *čipovje (angl. chipset*) njegovo "srce" – opravlja vlogo *krmilne enote*.

Čipovje v grobem delimo na:

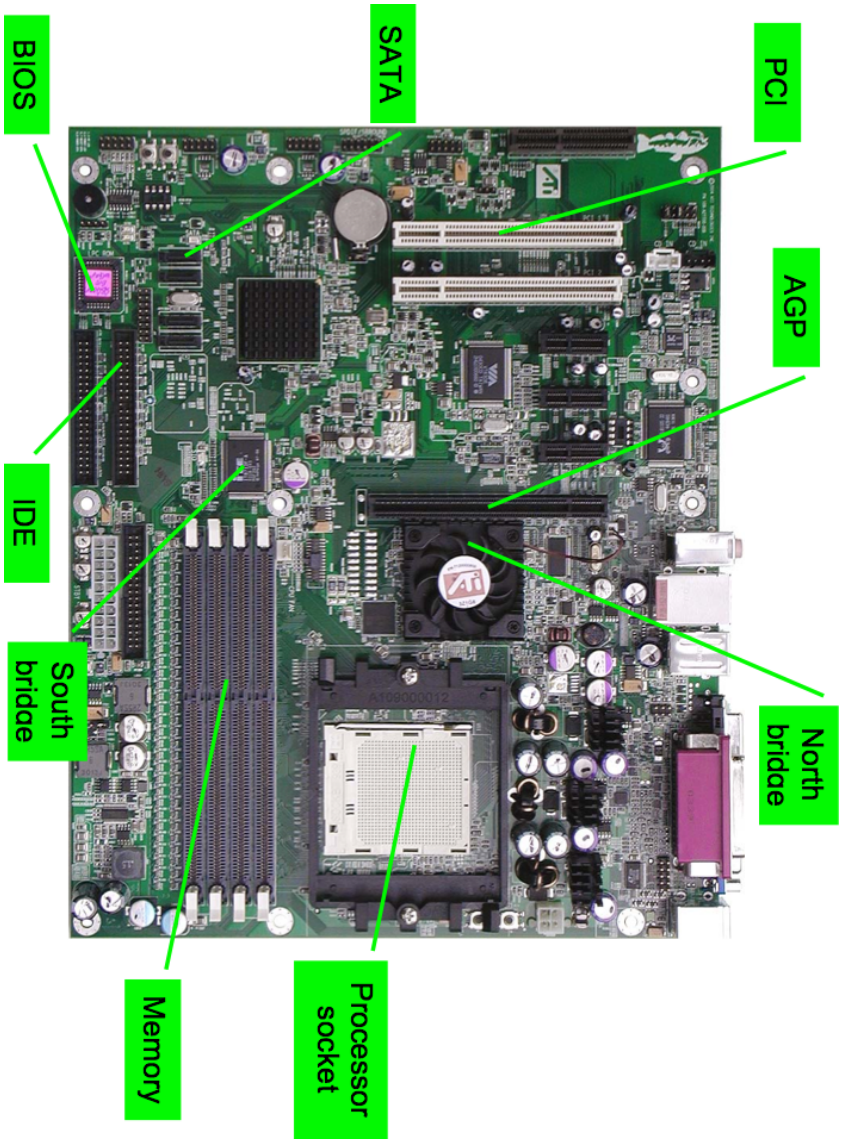
1. North-Bridge, ki nadzira komunikacijo med procesorjem in notranjimi perifernimi napravami (npr. delovni pomnilnik, trdi disk, grafični procesor, ipd.)
2. South-Bridge, ki nadzira komunikacijo med procesorjem in zunanjimi perifernimi napravami (npr. miška, tipkovnica, tiskalnik, optični čitalnik, ipd.)

V arhitekturi osebnih računalnikov je North-Bridge običajno večji čip, ki se na matični plošči nahaja bližje procesorju in nosi pasivni ali aktivni hladilnik, medtem ko je South-Bridge manjši in nekoliko bolj oddaljen od procesorja.

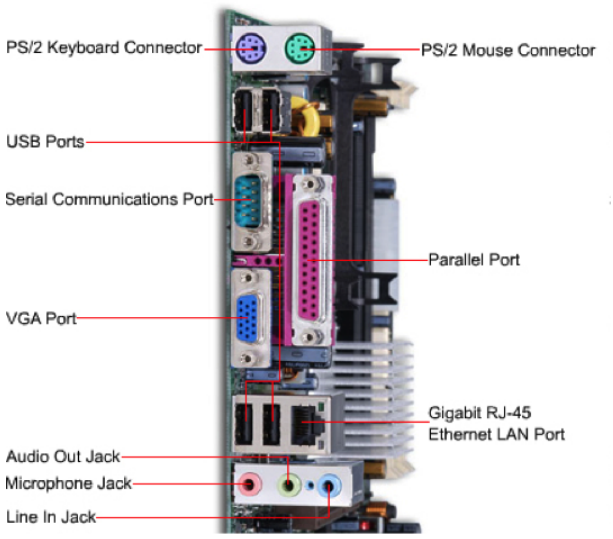
Vodila so sklopi vodnikov, po katerih se prenašajo podatki, naslovi in krmilni signali. Razlikujejo se predvsem po hitrosti prenosa. Za prenos podatkov med različnimi enotami se lahko uporablja isto vodilo, vendar je te prenose potrebno uskladiti (arbitraža vodila), kar je naloga krmilne enote.

Končni člen vodil predstavljajo vhodno/izhodni vmesniki:

1. vrata (angl. ports) za priključevanje zunanjih perifernih naprav (npr. Slika 20) in
2. reže (angl. slots) za priključevanje notranjih perifernih naprav (npr. Slika 19).




Slika 19: *Matična plošča*
Vir: lasten



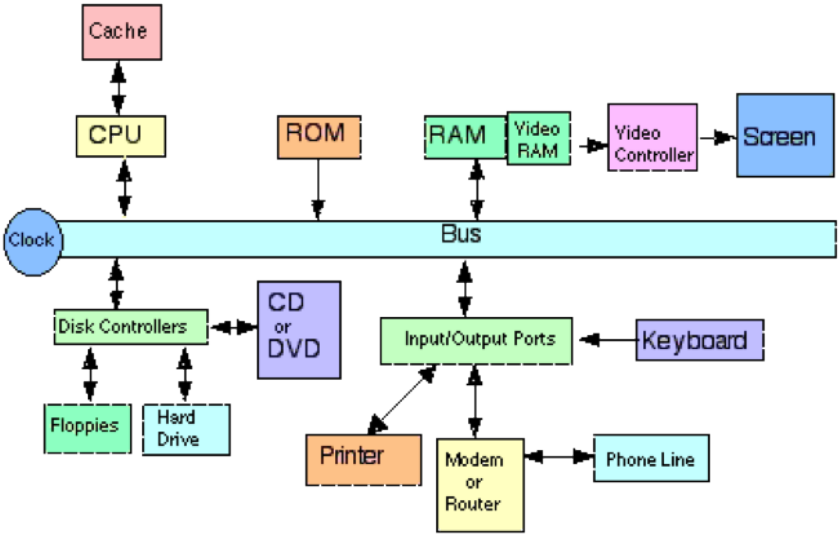

Slika 20: *Vrata na matični plošči*
Vir: lasten

Na matični plošči so lahko razen čipovja nameščene tudi integrirane vhodno-izhodne komponente (npr. grafična ali zvočna kartica, modem, mrežna kartica ipd.), ki običajno ne dosegajo kakovosti/hitrosti priključenih, vendar so praviloma cenejše.

Slika 21 nam pregledno prikazuje vse možnosti priključitve notranjih in zunanjih perifernih naprav na vodilo, ki predstavlja ”interno avtocesto” za vse podatke, ki jih računalniški sistem zajema, obdeluje, hrani ali prikazuje. Na krmilno enoto je priključena tudi ”ura”, ki diktira osnovni takt vodila in s tem posredno hitrost prenosa podatkov priključenih enot.

## 3.4 Pomnilnik

Razlikujemo dva osnovna tipa pomnilnika:




Slika 21: *Vodilo kot centralni vezni člen komponent računalnika*
Vir: http://www.laits.utexas.edu/∼anorman/long/hard-soft.html

1. primarni (angl. primary) – hrani majhne količine podatkov, ki jih v času delovanja lahko procesor takoj uporabi
2. sekundarni (angl. secondary) – hrani večje količine podatkov (podatkovne zbirke, baze, operacijski sistem, aplikacijski programi,...) za daljši čas

Pomnilnik si lahko predstavljamo kot veliko tabelo po zaporednih naslovih urejenih pomnilniških celic, kamor lahko zapisujemo vse vrste podatkov. Glede na organizacijo je dolžina besed v pomnilniku lahko različna (npr. 16-bitna na Sliki 22).

Kapaciteto pomnilnika merimo v *zlogih* (angl. *byte*, B), ki so sestavljeni iz 8 osnovnih enot informacije (*bitov*). Ker so postali zlogi kot enota premajhni, uporabljamo izvedenke te osnovne enote:

1. Kilo-Byte (KB): 1024 B
2. Mega-Byte (MB): 1.048.576 B ali 1024 x 1024 B
3. Giga-Byte (GB): 1.073.741.824 B ali 1024 MB
4. Tera-Byte: 1024 GB
5. Peta-Byte: 1024TB
6. Exa-Byte: 1024 PB

Opazite lahko, da v računalništvu 1 K ni 1000, ampak $2^{10} = 1024$ v smislu binarnega sistema. Medtem ko omenjene enote uporabljamo za izražavo količine shranjenih/prenesenih podatkov, število *bitov v sekundi* (*b/s, bps*) uporabljamo kot osnovno enoto za izražanje hitrosti prenosov po telekomunikacijskih kanalih, ki jih na enak način izražamo z večjimi enotami (npr. sodobno računalniško omrežje ima hitrost 100 Mbps).

| | |
|---|---|
| 2AF0 | 1110010010110101 |
| 2AF1 | 0010011110110101 |
| 2AF2 | 1110010010110101 |
| 2AF3 | 1000010010110111 |
| 2AF4 | 1110010010110101 |
| 2AF5 | 1110010010110101 |
| 2AF6 | 0000010010110101 |
| 2AF7 | 1111111111110101 |
| 2AF8 | 0000010011111111 |

Slika 22: *Shema pomnilnika*
Vir: lasten

### 3.4.1 Primarni pomnilnik

*Primarni pomnilnik* služi za kratkoročno hrambo treh tipov informacij:

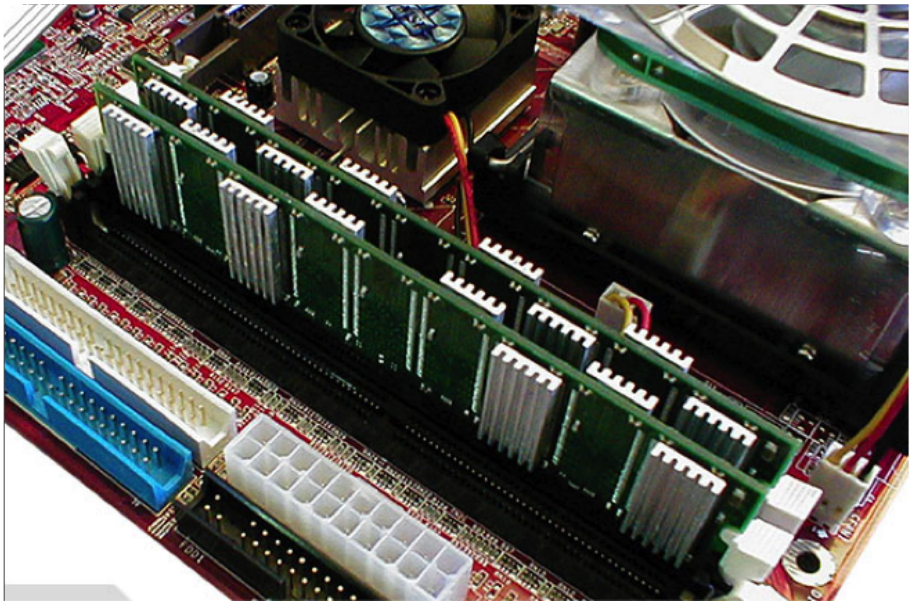
1. podatke, ki jih procesor obdeluje
2. ukaze procesorju, kako naj obdeluje podatke
3. dele operacijskega sistema, ki opravljajo različne funkcije

*Primarni pomnilnik* se običajno nahaja na raznih čipih, ki so vgrajeni ali vstavljeni reže matične plošče (Slika 23). Ločimo štiri glavne tipe primarnega pomnilnika:

1. registri procesorja,
2. delovni – Random Access Memory (RAM),
3. (hitri) predpomnilnik – Cache in
4. vdelan – Read-Only Memory (ROM).

Oznaka *RAM (Random Access Memory)* pomeni *pomnilnik s poljubnim dostopom* (lahko neposredno dostopamo do katere koli pomnilne celice – npr. da bi prebrali sedmo, ne rabimo najprej prebrati prvih šest) oz. *bralno-pisalni* pomnilnik (Read/Write, R/W Memory) – za razliko od *bralnega* ROM (Read-Only Memory). Dva osnovna tipa RAM sta:

1. SRAM (Static Random Access Memory) je drag, vendar zelo hiter RAM; uporabljamo ga za registre in procesorski cache pomnilnik
2. DRAM (Dynamic Random Access Memory) je cenejši od SRAM-a in ga odlikuje večja gostota bitov na čipu, troši pa tudi manj energije in se manj greje; uporabljamo ga za delovni pomnilnik



Slika 23: *Pomnilni čipi delovnega pomnilnika (RAM)*
Vir: lasten

DRAM je lahko dveh tipov:

1. SDRAM (Synchronous Dynamic Random Access Memory )
2. DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory)

Glavna razlika med njima je v hitrosti delovanja in načinu branja/zapisovanja podatkov. Prvi tip lahko zapisuje/čita podatke samo na eni fronti taktnega signala, drugi tip pa lahko te operacije izvaja ob naraščajoči in ob padajoči fronti taktnega signala – zato ju pogosto imenujemo kar SDRAM (Single Data Random Access Memory) oz. DDRAM (Double Data Random Access Memory).

Ločimo tri vrste cache pomnilnika:

1. interni "on-chip" cache je predpomnilnik, integriran na samem čipu procesorja
2. med procesorjem in delovnim pomnilnikom
3. med sekundarnim in delovnim pomnilnikom

Bližje kot smo procesorju/procesorskemu jedru, manjša je njihova kapaciteta, vendar je krajši tudi *dostopni čas* do podatkov, ki jih hranijo. Gre za čas, ki poteče od pojavitve naslova spominske lokacije na naslovnem vodilu do trenutka, ko se njena vsebina pojavi na podatkovnem vodilu.

Glavna pomanjkljivost primarnega pomnilnika je, da ob izključitvi računalnika iz napajanja običajno izgubi vsebino. Izjema pri tem je vdelan spomin *ROM (Read Only Memory)*, ki omogoča samo branje. Njegova vsebina je tovarniško zapisana in je ne moremo niti spreminjati niti brisati – ni odvisna od napajanja. ROM je tipično najmanjši spomin v računalniku. Pri arhitekturi osebnega računalnika (PC) se nahaja na matični plošči blizu South Bridge-a (Slika 19).

Uporabljamo ga za hranjenje upravljavskih programov *BIOS* (*Basic Input Output System*). Po zagonu računalnika se prvi aktivira, zažene osnovno diagnostiko, nakar kontrolo prepusti operacijskemu sistemu.

Obstajajo posebne izvedbe ROM, katerih vsebino je možno spreminjati – programabilni ROM (PROM) oz. električno programabilni (EPROM) ter električno izbrisljiv in programabilen EEPROM. V zadnjem času je pogosta posebna izvedba EEPROM pomnilnikov – t.i. *bliskovni oz. FLASH pomnilnik* (Slika 24). Za razliko od EEPROMa njegove vsebine ne prepisujemo po zlogih ampak v večjih blokih spomina "na mah", operacijo pa imenujemo "flashing" (od tod ime). Je prenosen, kompakten in troši malo energije. Lahko je vgrajen ali pa predstavlja nadgradnjo oz. razširitev internega spomina v obliki *spominske kartice*, kot jih uporabljamo v mobilnih telefonih, digitalnih kamerah, ipd. napravah. Glede na to, da gre v tem primeru za obliko trajnega spomina, ki ga lahko spreminjamo, jih lahko obravnavamo tudi kot obliko sekundarnega pomnilnega medija, o katerih bomo več povedali v naslednjem poglavju.

### 3.4.2 Sekundarni pomnilnik

Če želimo vsebino delovnega pomnilnika ohraniti za ponovno rabo, jo moramo shraniti na pomnilni medij, ki ne izgubi vsebine ob izklopu računalnika – najpogosteje je to t.i. trdi disk (angl. Hard Disk Drive, HDD). V to skupino spadajo tudi na primer gibki diski, magnetni trakovi, optični diski, spominske kartice, ipd.

Sekundarni pomnilnik je namenjen hrambi velike količine podatkov za daljši čas. Zanje so značilne naslednje lastnosti:

1. obstojnost podatkov (angl. non-volatile)
2. dostopni čas je precej daljši kot do primarnega pomnilnika (razlog je njihova elektromehanska izvedba)


Slika 24: *Flash pomnilni mediji*
Vir: lasten

3. so cenejši od primarnega pomnilnika (cena medija na enoto shranjenih podatkov je bistveno manjša)
4. različne tehnološke izvedbe (magnetni, optični,...)

Pri tem je treba pripomniti, da gre za trajno, ne pa večno hrambo. Kot vse elektronske naprave so tudi sekundarni pomnilni mediji podvrženi staranju. Po eni strani tu gre za mehansko obrabo/okvaro gibljivih delov, po drugi pa so magnetni mediji podvrženi tudi razmagnetenju ali popačenju vsebine zaradi škodljivih okoljskih elektromagnetnih sevanj – oboje lahko povzroči delno ali popolno izgubo podatkov. Trendi v razvoju sekundarnih pomnilnih medijev se nanašajo na direkten dostop (RAM), povečano kapaciteto, odstranitev mehanskih delov (npr. SSD (angl. Solid State Drive), ki delujejo po istem principu kot FLASH pomnilniki) in prenosljivost (npr. USB ključi, spominske kartice).

Magnetne trakove (Slika 25) so običajno uporabljali v centralnih računalniških sistemih za periodično izdelavo varnostnih kopij podatkov večjih informacijskih sistemov (npr. banke, zavarovalnice, javna uprava,...). Pogoni, ki so jih uporabljali za branje in zapisovanje trakov so bile t.i. tračne enote (TRAX), ki so lahko bile samostojne ali pa vgrajene v računalnikih. Zlasti tiste večjih kapacitet so bile običajno samostojne (Slika 26). Za magnetne trakove je značilno, da je dostop do podatkov *zaporeden* – da pridemo do določenega podatka, moramo prebrati vse predhodne (kot pri audio kasetah). Lahko so shranili ogromne količine podatkov (ranga TB), za shranjevanje pa so uporabljali posebne kasete z več-steznim zapisom na magnetni trak. Gre za starejšo tehnologijo, ki so jo v dobršni meri nadomestili sodobnejši in bolj učinkoviti načini hrambe podatkov.

*Magnetni diski* nastopajo v dveh vrstah – trdi diski (angl. Hard Disc Drive, HDD, Slika 27) in gibki diski (angl. Floppy Disc Drive, FDD, Slika 28) in so zelo različnih kapacitet (od nekaj KB do nekaj TB).

Slika 25: *Magnetni trakovi*
Vir: commons.wikimedia.org, Carol M. Highsmith, 2016


Slika 26: *Tračna enota*
Vir: commons.wikimedia.org, Wernher-s, CC BY-SA, 2018

Slika 27: *Diskovni pogon*
Vir: Wikipedia, Evan-Amos, CC BY-SA, 2013

Značilnosti *gibkih diskov* (diskete, angl. Floppy Discs) so:

1. majhna kapaciteta (1.2 oz. 1.44MB – HD High Density 5.2" oz. 3.2" diskete)
2. nezanesljivi mediji (fizične poškodbe, razmagnetenje, temperaturno občutljivi)
3. za branje in pisanje na disketo je potrebna disketna enota (angl. Floppy Disk Drive, FDD)
4. počasnost pogonov FDD

Podobno kot pri magnetnih trakovih tudi tu gre za starejšo tehnologijo, ki so jo že nadomestili optični diski in spominske kartice. Zanimivi so zato, ker so bili tako po načinu dostopa kot formatu osnova za razvoj trdih in optičnih diskov.

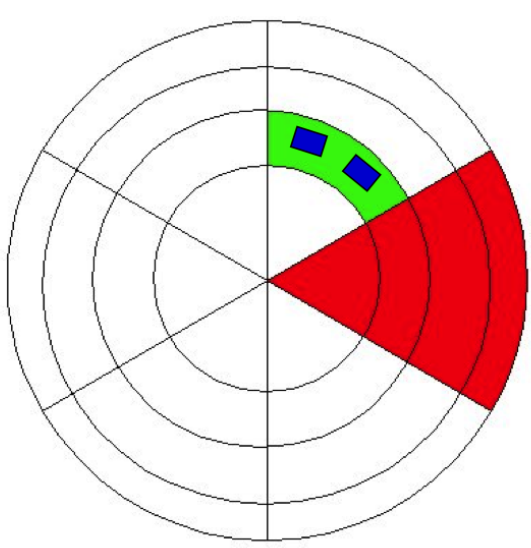
Slika 28: *Gibki diski*
Vir: Wikipedia, George Chernilevsky, CC BY-SA, 2009

*Trdi disk* (angl. Hard Disc Drive, HDD) je sestavljen iz več magnetno občutljivih kovinskih diskov na skupni osi, mehanizma za premikanje glav za branje in zapisovanje podatkov, krmilnega mehanizma in ohišja. Omogočajo naključen (RAM) in relativno hiter dostop do podatkov. Na njih lahko shranjujemo relativno velike količine podatkov – sodobni trdi diski imajo kapacitete od nekaj sto GB (tipična kapaciteta v standardni PC konfiguraciji je trenutno okoli 500 GB) do nekaj TB.

Tipični podatki, ki vplivajo na hitrost branja in zapisovanja na trdi disk, so:

1. hitrost vrtenja diskov (tipično 5400 obratov/min ali 7200 obratov/min)
2. vgrajen vmesni spomin – cache (tipično 8 ali 16 MB)
3. tip priključka na vodilo (IDE, SATA,...); na trgu prevladujejo SATA diski predvsem zaradi večje hitrosti pretoka podatkov

Podatki se v HDD podobno kot pri FDD shranjujejo po t.i. sektorjih in sledeh. Do podatkov dostopamo neposredno (angl. Direct Memory Access, DMA) – vsak podatek ima enolično lokacijo v okviru sektorja in sledi. Na sliki (Slika 29) je sled označena z zeleno barvo, sektor pa z rdečo.



Slika 29: *Zapis podatkov na enem cilindru diska, v sektorju, znotraj sledi*
Vir: lasten

*Optični diski* (angl. Optical Disc Drive) uporabljajo *optične medije za hrambo podatkov*. Za razliko od magnetnih diskov (FDD, HDD) tukaj laser odčitava podatke z reflektirajoče površine diska.

Za branje/zapisovanje podatkov na *zgoščenkah* (angl. Compact Disc, CD) potrebujemo ustrezen *optični pogon* (angl. Compact Disc Drive, CDD). Glede na izvedbo so zgoščenke lahko CD ROM (tudi CD-R ali Compact Disc Read Only Memory) ali CD-RW (Compact Disc ReWritable – t.i. ”CD-pekač”). CD-RW je optični disk z možnostjo pisanja in brisanja (zato ga včasih imenujejo tudi “Erasable”).

Optične medije uporabljamo tudi za hrambo multimedijskih vsebin (od tod tudi navedbe o kapaciteti CD medijev – min. glasbe v digitalni stereo tehniki). Tipična kapaciteta CD medijev je 650 oz. 700MB (74 oz. 80min glasbe). Z zapisi višje gostote je bila omogočena tudi hramba video posnetkov v digitalni kakovosti slike in zvoka ter višji ločljivosti, kot je ta bila običajna pri predhodnem analognem zapisu (npr. Video Home System, VHS). Bistvene

prednosti digitalnega video zapisa so: kvalitetnejša slika, digitalni zvok (tudi večkanalni), interaktivni meniji, možnost izbire jezika podnapisov in številne druge. V ta namen sta se uveljavila dva standarda za digitalne medije višje gostote zapisa, ki si s CD delita enako fizično dimenzijo:

1. DVD (Digital Video (Versatile) Disc) ima kapaciteto 4,7 GB (enostranski) ali 9.4 GB (dvostranski)
2. HD-DVD (high definition DVD) ima kapaciteto 15 oz. 30 GB, standardiziran pa je tudi že več-slojni 51 GB HD DVD (Toshiba)
3. BD (Blue-ray Disc) uporablja modri laser za razliko od CD in DVD, kjer uporabljajo rdečega, ima pa kapaciteto 25 GB oz. 50 GB

DVD pogoni razen DVD diskov lahko berejo vse vrste CD, DVD medijev, Blue Ray pogoni (angl. Blueray Drive, BD) pa razen naštetih še Blue Ray medije. DVD in BD RW pogoni (”pekači”) omogočajo tudi zapisovanje teh medijev.

Dodatni karakteristični podatki o optičnih pogonih se nanašajo predvsem na hitrost prenosa podatkov med pogonom in delovnim spominom:

1. 1x hitrost pri CD optičnih pogonih ustreza hitrosti pretoka podatkov 150 KB/s,
2. 1x hitrost pri DVD optičnih pogonih ustreza hitrosti pretoka podatkov 1,32 MB/s
3. 1x hitrost pri BD optičnih pogonih ustreza hitrosti pretoka podatkov 4,5 MB/s
4. vgrajen vmesni spomin – cache (tipično 2 MB)

Tipične vrednosti za branje so na primer za CD pogone 48x, za DVD pogone 16x, za BD pogone pa 10x, za zapisovanje pa za CD pogone 24x, za DVD pogone 8x in za BD pogone 2x.

*Spominske kartice* so pomnilne naprave različnih formatov za hrambo pretežno podatkov (npr. CF card, SD card, MS card), ki jih lahko vstavimo v vmesnik/režo digitalnega fotoaparata, video kamere, mobilnega telefona ali osebnega računalnika. Njihova glavna slabost je cena hrambe MB podatka, ki je višja kot pri trdih diskih, njihova glavna prednost pa je, da so zanesljivejše, bolj prenosljive, in enostavne za uporabo. Tehnologijo smo že razložili pri FLASH pomnilnikih, nekaj tipičnih predstavnikov pa prikazuje Slika 24.

Tipični predstavnik spominskih kartic je t.i. USB ključ (angl. USB Memory Stick ali USB flash drive), ki ima naslednje lastnosti:

1. vsebuje FLASH pomnilnik – podatke lahko hranimo in beremo, brisanje pa je nepovratna akcija
2. tipične kapacitete: 8GB, 16GB, 32GB, 64GB, 128GB s tendenco naraščanja ob hkratnem nižanju cen
3. sodobni standard za prenos podatkov – USB 3.0 (teoretična hitrost prenosa podatkov je 5 Gbit/s ali 625 MB/s, kar je več kot 10 krat hitreje kot pri USB 2.0, ki ima teoretično hitrost prenosa podatkov 480Mb/s, kar je bistveno hitreje od osnovnega standarda USB 1.1 s hitrostjo prenosa 12Mb/s)

Aktualno se uveljavljajo *polprevodniški diski* (angl. Solid State Disc, SDD), ki so po funkcionalnosti enaki kot HDD. Njihova glavna prednost je, da so po izvedbi bolj podobni spominskim karticam, saj nimajo gibljivih delov. Njihova glavna slabost je, da še ne dosegajo kapacitet HDD. Kljub temu s tipskimi kapacitetami 256 oz. 512GB že lahko zadovoljijo potrebe zlasti mobilnih uporabnikov po hrambi podatkov in programov. Mobilnega uporabnika omenjamo, ker imajo SDD tudi dosti nižjo potrošnjo energije kot HDD in je zato avtonomija delovanja mobilnih računalnikov na baterijskem napajanju boljša. Po drugi strani se SDD pogosto uporabljajo v kombinaciji s HDD v zmogljivejših računalniških konfiguracijah, ki lahko zaradi hitrejšega dostopa do bliskovnega pomnilnika omogočijo hitrejše nalaganje in izvajanje programov ob hkratni nezmanjšani kapaciteti hrambe podatkov.

### 3.4.3 Podatkovna shramba organizacije

Zaradi padca cen in večanja kapacitet predvsem sekundarnih pomnilnikov, si organizacije lahko omislijo lastni center za varno hrambo podatkov. *Podatkovna shramba organizacije* (*Enterprise Storage System, ESS*) je neodvisen zunanji sekundarni pomnilnik, ki ga tvorita dve ali več pomnilnih komponent z inteligentnim sistemom za nadzor.

Značilnosti podatkovne shrambe organizacije so:

- velika kapaciteta hrambe podatkov
- velika hitrost prenosa in dostopa do podatkov
- visoka stopnja razpoložljivosti
- sofisticiran sistem upravljanja podatkov

Osnovni tipi podatkovnih shramb organizacije so:

- Redundantno polje neodvisnih diskov (angl. Redudant Array of Independent Disks, RAID)
- Omrežni pomnilnik (angl. Network Attached Storage, NAS)
- Omrežje za shranjevanje podatkov (angl. Storage Area Network, SAN)

*Redundantno polje neodvisnih diskov (Redudant Array of Independent Disks, RAID)* je sistem hrambe podatkov, ki povezuje več standardnih trdih diskov preko skupnega mikrokrmilnika. Ta koordinira delo diskov, tako da se le-ti navzven obnašajo kot en-sam, podatki pa se istočasno shranjujejo na dva ali več diskov. Obstaja več različic RAID, vse pa ščitijo organizacijo pred izgubo dragocenih podatkov.

*Omrežni pomnilnik (Network-Attached Storage, NAS)* predstavlja namenski strežnik za hrambo podatkov. Uporabniki do njih dostopajo preko omrežja, za kar se morajo (npr. preko svojega uporabniškega imena in gesla) prijaviti na strežnik. Le-ta jim nato preko njihovega profila omogoči dostop do lastnih in skupnih podatkov. Ker ta strežnik deluje kot ostali splošno namenski strežniki (splet, e-pošta, ipd.), sta njegova implementacija in vzdrževanje relativno enostavni. Ni potrebe po dodatnem izobraževanju osebja ali nakupu kakšne dodatne programske opreme za administracijo podatkov, saj za nadzor pravic in dodeljevanje dostopa administratorji uporabljajo iste mehanizme kot sicer. Seveda je zaradi varnosti podatkov v takšnem strežniku koristno implementirati tudi RAID.

*Omrežje za shranjevanje podatkov (Storage Area Network, SAN)* je podatkovna shramba organizacije, ki je zasnovana na posebnih arhitekturah namenskih mrež za hitro in zanesljivo hrambo podatkov ter porazdeljen dostop preko več strežnikov. Gre za t.i. *storage over IP* tehnologijo prenosa/hrambe podatkov med odjemalci in strežniki, zasnovano na *internetnem protokolu (IP)*. Implementacija zahteva posebno *programsko opremo za vzdrževanje pomnilnika (angl. Storage vizualization software)*, ki grafično prikazuje celotno podatkovno mrežo, kar administratorjem omogoča, da spremljajo stanje in nadzirajo vse naprave v SAN omrežju preko ustrezne konzole. Ta sistem v nasprotju z RAID in NAS zahteva kompleksno administracijo.

Podrobna obravnava omenjenih podatkovnih shramb organizacij presega okvir pričujočega predmeta. Zato so tu navedene samo njihove glavne značilnosti, zainteresirani bralec pa lahko več o tem prebere na spletu.

## 3.5 Periferne naprave

***Periferne naprave*** *so vse naprave, ki jih na CPE povezujemo preko vhodno/izhodnih vmesnikov.* Gre za vse priključene naprave, ki služijo za vnos/izpis in hrambo podatkov. To so na primer:

1. vhodne naprave za vnos informacij v računalnik (tipkovnica, miška, mikrofon, kamera, skener, razni senzorji, ipd.) in

2. izhodne naprave omogočajo posredovanje rezultatov delovanja računalnika v okolje (zaslon, tiskalnik, projektor, zvočniki, razni aktuatorji, ipd.).

Poenostavljeno kot periferne pogosto označujemo vse naprave, ki so priključene na računalnik. Glede na razne vrste računalnikov (od centralnih do osebnih), zlasti pa od pojavitve majhnih prenosnih računalnikov, ta definicija več ne zdrži, saj so periferne naprave zlasti pri prenosnih računalnikih lahko tudi integrirane v ohišje računalnika. Zato raje govorimo o dveh vrstah perifernih naprav glede na način priključitve na vodilo CPE: *notranjih* (internih) in *zunanj*ih (eksternih).

Periferne naprave lahko priključimo na CPE preko enega izmed predvidenih vmesnikov, ki smo jih že spoznali pri obravnavi matične plošče:

1. *rež* (sleng: "slot") – predvidene predvsem za priključitev *notranjih perifernih naprav* (grafične kartice, trdega diska, glasbene kartice, TV kartice, ipd.)

2. *vrat* (sleng: "port") – predvidene predvsem za priključitev *zunanjih perifernih naprav* (monitorja, tiskalnika, miške, optičnega čitalnika, tipkovnice, ipd.)

3. *(brez-)žičnih* (angl. wireless) vmesnikov za priključitev *zunanjih perifernih naprav* (npr. infra rdeči (IR), bluetooth (BT), omrežni (WiFi), RFID, ipd.)

Opis vseh zelo raznolikih perifernih naprav presega okvir tega učnega predmeta. Zaradi pomembnosti tematike za področje logistike, se bomo v poglavju o računalniški podatkovni opremi (Poglavje 6), kjer gre za zajem, hrambo,

obdelavo in prenos digitaliziranih informacij, ukvarjali tudi s perifernimi napravami in tehnologijami za pretvorbo zvoka in slike iz analogne v digitalno obliko in obratno. Ker gre pri omenjeni pretvorbi za zadnji stik analognega sveta pred digitalno obdelavo in posredovanjem podatkov, je kakovost te pretvorbe – analogno-digitalna (A/D) pretvorba – kritičnega pomena za kakovosten zajem in obdelavo podatkov. Prav tako pa seveda velja, da je pomembna tudi pravočasna in dovolj kakovostna pretvorba v obratni smeri – digitalno-analogna (D/A) pretvorba – za prenos rezultatov obdelave nazaj v analogni svet (npr. ob poslušanju glasbe z optičnih nosilcev podatkov, krmiljenju aktuatorjev v industrijskem okolju, ipd.).

## 3.6 Računalniške platforme

Splošno priznanih je pet generacij računalnikov glede na kronologijo njihovega nastanka in namembnost (Tabela 1). Vsako novo generacijo karakterizira povečanje procesorske moči, spominskih kapacitet in zanesljivosti ob nižji ceni hrambe in procesiranja podatkov.

Zmogljivost računalnikov je tesno povezana z uporabo v določenem okolju oz. področju uporabe. Sodobne računalnike delimo po zmogljivosti v šest osnovnih kategorij ali *platform*:

1. Super računalniki (angl. Super Computers): zelo zmogljivi specializirani računalniki
2. Centralni računalniki (angl. Mainframe Computers): splošno namenski centralni korporativni računalniki
3. Midi/mini-računalniki (angl. Midrange Computers): splošno namenski centralni računalniki za majhna in srednja podjetja
4. Delovne postaje (angl. Workstations): splošno namenski zmogljivi namizni računalniki za analitike, načrtovalce in inženirje

5. Notesniki in namizni računalniki (angl. Notebooks and Desktop Computers): splošno namenski računalniki za vsakodnevno poslovno ali domačo rabo

6. Naprave z vgrajenimi računalniki (angl. Appliance): industrijski stroji, vozila, gospodinjski aparati, ipd.

Za računsko intenzivne obdelave v zelo specifičnih področjih uporabe so bili razviti t.i. *super računalniki*. Gre za namenske arhitekture računalnikov, ki so prirejene za *obdelave masovnih podatkov* (angl. high-performance computing, HPC), kot so vremenski prognostični programi, analize in prognoze potresov, raziskave vesolja, ipd. Ti računalniki se pogosto že v arhitekturi uporabljenih CPE, običajno pa tudi v ustroju njihovih komponent, razlikujejo od ostalih računalniških platform. Gre za maloserijske specializirane izvedbe pomnilniških in/ali procesnih komponent, ki so optimizirane za ciljne aplikacije.

*Centralni računalniki* so bili razviti, da bi s svojo procesno močjo zadostili potrebam večuporabniških in večprocesnih okolij. Pogosto gre tukaj za razpršene lokacije podružnic poslovnih sistemov, ki pa uporabljajo skupne računalniške kapacitete (npr. banke, borze, ipd.). Po namenu in strukturi enaki, le nekoliko manj zmogljivi so t.i. midi/mini-računalniki. V sodobnem času so jih pretežno nadomestili zmogljivi namizni računalniki, uporabljeni kot strežniki. Strežniki primanjkljaje v procesni moči kompenzirajo z mreženjem in porazdelitvijo procesnega bremena med več povezanimi računalniki, večje potrebe po hrambi podatkov pa s sistemi za hrambo podatkov, o katerih smo že govorili.

Prav tako so se zmogljivi namizni računalniki z bolj zmogljivimi procesorji, pomnilniki in multi-medijskimi komponentami uveljavili kot *delovne postaje*. Le-te pokrivajo cel spekter aplikacij – od zahtevnih poslovnih analiz, načrtovanja novih proizvodov, do poslovodenja manjših podjetij. Težko je potegniti strogo ločnico med delovnimi postajami in strežniki, saj so po zmogljivosti lahko zelo podobni, vendar so delovne postaje pretežno namenjene osebni, strežniki pa skupinski rabi.

*Notesnikov in namiznih računalnikov* najrazličnejših formatov (prenosniki, osebni računalniki, tablice, pametni telefoni,...) ni potrebno posebej predstavljati, saj imajo enak ustroj, čeprav se po procesni moči lahko precej razlikujejo. Gre za računalnike, ki so namenjeni vsakodnevni osebni in pogosto tudi mobilni rabi.

Končno, dandanes skoraj več ni elektronske naprave, ki v sebi ne bi skrivala vsaj enega *vgrajenega namenskega računalnika*, ki nadomešča diskretno vezje z elektromehanskimi stikali (npr. pralni, pomivalni stroj, ipd.). Običajno jih poganjajo preprostejši namenski mikroprocesorji (t.i. mikrokrmilniki), ki svojo nalogo učinkovito opravljajo ob minimalnem številu komponent, stroških vgradnje in porabi energije. Z njihovim mreženjem dosegamo učinek *interneta stvari* (angl. Internet of Things, IoT), preko katerega tudi lažje upravljamo življenjski cikel teh naprav.

### 3.6.1 Izbira računalniške platforme

Pri izbiri najprimernejše računalniške platforme za našo rabo (aplikacijo) se običajno ravnamo po naslednjih vprašanjih:

- Čemu bo računalnik namenjen?
- Kakšne komponente potrebujemo?
- Kakšen operacijski sistem (OS) bomo uporabljali?
- Kakšno programsko opremo potrebujemo?
- Katere periferne enote potrebujemo?
- Cena?

### 3.6.2 Napredne računalniške platforme

Za zaključek bi želel omeniti še nekaj naprednih računalniških platform, ki so si že utrle svojo pot tako v osebni kot poslovni rabi, vendar (še) niso prevladujoče. Zaznamovale bodo bodoče generacije računalnikov in na njih temelječih informacijskih sistemov.

*Uporabniško računalništvo* (angl. *Utility computing* ali *Subscription computing* ali *On-demand computing*) predstavlja ponudbo računalniške opreme in storitev strankam. Glede na to, da nakup in vzdrževanje lastne računalniške opreme zahteva precejšnjo investicijo, se lahko podjetja odločijo tudi, da potrebne računalniške kapacitete najamejo. Ponudniki storitev uporabniškega računalništva glede na potrebe strank zagotavljajo računalniške vire svojim uporabnikom. Običajno *ponudnik aplikacijskih storitev (angl. Application Service Provider, ASP)* zagotavlja svojim naročnikom (uporabnikom) dostop do svojih strežnikov ter programov in podatkov na njih preko spletnih ali programskih odjemalcev.

*Mrežno računalništvo* (angl. *Grid computing)* združuje več v mrežo povezanih računalnikov, ki obdelujejo isto nalogo. Problemi, ki jih rešujejo, so večinoma znanstvene narave in zahtevajo veliko računskih ciklov ter obdelavo velike količine podatkov. Začetnik je bil projekt SETI@home (Search for Extraterrestrial Intelligence). Gre za množični (angl. crowdsource) znanstveni projekt, namenjen iskanju nezemeljske inteligence, v katerega se je lahko vključil kdorkoli z omreženim računalnikom, preko brezplačnega aplikacijskega programa za analizo radio-teleskopskih podatkov. Iz mrežnega računalništva se je razvilo *računalništvo v oblaku (angl. Cloud Computing)*. V mreži povezanih računalnikov se nahajajo vsi podatki in programi za njihovo obdelavo. Pri tem ni pomembno, kje se le-ti fizično nahajajo, običajno pa so replicirani med več strežniki zaradi delitve bremena.

*Nano-tehnologija* (angl. *Nanotechnology*) se nanaša na materiale, naprave in sisteme velikosti 1–100 nm. Te platforme so v eksperimentalni fazi. V povezavi s *kvantnim računalništvom (angl. quantum-computing)* bi lahko revoluci-

onarno spremenile naš odnos do računalnikov – *nosljiv računalnik (angl. wearable computer)*. Izdelali so že prve nano-tranzistorje in pomnilne celice, ki lahko shranijo večvalentne *kvantne bite (qbit)* informacije, vendar so raziskave še na ravni prototipov. Vsekakor ta tehnologija nakazuje možen trend nadaljnje miniaturizacije, novih platform in potencialni skokovit razvoj informacijske tehnologije v smislu Moorove teorije, ki bi omogočal ne-slutene zmogljivosti kvantnih računalnikov na miniaturnih nano-vezjih.

# 4 Računalniška komunikacijska oprema (NW)

V tem poglavju bomo širše obravnavali *računalniško komunikacijsko opremo (angl. netware)*. V ta namen bomo najprej razložili pomen komunikacije in telekomunikacijskih mrež, uvedli klasifikacijo ter identificirali komponente telekomunikacijskih omrežij. Sledila bo podrobnejša predstavitev osnov računalniških omrežij z opisi topologij, komponent in protokolov, ki so v rabi v računalniških omrežjih.

## 4.1 Telekomunikacijska omrežja

Komunikacijske mreže lahko klasificiramo glede na različne kriterije: vrsto informacij, prostranost/obseg, namen, vrsto komunikacijskega medija, itd. Glede na vrsto informacij jih delimo na mreže za govorno in mreže za podatkovno komunikacijo, pri čemer v sodobnih mrežah težimo k integraciji obeh. Po prostranosti mreže delimo na *lokalne (angl. Local Area Networks, LAN)*, ki so omejene na eno stavbo ali njen del in *prostrane (angl. Wide Area Networks, WAN)*, ki pokrivajo večja zemljepisna področja. Mreže po namenu delimo na *privatne in javne*. V privatnih imajo dostop le omejene skupine uporabnikov ali posamezniki. Glede na uporabnike privatne mreže delimo na *akademske* (študenti, raziskovalci,...), *poslovne* (npr. bančne), *mreže sistemov* (npr. nadzor distribucije električne energije, ipd.). Javne mreže so dostopne vsem uporabnikom pod določenimi pogoji, ki so predmet sporazuma med uporabniki in mrežnim operaterjem (npr. javne WLAN dostopne točke). Glede na vrsto medija mreže v splošnem delimo na *žične (angl. wired)*, kjer prenos informacij poteka po žici (telefonska parica, koaksialni kabli, UTP ali optični kabli) in *brezžične (angl. wireless, WiFi)*, kjer prenos informacij poteka po etru (radijski valovi/frekvence).

Sedaj smo iz večine relevantnih vidikov osvetlili komunikacijske mreže in že lahko postavimo definicijo komunikacije:

***Komunikacija*** *je izmenjava podatkov med najmanj dvema udeležencema.*

Osnovna načina izmenjave podatkov sta:

1. *komutacija kanalov*, ki je primerna za kontinuiran prenos informacij oz. podatkov v realnem času; npr. govor (telefonsko omrežje, UKW radio, ...) in

2. *komutacija paketov*, ki nam omogoča boljšo izkoriščenost kapacitete komunikacijskega kanala in manj motenj pri komunikaciji, saj so podatki zakodirani digitalno, kanal pa je zaseden le v času prehoda paketov podatkov (npr. internetni radio).

***Telekomunikacijska mreža*** *je katerikoli sistem, ki prenaša sporočila od pošiljatelja k prejemniku.*

Nasploh lahko trdimo, da so osnovni členi telekomunikacijskih mrež *telekomunikacijski terminali*. Sem prištevamo zraven telefonov, faksov in drugih namenskih komunikacijskih naprav tudi računalnike vseh tipov in velikosti, ki mrežo uporabljajo za medsebojno komunikacijo. Ustrezne definicije relevantnih komponent telekomunikacijskih mrež so:

***Telekomunikacijski terminali*** *so katerakoli vhodno/izhodna naprava, ki uporablja mrežo za pošiljanje/sprejem podatkov.*

***Telekomunikacijski procesorji*** *so naprave, ki podpirajo sprejem in pošiljanje podatkov.*

***Telekomunikacijski kanal*** *je medij preko katerega se prenašajo podatki.*

***Telekomunikacijska upravljavska programska oprema*** *so programi, ki upravljajo in nadzirajo vse aktivnosti v telekomunikacijski mreži.*

***Telekomunikacijsko omrežje*** *predstavlja aparaturno-programsko celoto, v kateri se podatki prenašajo po komunikacijskih kanalih od enega do drugega vozlišča mreže.*

Vsak udeleženec komunikacije po mreži predstavlja *vozlišče omrežja (angl. network node)*. Vsi podatki, ki se prenašajo od enega do drugega vozlišča potujejo preko *telekomunikacijskega kanala*. Lastnosti tega kanala določajo pasovno širino in hitrost prenosa podatkov – *prepustnost kanala.*

Primeri sodobnih telekomunikacijskih mrež so:

1. *Fiksna telefonska javna mreža (angl. Public Switched Telephone Network, PSTN)*, kjer se je komunikacija prvotno odvijala analogno s komutacijo kanalov po *telefonski parici (angl. twisted pair)*, z digitalizacijo telefonskih povezav in paketnim prenosom podatkov pa so dosegli večjo hitrost in kakovost prenosa (zlasti po kabelskih in optičnih povezavah), omogočen pa je bil tudi integriran prenos govora, videa in podatkov brez dodatnih posegov v mrežo;

2. *Globalni sistem mobilnih komunikacij (angl. Global System for Mobile communication, GSM)* uporablja paketni prenos podatkov, kjer posamezna področja uporabe pokrivajo t.i. celice po zgledu *brezžičnih računalniških mrež (WiFi)*, radijski signal pa je zagotovljen na določenih (frekvenčnih) področjih; razen govorne komunikacije omogoča tudi izmenjavo kratkih (multimedijskih) sporočil (SMS – Short Message Service, MMS – Multimedia Messaging System) in brezžičen dostop do Interneta;

3. *Iridium mreža satelitskih komunikacij* uporablja konstelacijo komunikacijskih satelitov, ki krožijo okoli zemlje na višini okoli 780 km (za razliko od večine preostalih komunikacijskih satelitov, ki krožijo v geostacionarni orbiti na okoli 35000 km). Glavna prednost satelitske mreže je, da zagotavlja pokritost z omrežjem in dostopnost do Interneta, kjer ta sicer ne bi bil na voljo (v divjini, na oceanih ter na severnem/južnem tečaju).

4. *Internet* – o njem malo več kasneje;

Generacije sistema mobilnih komunikacij (GSM):

**1G** NMT – prvi analogni sistemi mobilnih komunikacij

**2G** GSM – prvi digitalni sistemi mobilnih komunikacij (900/1800 MHz, SMS/MMS)

**2.5G** GPRS (General Packet Radio Service) – dostop do Interneta preko WAP protokola (do 140kb/s)

**3G** UMTS (Universal Mobile Telecommunication System) – širokopasovni dostop do Interneta (1,9-2,1 GHz, do 21,6 Mb/s)

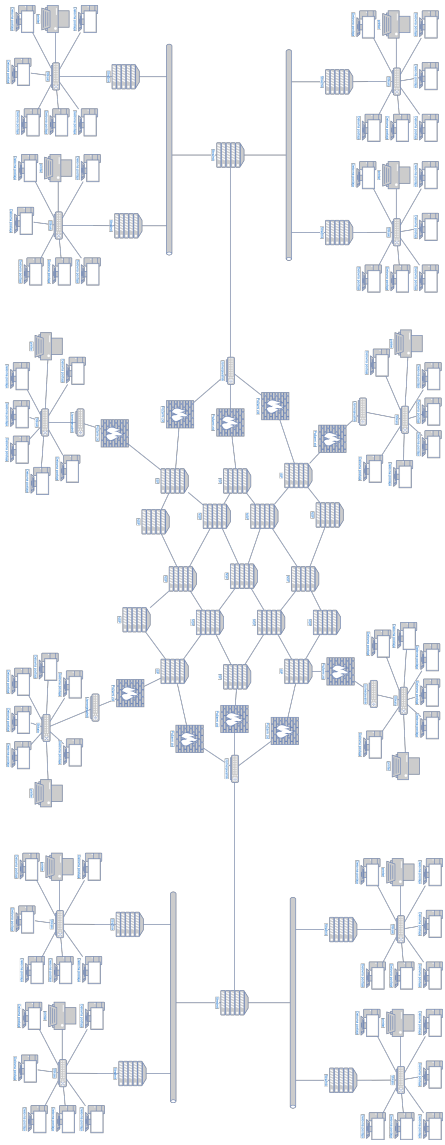
**4G** LTE (Long Term Evolution) in WiMAX (Worldwide Interoperability for Microwave Access) – širokopasnovni dostop do Interneta (1,9 – 3,7 GHz, do 299.6 Mbit/s)

**5G** 5G brezžično omrežje najnovejše generacije – širokopasnovni dostop do Interneta (3.4-3.8 GHz, 26-27.5 GHz; 10-100x hitrejše od 4G)

Začetki GSM sistema segajo v 1980a leta (1G), razmah pa so dosegle v 1990h (2G). Mobilni dostop do Interneta je bil mogoč nekje na prelomu tisočletja (2.5G), ko je bila vzpostavljena tudi satelitska mreža mobilnih komunikacij (1998). Odtlej spremljamo razvoj novih generacij mobilnih komunikacij približno vsakih 5-10 let. V teku je povezovanje fiksnih in mobilnih telekomunikacijskih mrež, satelitska mreža pa je tudi nemoteno doživela zadnjo posodobitev leta 2019.

## 4.2 Računalniška omrežja

Za telekomunikacijske mreže skrbijo *ponudniki telekomunikacijskih storitev (angl. Network Service Provider, NSP)*, ki v sodelovanju s *ponudniki internetnih storitev (angl. Internet Service Provider, ISP)* omogočajo globalno komunikacijo po Internetu kot največjem prostranem računalniškem omrežju (Slika 30). Pa poglejmo, kaj je in iz česa je sestavljeno računalniško omrežje.



Slika 30: *Povezovanje lokalnih mrež v prostrane*
Vir: lasten

***Računalniška mreža*** *predstavlja skupino računalnikov, ki so med sabo povezani preko komunikacijskega medija in so lahko poljubno oddaljeni.*

Sedaj, ko vemo, kaj sta telekomunikacijska in računalniška mreža, je čas da postavimo definicijo računalniškega omrežja:

***Računalniško omrežje*** *je skupek medsebojno povezanih računalnikov, ki komunicirajo po enem ali več protokolih prenosa podatkov.*

***Telekomunikacijski protokol*** *je nabor pravil in dogovorov o komunikaciji med napravami.*

Osnovni razlogi za povezovanje računalnikov v omrežja so:

1. enostaven dostop do deljenih skupnih vsebin (podatkov, servisov)
2. podatke med uporabniki enostavno in hitro izmenjujemo (delitev dostopa ali pretakanje po omrežju)
3. delitev opreme med več uporabniki (npr. strežnik, barvni laserski tiskalnik ali risalnik, ipd.)
4. porazdeljena obdelava podatkov (delitev procesnega bremena)
5. zabava (npr. mrežne igre, video konference, ipd.)

### 4.2.1 Mrežna oprema

Mrežna oprema, ki jo uporabljamo v računalniških omrežjih sestavljajo naslednji tipi naprav:

1. Modem
2. Replikator (hub)
3. Stikalo (switch)

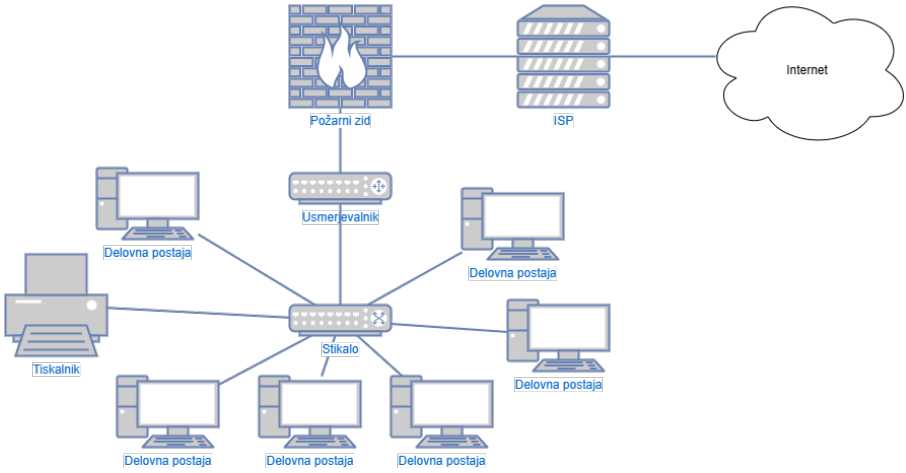
4. Usmerjevalnik (router) - prehod (gateway)

*Modem* (angl. *MOdulator-DEMmodulator*) je naprava za pretvorbo podatkov v obliko, primerno za prenos po telekomunikacijskih linijah. Modem pri pošiljanju podatkov modulira digitalni signal iz računalnika v analogni signal in obratno pri sprejemu. Glede na vrsto prenosnega medija poznamo več vrst modemov: telefonske, kabelske in GSM.

Podobno kot pri razdelilniku napajanja električnih naprav se je tudi pri omrežnih napravah pojavila potreba po delitvi oz. razširitvi (obsega) omrežne povezave. *Replikator (angl. hub)* predstavlja spoj vseh mrežnih kablov, ki se vežejo nanj in tvori skupen prenosni medij za podatkovne pakete povezanih računalnikov. Pri tem seveda lahko pride tudi do trkov in s tem izgubo podatkovnih paketov, ki jih replikator sam ne more razrešiti. Enako vlogo imajo stikala, ki so "pametnejša" od replikatorjev in bolje prenašajo obremenitve prometa.

*Stikalo (angl. switch)* interno preklaplja linije med povezanimi računalniki (podobno kot arbitraža vodila) in na ta način navidezno ustvari neodvisne linije preko istega prenosnega medija. Stikalo s tem zmanjša število *trkov (angl. collisions)* in potrebo po ponovnem prenosu podatkovnih paketov po Ethernet povezavi.

*Usmerjevalnik (angl. router)* uporabimo, ko želimo povezati mreže različnih karakteristik (npr. različnih protokolov, različnih hitrosti, ipd.). Preko usmerjevalnika lahko LAN priključimo na Internet. Usmerjevalniki imajo v telekomunikacijskih omrežjih funkcijo *prehodov* (angl. *gateway*). Prehodi vsebujejo naprave, kot so prevajalniki protokolov, naprave za usklajevanje upornosti, pretvorniki hitrosti prenosa, izolatorji napak, pretvorniki signalov za zagotavljanje interoperabilnosti komunikacijskih sistemov. Prehod z izvedbo konverzij protokolov izvaja prevajanje/usklajevanje protokolov povezanih heterogenih omrežij. Vloga prehoda zajema vzpostavitev vzajemno sprejemljivih administrativnih procedur med obema omrežjema. Sodobni usmerjevalniki služijo tudi kot *požarni zidovi (angl. firewall)*, ki ščitijo omrežje pred nezaželenimi vhodnimi podatki.



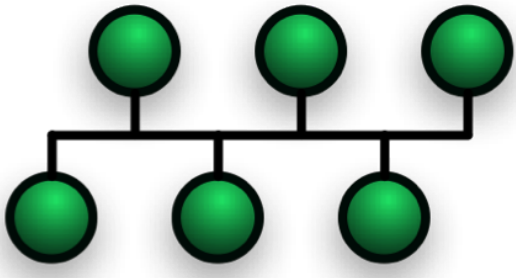

Slika 31: *Mrežne naprave v lokalni računalniški mreži*
Vir: lasten

Slika 31 prikazuje tipično arhitekturo mrežnih naprav in terminalov v lokalnem računalniškem omrežju.

### 4.2.2 Mrežne topologije

S. A. Tanenbaum je leta 1988 postavil temelje načrtovanja računalniških mrež (topologije) na visoko strukturiran način – večinoma hierarhično. Prostranost, vrsta informacij in namen se od mreže do mreže razlikujejo. Tukaj predstavljamo nekaj arhetipov mrežnih topologij.

Standard za povezovanje računalniških omrežij je *Ethernet*. Ta omogoča, da si isti prenosni medij delita dve ali več vozlišč v omrežju. Sestavljajo ga komunikacijski protokoli in mrežne topologije.

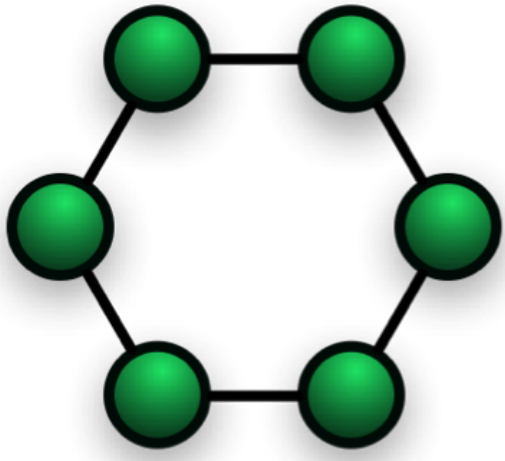


Slika 32: *Vodilo (bus)*
Vir: Wikipedia, 2006

Topologija *vodilo (angl. bus)* (Slika 32) predstavlja vozlišča, povezana linearno po sistemu računalniškega vodila. Glavna prednost te topologije je enostaven priklop in manj ”kabliranja” (polaganja povezovalnih kablov), glavna pomanjkljivost pa, da prekinitev mreže na kateri koli točki pomeni prenehanje delovanja celotne mreže.

V topologiji *prstan (angl. ring)* (Slika 33) so vozlišča povezana tako, da ima vsako vozlišče v mreži natanko dva soseda. Paket informacij kroži po omrežju dokler ne doseže ciljnega vozlišča. V tem času morajo biti vsa ostala vozlišča “tiho” (ne morejo oddajati zaradi zasedenosti mreže). Je manj občutljiva od topologije vodila, saj njena prekinitev pomeni le nedostopnost vozlišč od točke prekinitve naprej.

Zaradi očitnih pomanjkljivosti topologij vodilo in prstan se je uveljavila topologija zvezda. Vozlišča topologije *zvezda (angl. star)* (Slika 34) za priklop v omrežje uporabljajo *koncentrator (hub, switch ali router)*. Njena šibka točka so prav koncentratorji, preko katerih vozlišča mreže komunicirajo med sabo



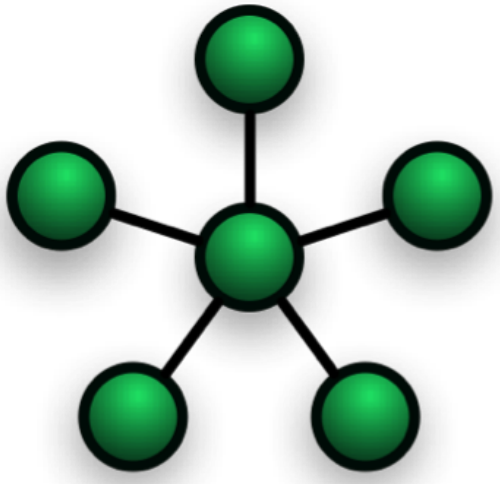
Slika 33: *Prstan (ring)*
Vir: de.wikipedia.org, Foobaz, Parzi, Predatorix, CC-BY-SA, 2023

in z oddaljenimi vozlišči, njena glavna prednost pa, da odpoved posameznega koncentratorja pomeni samo nedostopnost nanj povezanih vozlišč, medtem ko preostala mreža normalno deluje.

### Mrežne arhitekture

Pri računalniških omrežjih običajno govorimo o lokalnih (LAN) omrežjih, saj prostrana (WAN) omrežja pravzaprav predstavljajo samo več med sabo povezanih lokalnih omrežij. *Internet* je največje svetovno računalniško omrežje, ki združuje številna tehnološko heterogena lokalna omrežja v enoten *medmrežni* prostor zato ga pogosto imenujemo tudi *Medmrežje*.

Glede na vlogo vozlišč v omrežjih ločimo dve logični arhitekturi računalniških omrežij, t.i.:



Slika 34: *Zvezda (star)*
Vir: de.wikipedia.org, Foobaz, Parzi, Predatorix, CC-BY-SA, 2023

1. *Odjemalec-strežnik (angl. client-server) omrežja*, v katerih je običajno več odjemalcev povezanih na nek strežnik, lahko pa je tudi več strežnikov in

2. *Vsak-z-vsakim (angl. peer-to-peer oz. P2P) omrežja,* v katerih so vsi računalniki enakopravni – lahko so hkrati odjemalci in strežniki.

Arhitektura odjemalec-strežnik je zasnovana na principu – postrežen je le tisti odjemalec, ki poda zahtevo. Princip delovanja strežnika je analogen delovanju npr. klicnega centra – uporabniki postavljajo vprašanja in prejmejo informacije. Računalnik, ki streže zahteve imenujemo *strežnik* (*angl. server*). To je računalnik, ki daje na voljo in posreduje podatke, servise in naprave registriranim *odjemalcem*. Hkrati ima odgovornost nadzora nad varnostjo podatkov in lahko nadzira aktivnosti odjemalcev. Računalnik, ki prejema podatke od strežnika imenujemo *odjemalec (angl. client*). Le-ta nam omogoča dostop do podatkov in servisov, ki jih nudi strežnik.

V LAN obstaja več tipov namenskih strežnikov, ki pa s kombiniranjem njihovih servisov lahko postanejo tudi več-namenski:

1. avtorizacijski (avtentikacija in avtorizacija uporabnikov)

2. datotečni (deljenje podatkov)

3. E–mail (elektronska pošta)

4. Web (spletni strežnik)

5. Backup (varnostna kopija)...

P2P omrežja pogosto srečamo, kjer odjemalci želijo deliti določene vsebine med sabo preko Interneta (npr. eMule, BearShare, Shareaza, LimeWire, itd.), v uporabi pa so tudi pri internetni video-telefoniji (npr. Zoom, Viber, ipd.). Odjemalci lahko med sabo izmenjujejo podatke tudi s posredovanjem strežnika (npr. Arnes Filesender, MS Sharepoint, MS Teams, Skype, ipd.). Takšna izmenjava je praviloma nekoliko počasnejša, je pa varnejša.

### 4.2.3 Brezžična omrežja

V sodobnem svetu je vedno več mrežnih povezav brezžičnih. Sodobna omrežja so konglomerati žičnih LAN in WAN ter brezžičnih lokalnih WLAN in prostranih WWAN mrež. Popularni naziv za standarde in tehnologije brezžičnih lokalnih mrež je *WiFi* (*angl. Wireless Fidelity*).

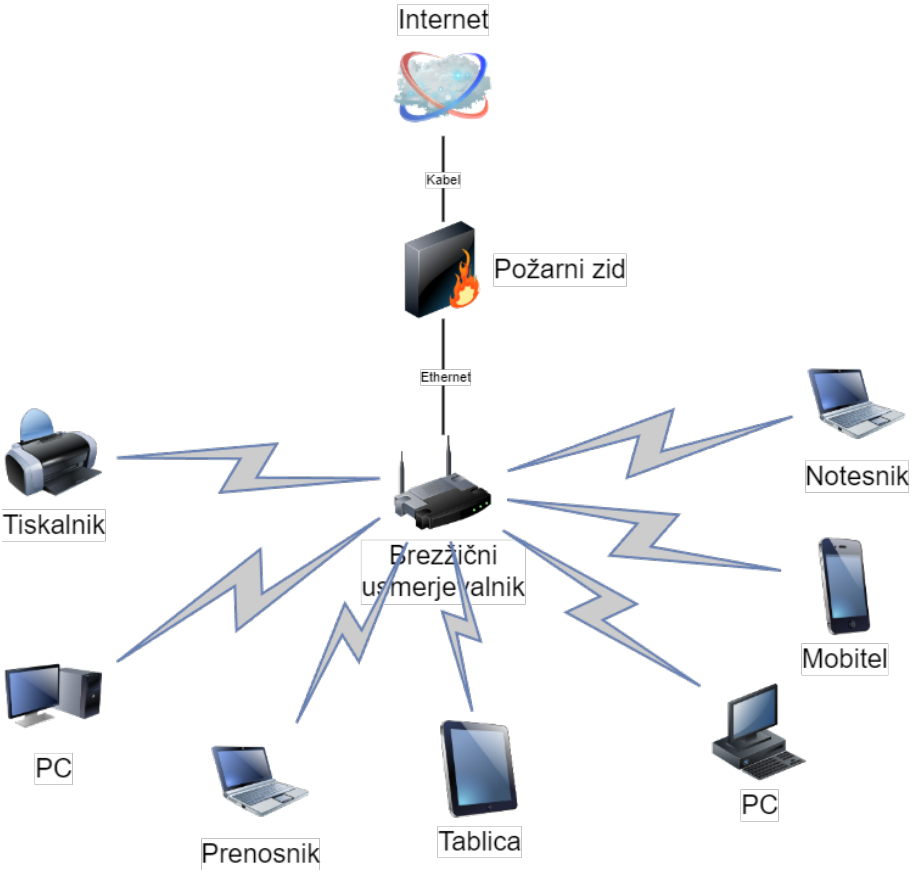
WLAN delujejo na podobnem principu kot GSM omrežja – pokrivajo določen prostor po načelu celic. Navzven jih predstavljajo *dostopne točke (angl. Access Point, AP)*, ki jih priključujemo preko koncentratorjev na žična LAN ali brezžična GSM omrežja.

Ključni element dostopne točke je radijski sprejemnik/oddajnik, ki omogoča dvosmerni pretok podatkov. Dostopne točke so pogosto izvedene kot *usmerjevalniki* (angl. *router*), ki vsebujejo tudi potrebno logiko za prenos podatkov med brezžičnim in ožičenim delom mreže. Da bi računalniki lahko komunicirali preko dostopnih točk, potrebujejo brezžične mrežne kartice (z ustreznimi gonilniki).

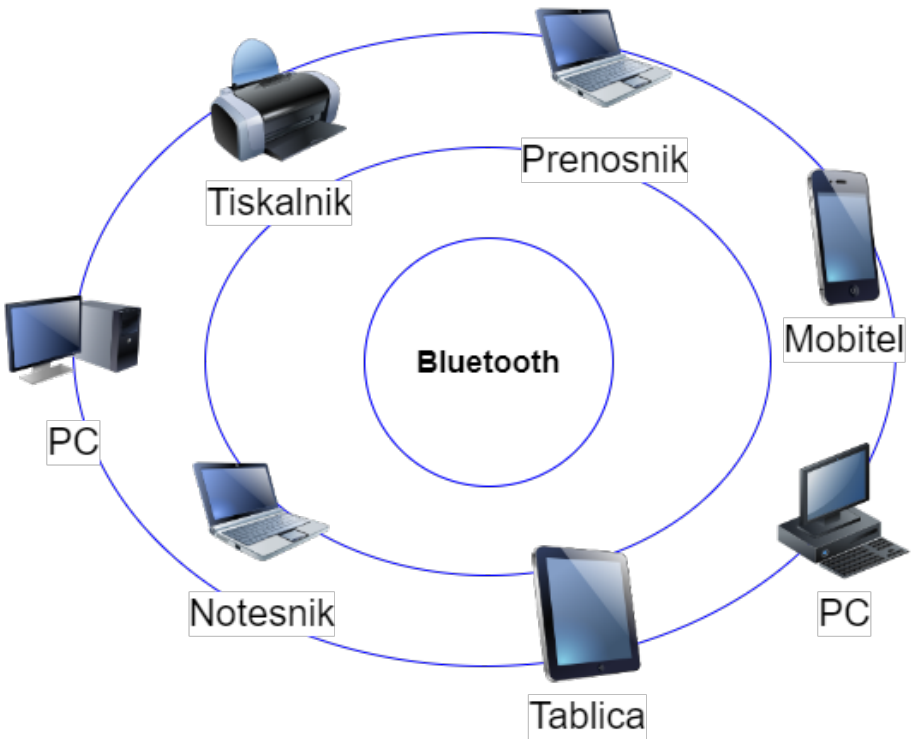
WLAN omrežja delujejo na frekvenčnih področjih 2,4 in 5 GHz. Hitrosti prenosa podatkov v sodobnih WLAN omrežjih teoretično dosegajo 11 Mbit/s – 10 Gbit/s. Slika 35 prikazuje način povezovanja brezžičnih komunikacijskih naprav na Internet.

Pomanjkljivosti WLAN omrežij so:

1. popačitve signala – napake v prenosu podatkov
2. oteženo vzdrževanje zveze
3. prostorske omejitve dosega signala
4. varnostni problemi




Slika 35: *WiFi naprave in internet*
Vir: lasten




Slika 36: *Bluetooth naprave*
Vir: lasten

Zaradi omenjenih pomanjkljivosti in visokih stroškov ”kabliranja” so za povezovanje naprav na kratkih razdaljah (znotraj sobe/pisarne, neposredni bližini) iznašli tehnologiji Bluetooth in NFC, ki prav tako delujeta na osnovi radijskih zvez, vendar sta v osnovi preprostejši. Tudi tu gre za povezovanje zelo raznovrstnih naprav (Slika 36). Poudarek je na njihovi enostavni P2P povezljivosti za osnovno izmenjavo informacij.

Bluetooth omogoča povezovanje do 8 naprav na majhnih razdaljah (do 10 metrov). Frekvenca delovanja je 2,4 – 2,485 GHz. NFC (Near Field Communication) omogoča povezovanje naprav v neposredni bližini (do 10 cm). Frekvenca

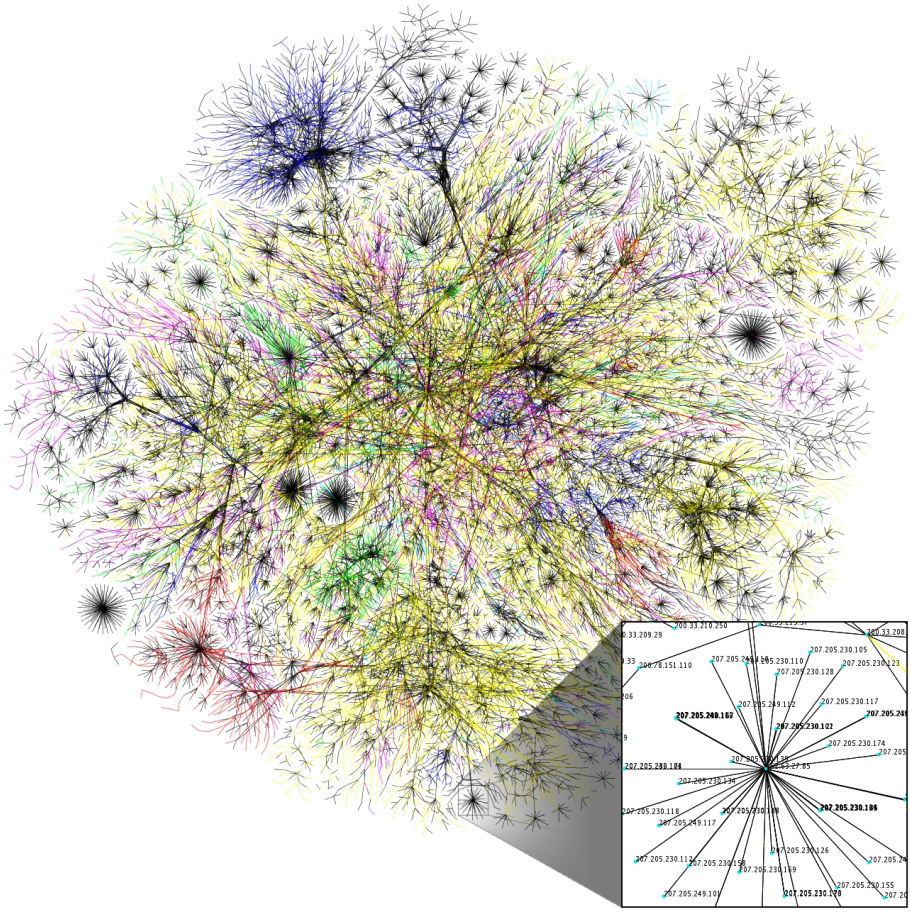
delovanja je 13,56 MHz Omogoča prenosne hitrosti do 424 Kbit/s. NFC naprava lahko deluje kot pametna kartica, kot čitalec pametnih kartic ali kot P2P komunikacijska naprava, kjer lahko podobno kot pri Bluetoothu pari NFC naprav varno komunicirajo med seboj. Prav varnost prenosa podatkov je tista lastnost, ki predstavlja bistveno razliko med Bluetooth in NFC tehnologijo.

## 4.3 Internet

*Internet ali medmrežje* je globalno računalniško omrežje. Slikovito ga lahko predstavimo kot pajkovo mrežo, ki povezuje računalnike po celem svetu. Slika 37 slikovito prikazuje povezave med vozlišči interneta, ki pripadajo različnim omrežjem. Za povezavo v medmrežje računalniki uporabljajo že omenjene mrežne naprave ter žične ali brezžične povezave.

Internet je *globalna javna dobrina* – ne pripada nikomur in nihče si ga ne lasti. Celoten internetni prostor v fizičnem in informacijskem smislu imenujemo *kibernetski prostor (angl. cyberspace).* Uporaba Interneta kot takšnega ni plačljiva, uporabnik pa običajno ima stroške nabave opreme in stroške vzpostavitve in vzdrževanja komunikacijske povezave. Organizacije, ki nudijo storitve priključitve na Internet imenujemo *ISP-ji (angl. Internet Service Provider).*

Ni odveč če se na kratko seznanimo z zgodovino Interneta, da bomo lažje razumeli vzgibe, na podlagi katerih je nastal ter skozi čas prerasel prvotno zamisel – medsebojno povezovanje neodvisnih heterogenih omrežij. V ZDA so leta 1958 ustanovili agencijo ARPA (Advanced Research Projects Agency). Naloga te organizacije je bila izgradnja računalniškega omrežja, ki bi lahko neovirano delovalo tudi če bi bil njen večji del uničen (npr. po jedrskem napadu). Raziskovanje se je začelo leta 1968, prvi rezultat pa je bilo testno računalniško omrežje ARPANET, ki je povezovalo 4 univerze v ZDA (4 računalnike z različnimi operacijskimi sistemi). To omrežje je bilo osnutek Interneta. Nadaljnji miljski kamni plastično prikazujejo njegov razmah:



Slika 37: *Internet*
Vir: commons.wikimedia.org, The Opte Project, CC-BY, 2006

**1971** v ARPANET povežejo 23 računalnikov (predvsem za vojaške namene)

**1987** ARPANET počasi preraste v Internet, ki povezuje okoli 10.000 računalnikov več akademskih institucij in preneha biti vojaško omrežje

**1989** 100.000 povezanih računalnikov; Internet postane svetovno omrežje

**1991** povezuje 33 držav s 700.000 računalniki

**1992** povezuje 49 držav z 1.200.000 računalniki

**1993** Bela hiša in Združeni narodi so dostopni preko Interneta; pojav prvih komercialnih oglaševalcev; 59 držav z 2.000.000 računalniki

**1994** 75 držav z 3.500.000 računalniki; popularizacija Interneta na osnovi WWW servisa

**1996** 50.000 računalniških omrežij z okoli 6.000.000 računalniki; Internet uporablja okoli 40 milijonov ljudi

**2007** Internet postaja nuja vsakdanjika; uporablja ga okoli milijarda ljudi v skoraj vseh državah sveta ...

Ta skokovit razvoj je dobro izrazil Bill Gates: "*Internet je plimni val. Odplavil bo računalniško in druge industrije in potopil tiste, ki se ne bodo naučili plavati na njegovih valovih!*"

### 4.3.1 Povezovanje v Internet

Kaj potrebujemo za povezavo z Internetom? Odvisno od točke priklopa potrebujemo mrežno kartico, ki omogoča žično ali brezžično povezavo v LAN ali modem (npr. ADSL, kabelski, GSM,...).

Omrežje od nas zahteva konfiguracijo naše Internetne povezave:

1. IP številka (enolična identifikacija terminala)

2. Številka prehoda
3. Številka DNS strežnika
4. Način dostopa (PPP, LAN, WLAN, GSM, ipd.)
5. Posebne zahteve:
   (a) uporabniško ime in geslo
   (b) SSID (WLAN)
   (c) APN (GSM)

*IP številka (angl. IP Address)* enolično definira računalnik na internetu, kot telefonska številka enolično definira telefonski priključek. V osnovi je IP naslov (IPv4) 32-bitno število, sestavljeno iz štirih zlogov – 4 decimalnih števil iz obsega 0-255 (4x8 bitov), ki so zaradi preglednosti običajno predstavljeni kot desetiška števila, ločena s piko (npr. 164.8.132.245). Seveda ta delitev ni le optična, ampak ima (spet podobno kot pri telefonskih številkah) tudi svoj pomen (Tabela 2):

**A** številka domene (UM)

**B** številka omrežja (FL UM)

**C** številka podomrežja (segmenta)

**D** številka priključka

V luči uvajanja interneta stvari in posledične potrebe po razširitvi prostora IP naslovov so standardizirali IPv6, ki je 128-bitni IP naslov.

IP naslovi so lahko:

1. stalni/fiksni (imajo jih strežniki)
2. začasni/dinamični (imajo jih odjemalci)

Tabela 2: *Razlaga IP naslova*

| **A** | **B** | **C** | **D** |
|---|---|---|---|
| 164 | 8 | 132 | 245 |

Začasni naslov pridobimo ob priklopu na Internet in je vsakič drugačen – ko računalnik "odklopimo" z Interneta, ta IP naslov lahko prevzame drug odjemalec.

Podobno kot pri telefonskih imenikih so se tudi v primeru IP naslovov uveljavila (simbolična) imena, saj si jih lažje zapomnimo. Predvsem je uporaba simboličnih imen razširjena pri dostopu do računalnikov, ki hranijo in posredujejo vsebine na Internet – spletnih strežnikov (npr. www.um.si, ftp.arnes.si). Tudi ta imena so sestavljena na določen način:

1. ime servisa (je lahko www, ftp, smtp, pop,...)
2. ime strežnika (ni obvezno) – dodeli ga administrator strežnika in pogosto označuje delovno mesto ali servis, ki ga nudi
3. ime domene (najpogosteje ime institucije ali podjetja) – potrebno ga je registrirati s strani državne institucije (npr. Arnes) ali preko *posrednika (angl. web hosting providers)*
4. ime korenske domene

*Korenska (angl. root) ali glavna domena* je fiksna in je določena z mednarodnimi standardi (glede na profil podjetja oz. njegovo primarno dejavnost ali državo kjer je ustanova registrirana):

1. .com (komercialna podjetja)
2. .edu (izobraževalne ustanove)
3. .gov (državne ustanove)

4. .mil (vojaške ustanove)
5. .net (organizacije mrežnih dejavnosti)
6. .org (neprofitne organizacije)
7. .si (Slovenija)
8. .hr (Hrvaška)
9. .de (Nemčija)...

Potrebno je bilo zagotoviti "telefonski imenik" strežnikov, saj uporabniki uporabljajo simbolične naslove, računalniki pa razumejo samo binarne IP naslove. Najprej je to vlogo opravljal le en strežnik SRI (Stanford Research Institute), katerega *hosts.txt* datoteka je hranila naslove najpomembnejših internetnih strežnikov.

Zaradi naraščanja števila vnosov in pogostih sprememb, je bil razvit *sistem imenskih strežnikov* (*Domain Name System*, *DNS)*, ki omogoča avtomatsko sporočanje sprememb naslovov preko omrežja in ga vzdržuje sistem distribuiranih podatkovnih baz, ki deluje po principu odjemalec-strežnik. Vozlišča tega sistema imenujemo *imenski strežniki* (angl. *nameservers*). Vsaka domena ima vsaj en avtoritativen imenski strežnik, ki objavlja podatke domene in njenih pod-domen; na vrhu hierarhije (angl. Top Level Domain oz. TLD) je t.i. *korenski imenski strežnik* (angl. *root nameserver*).

### 4.3.2 Internetni protokol

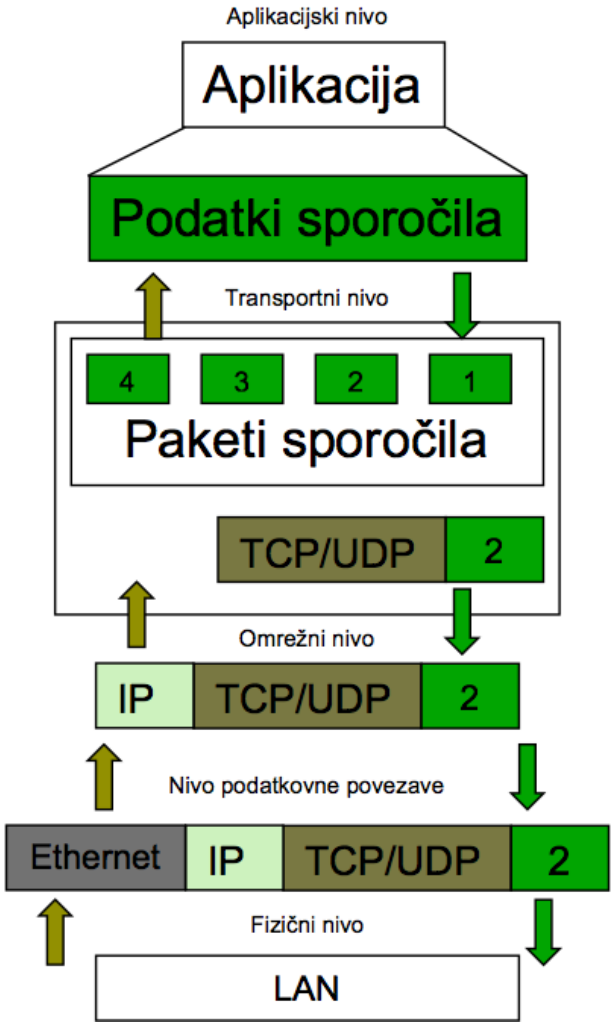
Seveda moramo za povezavo z Internetom, razen omenjenih pristopnih podatkov, ki nas definirajo kot uporabnika omrežja, preko katerega se priključujemo in način naše povezave, govoriti tudi ustrezen "jezik". Na najvišjem nivoju (operacijskega sistema) je ta jezik poenoten, na nižjem pa zaradi raznolikosti opreme nujno prihaja do razlik v izvedbi. Zato je bilo na nivoju servisov,

ki jih nudijo, poenotiti več nivojev komunikacije – standardizacija. Najpomembnejša protokola sta *TCP (Transmission Control Protocol)* in *IP (Internet Protocol)*. Naloga TCP protokola je, da se podatki med prenosom ne izgubijo, naloga IP protokola pa, da poišče pot za komunikacijo od enega računalnika do drugega. Ker sta povezana (ne moreta eden brez drugega), večkrat govorimo kar o *TCP/IP protokolu*.

Slika 38 prikazuje vse faze komunikacije po nivojih *ISO/OSI referenčnega modela (angl. ISO/OSI Protocol Stack)*. Pot podatkov se začne pri vozlišču, ki podatke pošilja, oz. njegovi aplikaciji. Ta jih preda transportnemu nivoju, ki sporočila razbije na pakete. Zaradi raznolike dolžine in strukture sporočil ter kapacitete prenosnih medijev razbijemo sporočila na podatkovne pakete enake strukture in določene maksimalne velikosti. Da se ne izgubijo in da ohranimo izvirno strukturo sporočila, pakete na tem nivoju opremimo z zaporedno številko in upravljavskimi podatki. Omrežni nivo v glavo paketov doda njihov izvor in cilj. Nivo podatkovne povezave doda podatke o vozliščih in jih preda fizičnemu nivoju, da jih prenese. Vsak paket na svoji poti do cilja išče najmanj obremenjeno (najhitrejšo) pot. Na drugi strani ustrezni nivoji iz paketov izluščijo vsebino in jo pri prejemniku spet sestavijo v izvirno sporočilo. Prej omenjeni TCP protokol zagotavlja dostavo paketov v pravilnem vrstnem redu in če to ni mogoče, sporoči napako.

Struktura paketa v ISO/OSI referenčnem modelu:

1. *Glava (angl. header)* vsebuje naslov prejemnika sporočila
2. Informacijski del vsebuje izbrani del izvirnega sporočila
3. *Zaglavje (angl. flag)* vsebuje naslov pošiljatelja sporočila, zaporedno številko paketa, pariteto (kontrola prenosa) in še nekaj upravljavskih informacij




Slika 38: *ISO/OSI referenčni model*
Vir: lasten

Za manjše podatke, ki se praviloma prenesejo v enem paketu, je možno uporabiti UDP (User Datagram Protocol) protokol. Le-ta omogoča prenos *uporabniških sporočil* (*angl. datagram*), ki so med sabo neodvisni. Ne preverja prispetja paketov, vrstnega reda in morebitnih podvojenih paketov. UDP protokol se uporablja za pošiljanje točnega časa, preverjanje delovanja povezave (ping), ipd. Ker ne potrebuje vnaprejšnje vzpostavitve povezave med računalnikoma pred prenosom (IP) in preverjanja dostave paketov (TCP), je ta protokol hitrejši.

Delovanje Interneta sloni na arhitekturi *odjemalec–strežnik*, kar omogoča zmanjšanje obremenjenosti prenosnih poti. Strežniška *vrata (angl. port number)* so aplikacijsko- oz. procesno-specifičen programski konstrukt, ki služi kot komunikacijski priključek transportnemu nivoju nabora internetnih protokolov TCP in UDP. Vsaka vrata identificira številka vrat (16-bitna številka iz obsega 0 do 65.535) in IP naslov ter protokol za komunikacijo. Določene številke vrat (0 do 1023) so rezervirane za splošne internetne servise (npr. HTTP ima številko 80).

Na vratih strežnika se praviloma srečata komunikacijska programa odjemalca in strežnika. Na strežniku imamo za določena vrata registrirane ustrezne programe, ki strežejo ves promet na določenih vratih. V primeru TCP protokola odjemalec ”govori” neposredno s tem strežniškim programom. V primeru UDP protokola pa je številka vrat zakodirana v paketu in jo strežnik sprejme na namenskih vratih protokola, dekodira ter nato posreduje vsebino paketa ustreznemu strežniškemu programu glede na številko vrat. Zaradi načina obravnave UDP paketov jih požarni zidovi v računalniških omrežjih pogosto ne prepuščajo.

### 4.3.3 Internetni servisi

Sedaj, ko smo se spoznali z internetno infrastrukturo, si lahko pobližje pogledamo posamezne internetne servise. Ko dostopamo do vsebin na svetovnem spletu, se običajno ne zavedamo, kje se vsebine oz. strežnik, ki jih hrani,

Tabela 3: *Vrata znanih internetnih servisov*

| **Protokol** | **Številka vrat** | **Opis** |
|---|---|---|
| HTTP | 80, 8080 | prenos spletnih strani |
| HTTPS | 443 | varen prenos spletnih strani (uporaba protokola SSL) |
| SMTP | 25 | pošiljanje e-pošte |
| SMTP SSL | 465 | varno pošiljanje e-pošte |
| POP | 110 | sprejem e-pošte |
| POP SSL | 995 | varen sprejem e-pošte |
| TELNET | 23, 3535 | delo na oddaljenem računalniku |
| RMI | 1099 | oddaljeno proženje aplikacij |
| ping | (UDP) | preizkus delovanja omrežja |
| FTP | 20 in 21 | prenos datotek |
| sFTP/SSH | 22 | varen prenos datotek |

fizično nahajajo. Zmotno je tudi pogosto enačenje Interneta in svetovnega spleta. V tem primeru gre za odnos "infrastruktura – aplikacija", pri čemer Internet predstavlja infrastrukturo za to in še številne druge aplikacije oz. *internetne servise*. Tabela 3 vsebuje seznam nekaterih standardnih internetnih servisov.

**Svetovni splet** *WWW (World-Wide Web) ali W3* je najpogostejši in najustreznejši način prikaza informacij na Internetu. WWW strani so prikazane na uporabnikovem monitorju in so podobne stranem knjige. "Rojen" je bil leta 1991 v Švici na inštitutu CERN kot sredstvo med-oddelčne izmenjave raziskovalnih informacij.

*Spletne strani (angl. web page)* lahko vsebujejo tekst, (animirane) slike, zvok itd. Razen tega lahko vsebujejo še *hiperpovezave* (*angl. hyperlink*), preko katerih dostopamo do informacij na povezanih spletnih straneh. Skrajšano govorimo običajno o *linkih*. Pri tem je *link* lahko:

1. katerakoli beseda,

2. katerakoli slika,

3. drug naslovljiv element spletne strani.

Običajno nas ”klik na link” prenese na novo spletno stran v istem oknu, ali pa se za prikaz te strani odpre novo okno/zavihek v brskalniku.

Skupino spletnih strani, ki se nahajajo na isti domeni in so med sabo povezane, imenujemo *spletišče* (angl. *web site*). Korensko stran vsakega spletišča imenujemo *domača stran (angl. home page)* in služi kot *vstopna točka* pri brskanju po ostalih straneh istega ali povezanih spletišč. Strani lahko pregledujemo/brskamo z *brskalniki (angl. browser)*, kot so na primer Internet Explorer, Chrome, Firefox ali Opera. Posamične spletne strani so, ko so prečitane/naložene s strežnika na naš računalnik, prikazane v oknu brskalnika.

Kako torej dostopamo do vsebin spletnega strežnika? Poznati moramo ime (IP naslov) spletnega strežnika, vrsto servisa in natančno lokacijo vsebine – URI (Uniform Resource Identifier). URI enolično določa lokacijo vsakega vira na spletu. Sestavljen je iz dveh delov:

1. URN – Uniform Resource Name, ki določa ime vira, ne določa pa njegove lokacije (npr. studenti-prvega-letnika-UN) in/ali

2. URL – Uniform Resource Locator, ki natančno določa metodo za iskanje vira: protokol+lokacija v omrežju.

**Primer:** URI je lahko na primer ”http://fl.um.si/UN/studenti-prvega-letnika-UN.html”

Pa poglejmo ali imamo vse informacije. Protokol za dostop oz. ime servisa je v našem primeru HTTP in je ločen od ostanka URL z ”://”. To je običajni spletni protokol, ki nam omogoča objavo podatkov na spletu v obliki t.i. spletnih strani. HTTP je *request/response* protokol za komunikacijo med strežnikom in odjemalcem. Preko navedbe URL, HTTP odjemalec, kot na primer Internet

Explorer (IE), sproži prenos podatkov po tem, ko je vzpostavil TCP povezavo z oddaljenim spletnim strežnikom. Pred tem smo ga seveda morali najti – to nalogo je prevzel DNS strežnik, ki je razrešil spletni naslov *fl.um.si*. Na tem naslovu se običajno nahaja (korenska "/") domača stran (npr. *index.html*), ki se naloži, če ne navedemo poti do specifičnega spletnega vira. V našem primeru smo navedli celotno pot do spletnega vira in, če omenjena datoteka obstaja, se bo naložila in jo bo naš brskalnik prikazal.

**Elektronska pošta** *Elektronska pošta (e-mail)* je enostaven in hiter način komuniciranja po elektronski poti. Za to vrsto komunikacije potrebujemo računalnik in dostop do Interneta. Komunikacija se odvija preko e–poštnih strežnikov in programskih ali spletnih odjemalcev (npr. Outlook, Thunderbird, Gmail, ipd.).

ISP vsakemu uporabniku dodeli e–poštni naslov v obliki: *ime@organizacija.področje* (npr. roman.gumzej@um.si). Znak @ čitamo kot "afno" (at, master space ali monkey), *organizacija.področje* pa predstavlja domeno. e-poštni naslovi v osnovi ne morejo vsebovati diakritičnih znakov (npr. š,č,ž).

Spletni servisi, ki skrbijo za izmenjavo e-pošte pri e-poštnih strežnikih so:

1. SMTP (Simple Mail Transfer Protocol) omogoča pošiljanje e-sporočil, ki shranjena v e-poštnem predalu in nato pri e-poštnem strežniku čakajo na odjemalca

2. POP3 (Post Office Protocol v.3) in IMAP (Internet Message Access Protocol) omogočata uporabnikom prevzem/prebiranje njihovih e-sporočil iz e-poštnih predalov e-poštnih strežnikov preko omenjenih odjemalcev

Za spletni dostop do e-pošte se običajno uporablja protokol IMAP, ki privzeto pušča sporočila na strežniku, medtem ko jo POP3 po prevzemu izbriše.

**FTP** *FTP (File Transfer Protocol)* je protokol za prenos podatkov z enega računalnika (najpogosteje odjemalca) na nek drug računalnik (najpogosteje strežnik) preko Interneta ali lokalnega omrežja. *Pošiljanje datotek* oz. *nalaganje* na (oddaljeni) strežnik strokovno imenujemo *upload*. *Sprejem datotek* oz. *snemanje* s strežnika pa strokovno imenujemo *download*.

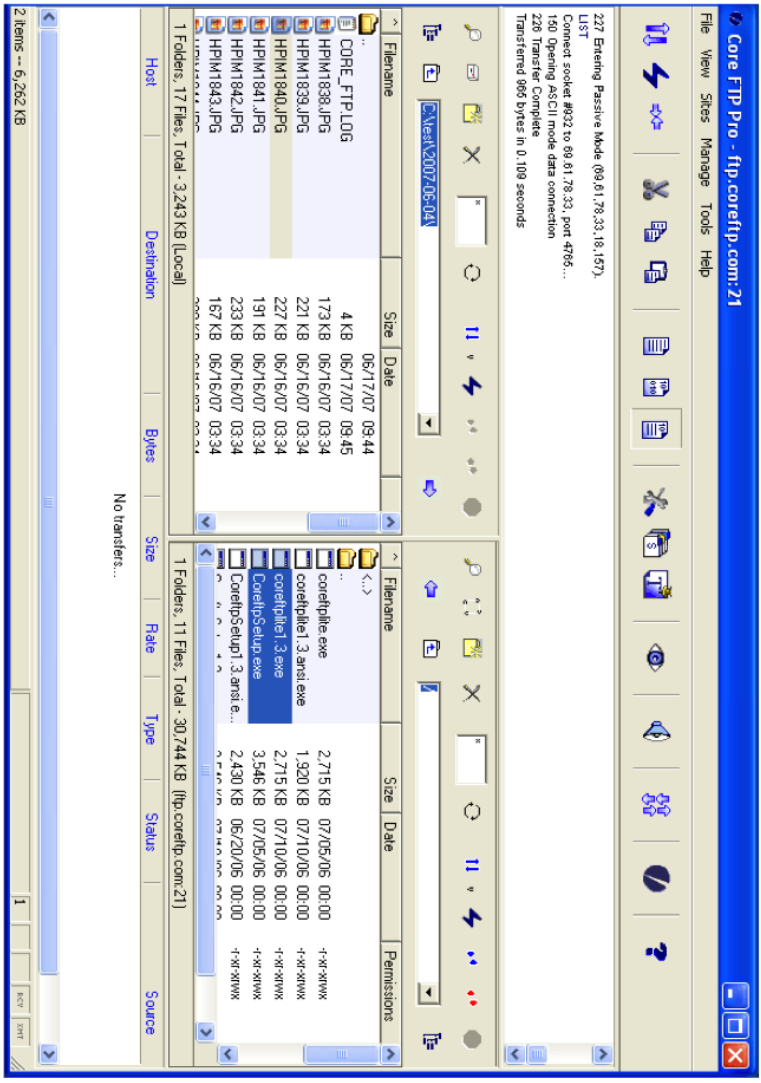
Pri povezovanju s FTP strežnikom, se vzpostavita dve seji:

1. kontrolna seja, ki sprejema ukaze
2. podatkovna seja, ki skrbi za prenos podatkov

Ta servis je vgrajen v operacijski sistem in spletne brskalnike, obstajajo pa tudi posebni programski odjemalci, ki nam lahko olajšajo delo pri prenašanju večjega števila raznovrstnih datotek. Pri operacijah nalaganja/snemanja potrebujemo uporabniško ime in geslo za avtorizacijo dostopa do strežnika. Včasih je dovoljeno tudi snemanje datotek brez posebnega dovoljenja (uporabniško ime: *anonymous*), običajno pa so javno dostopni le posamezni imeniki oz. datoteke znotraj njih (t.i. "FTP public folders").

Slika 39 prikazuje primer grafičnega FTP odjemalca. V zgornjem oknu lahko spremljamo elemente kontrolne seje, v spodnjem pa lahko manipuliramo datoteke podatkovne seje.

**Telnet** *Telnet* je programski klient za oddaljen dostop do računalniških sistemov preko Interneta. Omogoča delo na drugem računalniku v tekstovnem načinu (terminal/konzola). V zadnjem času ga v okolju Windows zamenjuje *Remote Desktop*, v okolju Unix/Linux operacijskih sistemov pa je običajen *X-terminal* dostop.




Slika 39: *FTP odjemalec*
Vir: lasten, zajem zaslona, 2023

Podobno kot pri FTP za prijavo na drug računalnik običajno potrebujemo uporabniško ime in geslo, lahko pa je omogočen tudi javen dostop (*guest/anonymous*). Računalnik, do katerega dostopamo na daljavo, se v tem primeru obnaša kot strežnik, ki omogoča dostop našemu telnet-odjemalcu.

**Drugi spletni servisi** *Programi za izmenjavo takojšnjih sporočil (angl. Instant messaging)* so odjemalci, ki nam omogočajo izmenjavo tekstovnih in multimedijskih sporočil v realnem času (npr. Skype, Viber, WeChat, ipd.).

Pojavili so se tudi *omrežni časopisi* (npr. *UseNet, Slo-Tech,...*), ki omogočajo elektronski način izmenjave mnenj med uporabniki. Uporabnik pošlje sporočilo časopisu, na katerega je prijavljen, in ima vpogled v sporočila drugih uporabnikov. Zaradi velike količine sporočil, so le-ta običajno razdeljena v *tematske skupine* (angl. *newsgroups*). Komunikacija je vsebinsko podobna e-pošti, le da so sporočila vidna vsem uporabnikom iste skupine.

S številom mobilnih uporabnikov in dostopnostjo Interneta na mobilnih napravah narašča tako obseg kot raznolikost spletnih servisov, ki obsegajo številna področja dela in zabave – od spletnega nakupovanja do spremljanja rezultatov svojega treninga.

## 4.4 Intranet in ekstranet

Številne organizacije so izgradile lastna omrežja, zasnovana na načelih delovanja Interneta. Tako so zasnovali t.i. *Intranet* in *Ekstranet*.

Lastnosti teh privatnih mrež so:

1. arhitektura odjemalec-strežnik
2. uporabljajo enake komunikacijske protokole kot Internet
3. servisi so organizirani na enak način kot v Internetu

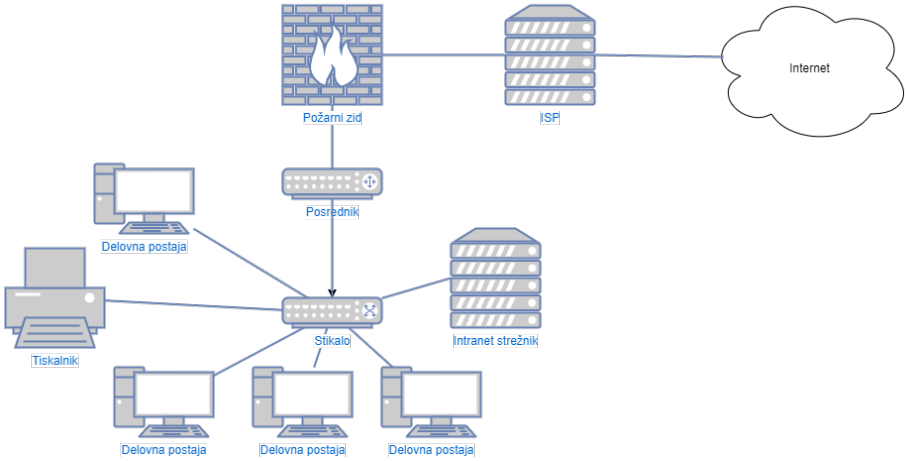
*Intranet* je privatna mreža organizacije, ki deluje po načelih Interneta. Možne vloge Intraneta v informacijskem sistemu so:

1. zamenjava za lokalno omrežje podjetja
2. povezovanje več lokalnih mrež podjetja
3. zamenjava za prostrano mrežo podjetja
4. povezovanje lokalnih in prostranih mrež velikih podjetij

Intranet je praviloma sestavni del Interneta, potrebna pa je vgradnja *požarnega zidu* med njima zaradi varnosti uporabnikov Intraneta in njihovih podatkov. *Požarni zid* (angl. *firewall*) je naprava ali računalniški program, ki zagotavlja:

1. neomejen pretok informacij iz Intraneta v Internet
2. omejen dostop iz Interneta k Intranetu

Slika 40 prikazuje tipično konfiguracijo Intraneta. V računalniških mrežah je *posrednik (angl. proxy)* računalniški sistem (strežnik) ali aplikacijski program, ki deluje kot posrednik med zahtevami odjemalcev, ki zahtevajo vire drugih strežnikov. Odjemalec se poveže s posredniškim strežnikom z zahtevo po določenem servisu (npr. datoteki, povezavi, spletni strani,...). Pri tem posrednik ovrednoti zahtevo glede na pravila filtriranja (promet filtrira glede na IP naslove ali uporabljene protokole). Če je zahteva veljavna, posrednik zagotovi dostop do spletnega vira s tem, da se poveže z ustreznim strežnikom in ga zahteva v imenu odjemalca. Posrednik lahko po potrebi spremeni/prilagodi zahtevo odjemalca ali strežnikov odgovor (npr. formatira spletno stran za mobilno napravo), lahko pa postreže zahtevo tudi brez posredovanja specificiranega strežnika, pri čemer običajno črpa iz zgodovine odzivov oddaljenega strežnika (angl. caching), da lahko sam postreže ponavljajoče se zahteve po istem viru.

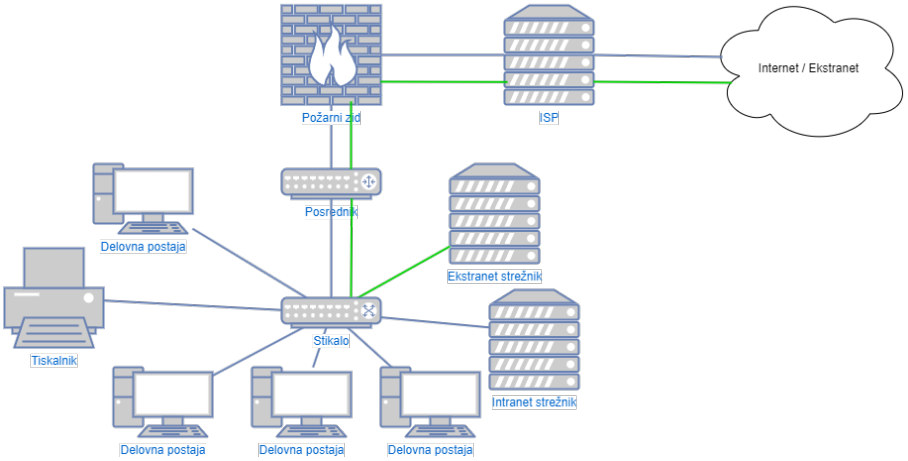



Slika 40: *Intranet in Internet*
Vir: lasten

Posredniški strežnik služi več namenom:

1. anonimnost odjemalcev/strežnikov za njim (zaradi varnosti)
2. pohitritev dostopa do spletnih virov (caching)
3. vpeljava pravil dostopa do spletnih servisov ali vsebin (npr. blokiranje nezaželenih vsebin)
4. avtorizacija in beleženje dostopov – poročanje o uporabi Interneta
5. pregled vsebine pred pošiljanjem (zaščita pred trojanskimi konji, preprečevanje "uhajanja" podatkov,...)
6. da obidemo regionalne omejitve

Za razliko od požarnega zidu, ki deluje na nivoju paketov podatkov, posrednik deluje na nivoju spletnih servisov.




Slika 41: *Povezava Intraneta, Ekstraneta in Interneta*
Vir: lasten

*Ekstranet* je skupina neodvisnih privatnih mrež dveh ali več poslovnih partnerjev, povezanih po načelih Interneta. Slika 41 prikazuje tipično konfiguracijo Ekstraneta. Smiselno ga je razvijati le, ko imajo njegovi udeleženci vzpostavljene lastne *Intranete*. Vsak Intranet ima lastni požarni zid za:

- zaščito neodvisnih Intranetov med sabo (potrebno je zagotoviti, da določene informacije ostanejo izven dosega poslovnih partnerjev)
- zaščito celotnega Ekstraneta pred nezaželenimi vpadi iz Interneta

# 5 Računalniška programska oprema (SW)

V okviru tega poglavja bomo opredelili pojem programske opreme, podali njeno klasifikacijo ter opisali vlogo in glavne funkcije operacijskega sistema, aplikacijskih programov in programskih jezikov. Spregovorili bomo tudi o vplivu programskih orodij, okolij in spletnih servisov na produktivnost.

Po tem, kar smo obravnavali doslej, lahko sklepate, da je računalnik zmogljiv računski stroj, ki je sposoben s pomočjo specializiranih perifernih naprav avtomatizirati izvedbo številnih in raznolikih nalog. Programska oprema, s katero se srečamo ob njegovi uporabi, je njegova nematerialna komponenta, ki omogoča izvedbo, nadzor in spremljanje rezultatov izvrševanja zahtevnih nalog, ki jih lahko s pomočjo računalnika izvedemo lažje in bolj učinkovito kot sicer.

Računalnik v osnovi interpretira ukaze, zapisane kot kombinacije ničel in enic v t.i. *strojnem jeziku (angl. machine language)*, ki jih prenaša iz delovnega pomnilnika v aritmetično logično enoto in jih tam izvaja enega za drugim. Strojni jezik (torej jezik, ki ga stroj razume) je neposredni prevod *zbirnega jezika (angl. assembly language)*, ki predstavlja njegovo *človeško berljivo (angl. human-readable)* različico. Prvi *računalniški programi* so bili zapisani kot nizi ukazov v zbirnem jeziku. Za izvajanje zahtevnih nalog potrebujemo kompleksnejše programe, ki jih programer v zbirnem jeziku ne bi mogel v doglednem času zapisati konsistentno in brez napak, četudi bi bil zelo pazljiv in sistematičen. Zaradi tega so bili razviti *visoko-nivojski programski jeziki*, ki omogočajo opis računskih postopkov na višjih nivojih abstrakcije in so manj podvrženi programerskim napakam. Specializirani programi – t.i. *prevajalniki (angl. compilers)* nato te programe sistematično prevedejo v nize ukazov zbirnega in končno strojnega jezika. Sedaj že lahko podamo definiciji programske opreme in računalniškega programa kot njegove sestavne komponente.

***Programska oprema*** *so računalniški programi, napisani v enem izmed programskih jezikov in prevedeni v strojni jezik računalnika, na katerem se izvajajo.*

***Računalniški program*** *je navodilo računalniku, zapisano kot zaporedje ukazov v strojnem jeziku.*

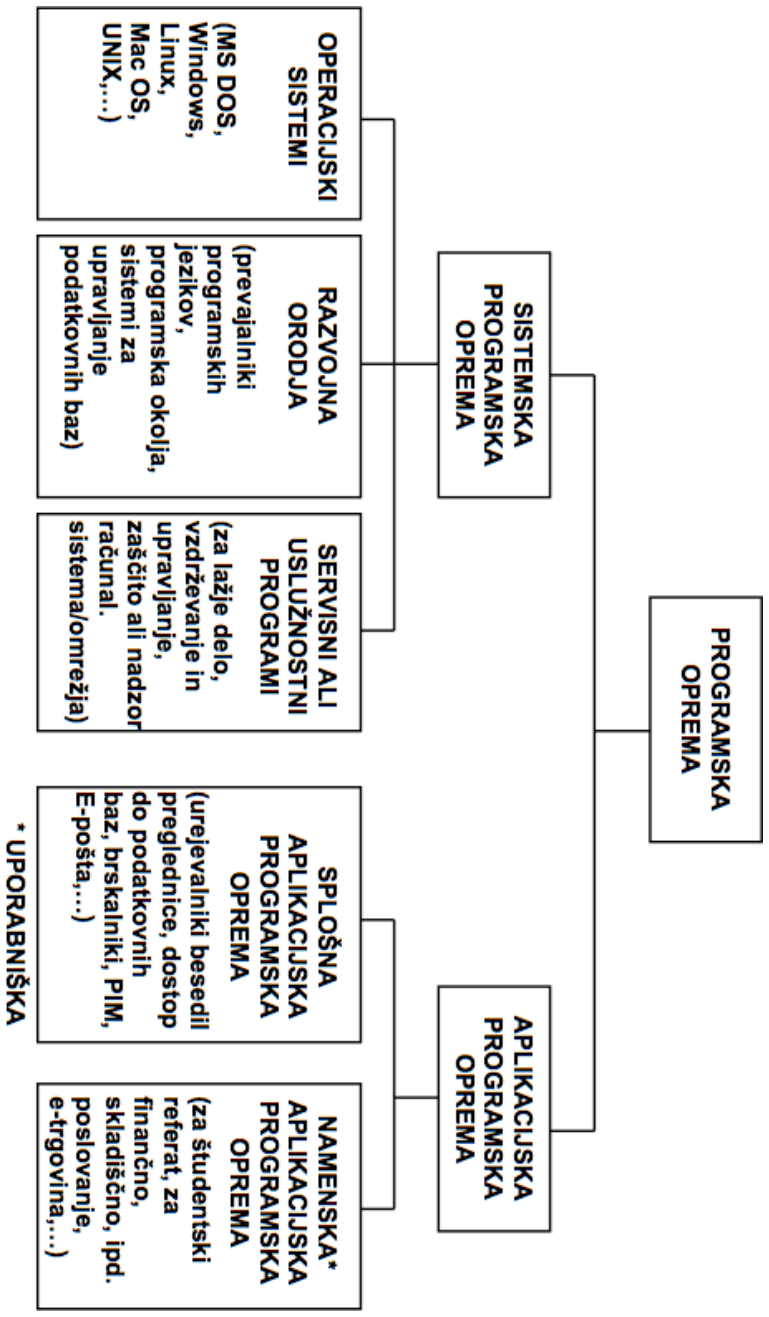
Programsko opremo delimo po funkcionalnosti na *sistemsko* in *aplikacijsko programsko opremo:*

***Sistemska programska oprema*** *je nabor računalniških programov, ki služijo primarno kot vmesnik med računalniško strojno in programsko opremo.*

***Aplikacijska programska oprema*** *je razred računalniških programov, ki služijo za usmerjeno delovanje računalniškega sistema, da opravlja specifične aktivnosti, ki služijo uporabniku.*

Slika 42 prikazuje klasifikacijo programske opreme. Med *sistemsko programsko opremo* prištevamo operacijske sisteme, razvojna orodja ter servisne/uslužnostne programe, med *aplikacijsko* pa *splošno namensko aplikacijsko programsko opremo* in *namensko aplikacijsko programsko opremo*. Ker gre pri namenski programski opremi za izpolnjevanje zahtev specifičnih uporabnikov ali uporabniških skupin, to programsko opremo večkrat imenujemo tudi *uporabniška*. *Uporabniško programsko opremo* običajno razvijamo znotraj organizacije za potrebe organizacije, lahko pa to nalogo zaupamo tudi zunanjemu podjetju. Komercialno *splošno namensko ali COTS (Commercial Off-The Shelf) programsko opremo* podjetja za razvoj programske opreme razvijajo za širši krog uporabnikov in jih ponujajo na trgu – od tod ime.

Aplikacijski računalniški programi se razvijajo za določene *računalniške platforme*, ki jih opredeljuje predvsem uporaba določenega *operacijskega sistema* in *procesorja*. Zaradi velike konkurence na trgu programske opreme imamo običajno na voljo več sorodnih aplikacijskih programov za različne platforme.




Slika 42: *Delitev programske opreme*
Vir: lasten

## 5.1 Lastništvo programske opreme

Programsko opremo lahko pridobimo v različnih oblikah in na različnih nosilcih podatkov. Lahko jo, kot na primer operacijski sistem, kupimo skupaj s strojno opremo. Lahko jo dokupujemo naknadno po potrebi. Medtem ko smo računalniške programe še nedavno dobili na eni ali več disketah, sodobni programi običajno zahtevajo več spomina, zato so shranjeni na CD, DVD ali celo BD nosilcih podatkov. Vse pogosteje pa si jih lahko tudi naložimo z Interneta oz. t.i. *aplikacijskih tržnic (app(lication) market/store, ...)* na lokalni trdi disk in od tam instaliramo. Obstajajo tudi spletni programi in servisi, do katerih dostopamo preko *spletnega brskalnika (angl. browserja)* in nam jih ni potrebno instalirati.

Glede na številne pojavne oblike se je že zgodaj pojavila potreba po zaščiti avtorskih pravic in lastništva programske opreme. Ker običajno gre za zunanji razvoj in vzdrževanje programske opreme, nam po navadi podjetje, ki je program izdelalo, hkrati z medijem za instalacijo programa ali programskega okolja podeli tudi licenco ali več licenc za uporabo. Kot lahko sklepate iz pravkar omenjenega, gre za to, da lahko določen program instalirate na več računalnikih, lahko pa si več uporabnikov tudi deli dostop do programa na primer na lokalnem strežniku. Najenostavnejša oblika licence je seveda: en program – en računalnik – en uporabnik.

Kot smo že omenili, obstaja tudi možnost uporabe *spletnih programov* in tudi za te načeloma moramo pridobiti licenco ali vsaj podati soglasje s pogoji njihove uporabe. Gre za že omenjeno *uporabniško računalništvo.* Podjetja, ki posedujejo, izdelujejo in vzdržujejo aplikacijske programe in računalniške resurse imenujemo *ponudniki aplikacijskih storitev (angl. Application Service Providers, ASP)*. Njihove aplikacijske programe uporabljamo preko Interneta, dostop do njih pa lahko zakupimo začasno (sezonsko ali enkratno) ali za stalno. Podobno delujejo tudi on-line videoteke ter video-on-demand (VOD) sistemi, kjer pa ne zakupimo pravice do uporabe programske opreme, ampak le pravico

za *snemanje (angl. download)* in/ali *nalaganje (angl. upload)* določenih avtorsko zaščitenih vsebin (glasbe ali videa). V tem primeru običajno ne gre za časovni zakup licence, ampak za posamične licence za ogled naloženih vsebin.

Pri programski opremi je običaj, da je le-ta pod licenco (COTS, ASP) v celoti. Pravzaprav, podobno kot pri e-videoteki, ne kupimo programa (njegove izvorne kode in pripadajočih programskih knjižnic), ampak le licenco za njegovo uporabo pod pogoji *licenčnega sporazuma (angl. licensing agreement)*. Ti narekujejo trajanje licence, upravičenost do posodobitev programske opreme, koliko oseb jo sme uporabljati, ali jo smemo razširjati (instalirati na več računalnikih), itd. Licenca ščiti pravice lastništva.

Kaj pomeni *odprtokodna licenca (angl. Open-Source Definition ali OSD)*? Za razliko od *zastonjske programske opreme (angl. freeware)*, pri kateri dobimo pravico za njeno uporabo zastonj in to v licenčni pogodbi tudi eksplicitno piše, gre tukaj za podelitev licence za celotno programsko opremo. Res gre tudi tukaj v osnovi za zastonjske programe, vendar pri teh ne dobimo le licence za uporabo, ampak dobimo v last hkrati še izvorno kodo programa in ustrezne programske knjižnice, če so le-te pod enako licenco. V tem primeru lahko programsko opremo za lastne potrebe tudi prilagajamo in spreminjamo, če smo to znanje pripravljeni deliti naprej pod enakimi pogoji. Običajno gre tu za določeno interesno skupino in dogovor v okviru skupine, da programsko opremo razvijajo skupaj in morebitne posodobitve in razširitve posredujejo ostalim ter s tem dosežejo njeno boljše delovanje in širšo uporabnost. Ta licenca nam skratka jamči pravico uporabe programske opreme, posedovanja izvorne kode, sprememb izvorne kode in proste distribucije programske opreme. Pogosto ta programska oprema vseeno ni povsem brez stroškov – plačati je treba npr. usposabljanje, uporabniško podporo in določeno spremljajočo dokumentacijo (npr. uporabniške priročnike, ipd.).

**Primeri** odprtokodne programske opreme:

1. Linux – operacijski sistem,

2. Apache Web server – spletni strežniški program,
3. Sendmail – upravitelj e-pošte,
4. Perl scripting language – programski jezik,
5. Open Office – programsko okolje, itd.

V zvezi z odprtokodno programsko opremo še vedno obstajajo tudi določena odprta vprašanja:

1. Skupna cena (vzdrževanje, podpora, dokumentacija,...)?
2. Storitve in podpora (dobimo hkrati tudi posodobitve in možnost javljanja in popravljanja morebitnih napak)?
3. Usposabljanje (šolanje uporabnikov)?
4. Nadgradnje (odvisno od interesne skupnosti so le-te na voljo samo članom skupnosti ali širše)?
5. Zanesljivost (tipično ta programska oprema ne more biti stestirana v tolikšni meri kot COTS programska oprema, vendar se običajno hitreje posodablja)?

To so pomembna vprašanja in jih odprtokodne skupnosti uporabnikov programske opreme same ne morejo zadovoljivo rešiti. Morda pa je rešitev uporaba spletnih aplikacij ASP.

## 5.2 Sistemska programska oprema

Med sistemsko programsko opremo štejemo programe, ki omogočajo in upravljajo delovanje računalniškega in omrežnega sistema. To so:

1. Programi za upravljanje s sistemom (angl. system management programs), ki
   (a) upravljajo s strojno opremo,
   (b) upravljajo z uporabniško programsko opremo,
   (c) upravljajo omrežja in mrežne povezave ter
   (d) upravljajo podatkovne resurse (npr. datotečni sistemi, sistemi za upravljanje baz podatkov), ipd.
2. Programska razvojna orodja in okolja (angl. software development tools and environments), ki
   (a) uporabnikom pomagajo pri razvoju aplikacijskih programov

Funkcionalnost prve skupine sistemskih programov je večinoma združena v sodobnih operacijskih sistemih, ki bodo na kratko predstavljeni za tem. Drugo pa predstavlja večja in bolj raznolika skupina programskih jezikov ter razvojnih programskih orodij in okolij, katerim je namenjeno naslednje poglavje.

### 5.2.1 Operacijski sistem

***Operacijski sistem*** *(angl. Operating System, OS) je osnovni program oz. sklop programov, ki upravljajo računalnik in skrbijo za njegovo učinkovito uporabo.*

Operacijski sistem predstavlja osnovno vez med strojno opremo in uporabnikom. Glede na to, da je računalniški strojni opremi najbližji, upravlja delovanje centralne procesne enote, vhodno/izhodnih in komponent za hrambo podatkov itd. Operacijski sistem tudi poskrbi za zagon ostalih računalniških programov ter krmili izvajanje aplikacijskih programov – jih zažene oz. jim (začasno) prepusti kontrolo nad centralno procesno enoto, upravlja (v primeru, da izvajamo več programov hkrati, koordinira njihovo izvajanje) in zaključi (ob prekinitvi izvajanja zaradi odpovedi programa ali ob normalnem zaključku programa znova prevzame kontrolo).

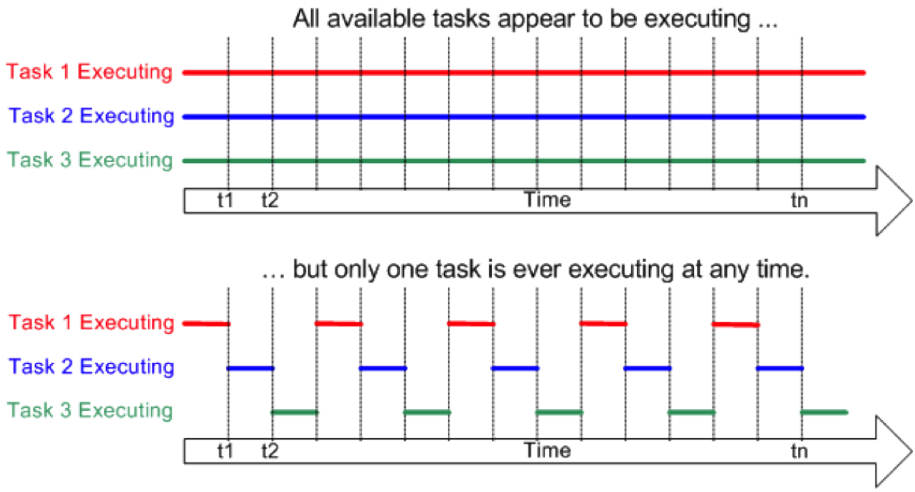
Glavne naloge OS so:

1. upravljanje s procesi
2. upravljanje s pomnilnikom
3. krmiljenje perifernih naprav
4. zagotavljanje uporabniškega vmesnika in dostopa do osnovnih (sistemskih) ukazov
5. delo z datotekami
6. nalaganje in izvajanje programov ter v sodobnem času še
7. zagotavljanje mrežne funkcionalnosti in varnosti ter
8. funkcionalnost Interneta

*Upravljanje s procesi* (angl. process management) nam omogoča urejeno izvajanje opravil na računalniku. Opravila so vsaka logično zaključena sekvenca *programskih ukazov (instrukcij)*, ki nam da nek uporaben rezultat. Običajno eno opravilo izvaja en proces, lahko pa opravilo izvaja tudi več procesov. Procesi predstavljajo opravila v izvajanju. Zato se je udomačila naslednja definicija procesa:

***Proces** je program ali del programa, ki se izvaja v centralni procesni enoti.*

Vsi sodobni operacijski sistemi so več-procesni – lahko izvajajo več procesov "hkrati". Zaradi zaporedne narave obdelave ukazov v procesorju seveda ne gre za dejansko vzporedno izvajanje (razen v primeru več-procesorskih ali več-jedrnih centralnih procesnih enot), ampak gre za navidezno vzporedno izvajanje več aktivnih opravil, med katerimi operacijski sistem deli procesor po določenem zaporedju in jim namenja (enakomerno porazdeljene) *časovne rezine* (angl. time-slice) izvajanja, kar ustvarja vtis vzporednega izvajanja (Slika 43).




Slika 43: *Več-opravilnost (angl. multitasking)*
Vir: medium.com, Medium, Sudeep Chandrasekeran, 2020

Opravilom lahko priredimo prioritete, ki *razvrščevalniku procesov (angl. scheduler)* povedo, da naj procesom, ki pripadajo opravilom z višjo prioriteto dodeli več časa, ali jih obravnava prednostno, da se bodo prej zaključila. Aktivno opravilo se od pasivnega loči po tem, da ima vse pogoje za izvajanje – je bilo aktivirano, se še ni izvedlo do kraja in ima vse potrebne resurse za izvedbo. Zaradi tesne povezanosti obeh pojmov pogosto uporabljamo izraza *več-procesno izvajanje (angl. multi-processing)* in *več-opravilnost (angl. multi-tasking)* izmenjaje. Med tipične predstavnike eno-opravilnih operacijskih sistemov prištevamo zgodnje in namenske operacijske sisteme (npr. DOS), vsi sodobni splošno-namenski operacijski sistemi pa so več-opravilni (npr. Unix, Linux, Windows, Mac OS).

Večina sodobnih operacijskih sistemov je tudi *več-uporabniških (angl. multi-user)*. To pomeni, da lahko v nekem trenutku računalnik uporablja več uporabnikov. Tipični predstavniki zgodnjih več-uporabniških operacijskih sistemov

so bili operacijski sistemi centralnih računalnikov (npr. VAX VMS, IBM zOS, Unix), prav tako pa so več-uporabniški tudi vsi sodobni operacijski sistemi, ki omogočajo oddaljen dostop do računalnikov (npr. Linux, Windows). Vsi več-uporabniški operacijski sistemi so hkrati tudi več-opravilni, ne velja pa nujno obratno (npr. Windows Mobile, Android, iOS).

Opravila lahko izvajamo zaporedno – eno naenkrat in v tem primeru govorimo o *zaporedni, paketni (angl. batch processing) obdelavi*. Od tod tudi ime *zagonskih (angl. batch; .BAT) datotek*, ki vsebujejo sistemska navodila (ukaze) za izvajanje zaporedja opravil. Več-opravilna obdelava implicira navidezno več-procesno izvajanje z *dodeljevanjem časovnih rezin (angl. time-sharing)* ali dejansko *vzporedno izvajanje opravil na vzporednih procesorjih (angl. multi-processing)*. Seveda se pojavi pri več-opravilnem izvajanju problem upravljanja računalniških virov, saj si jih procesi med sabo delijo. Ti so omejeni – zato nastopi t.i. problem *filozofov pri večerji* (angl. dining philisophers problem; Slika 44). Vsak od filozofov pri mizi potrebuje dva kosa pribora, da bi lahko jedel špagete – če hočejo vsi jesti hkrati, je tega očitno premalo, zato je potreben dogovor. Naloga operacijskega sistema je seveda, da poskrbi, da ne pride do konflikta med procesi in se le-ti lahko nemoteno izvedejo (”filozofi najedo”).

Ko več procesov dostopa do istega vira hkrati, potrebujemo posredovanje arbitra – podobno kot pri vodilu. Vir je lahko prost ali zaseden in ga lahko zaseda en ali več procesov. Običajno arbiter tvori čakalno vrsto procesov, ki skušajo dostopati do vira, pa ta zaradi zasedenosti ni bil na voljo. Ta vrsta je lahko običajna *FCFS oz. FIFO* (angl. First Come First Served oz. First-In-First-Out ali ”kdor prvi pride prvi melje”), obstajajo pa tudi različice, ki upoštevajo prioritete procesov. Splošen postopek je takšen: ko vir potrebujemo, ga *zasedemo (angl. allocate)* in ga po uporabi ponovno *sprostimo (angl. disallocate)*. Če je vir, ki ga potrebujemo, zaseden, počakamo v vrsti, dokler se ne sprosti, da ga lahko zasedemo. Primeri tipičnih virov: disk, tiskalnik, mrežna kartica, zvočna kartica, datoteka, ki jo urejamo, zapis v bazi podatkov, itd. Naloga ope-

Slika 44: *Filozofi pri večerji*
Vir: en.wikipedia.org, Benjamin D. Esham, CC-BY-SA, 2005

racijskega sistema je, da poskrbi, da ne pride do situacije, ko bi pasivni proces *blokiral* izvajanje aktivnega zaradi konflikta virov, saj bi ta zaradi tega lahko na vir čakal neomejeno dolgo – temu fenomenu pravimo *smrtni objem (angl. dead-lock)*.

*Upravljanje s pomnilnikom (angl. memory management)* nam omogoča nalaganje programov iz sekundarnega pomnilnika v primarni in izvedbo le-teh. Zaradi velikosti programov, večkrat ni možno naložiti celotnega programa v delovni pomnilnik. Da pa bi vseeno lahko obdelovali takšne programe, je treba najprej poskrbeti za dovolj velik naslovni prostor, da bomo lahko naslovili vsako lokacijo programa, potem pa še poskrbeti za avtomatski prenos ustreznih delov programa v delovni pomnilnik za obdelavo. Pri tem nastopi težava preslikave logičnih (v glavnem pomnilniku) v fizične naslove (v primarnem pomnilniku), saj je logični naslovni prostor mnogo večji od fizičnega, pri kopiranju dela sekundarnega pomnilnika v primarnega, pa se naslovi spremenijo. Zato moramo vedno vedeti, kateri del logičnega naslovnega prostora je trenutno tudi fizično dostopen – govorimo o *blokih (angl. block)* in *straneh (angl. page)* pomnilnika. Ko nastopi situacija, da logični naslov ni fizično dostopen, moramo prenesti ustrezno stran oz. blok iz sekundarnega pomnilnika v primarnega in pri tem eventualno zamenjati obstoječo stran oz. blok spomina, preden jo lahko uporabimo. Ta operacija je seveda zamudna in je zato naloga dobrega upravljavca pomnilnika, da jo izvaja na tak način, da so čim redkejše.

Da bi po potrebi lažje (spet) našli odložene dele programa, je bil definiran t.i. *izmenjalni (angl. swap) pomnilnik* – gre za del sekundarnega pomnilnika, ki ga uporabimo, ko se delovni pomnilnik prenapolni z naloženimi stranmi. Tukaj so lahko strani urejene tako, da jih pri (ponovnem) nalaganju hitreje najdemo in enostavneje naložimo. Pri večini sodobnih operacijskih sistemov imamo t.i. *swap* particijo, ki se nahaja na delu diska, ki ga lahko najhitreje dostopamo. Ta pomnilniški prostor lahko uporabljamo kot razširitev delovnega pomnilnika, saj je še vedno precej hitreje dosegljiv kot preostali pomnilniški prostor na drugih medijih in ima glede dostopa (branje, pisanje, brisanje) enake karakteristike kot delovni pomnilnik (RAM).

Pri *krmiljenju perifernih naprav* operacijski sistem ne sme čutiti razlike med narečji različnih istovrstnih naprav. Za vse tiskalnike na primer obstajajo enake vzhodno-izhodne operacije, ki se glede na tehnologijo tiskanja na nivoju naprave prevedejo v atomarne ukaze za izvedbo. Delno tu gre za ukaze, ki neposredno krmilijo mehanizem za odtis pik, črt ali tudi črk na papirju. Ti so praviloma zakodirani v *strojni programski opremi (angl. firmware)* in shranjeni v ROM pomnilnikih perifernih naprav. Na drugi strani pa imamo krmilne ukaze, ki z računalnika naložijo vsebino za odtis in določajo, kje in kako naj bo leta odtisnjena. Temu sledi izpisovanje, sestavljeno iz zaporedij prej omenjenih atomarnih ukazov. Krmilni ukazi so na nivoju operacijskega sistema poenoteni v obliki t.i. *gonilnikov naprav (angl. device-driver)*. Nekoliko poenostavljeno bi lahko rekli, da operacijski sistem gonilniku enostavno posreduje navodilo za izpis, ta pa ga nato interpretira na način, ki je lasten napravi (npr. določenemu tiskalniku). Za gonilnike velja, da so edina programska oprema, ki neposredno komunicira s strojno opremo. Praviloma je ob namestitvi nove strojne opreme zanjo treba v operacijskem sistemu namestiti tudi ustrezen gonilnik (npr. Windows), ni pa to nujno (npr. Linux gonilnike že vsebuje in se ob priključitvi določene naprave le ustrezno konfigurirajo).

*Uporabniški vmesnik* operacijskega sistema je lahko glede na izvedbo zelo raznolik. V splošnem ločimo tri osnovne tipe uporabniških vmesnikov:

1. znakovne vmesnike
2. grafične vmesnike
3. govorne vmesnike

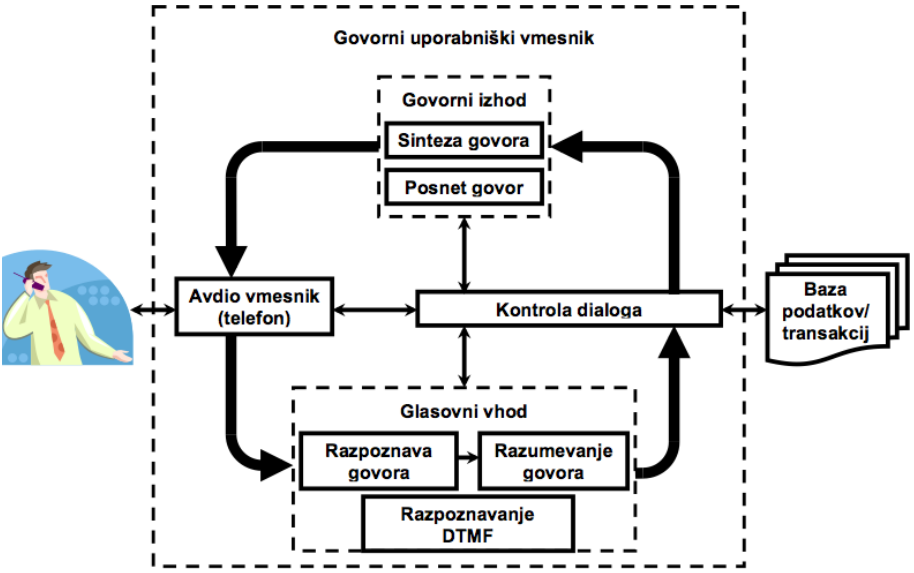
Pri *znakovnih vmesnikih (angl. character based interface)* gre za različice uporabniških vmesnikov, ki so bili in so še v uporabi pri večjih centralnih računalnikih s terminalskim (oddaljenim) dostopom, pri osebnih računalnikih pa je tak uporabniški vmesnik imel operacijski sistem DOS (angl. Disk Operating System). Značilno zanje je, da uporabnik z operacijskim sistemom komunicira preko vnosa znakov (črk, številk in simbolov ASCII abecede).

Pri *grafičnih vmesnikih (angl. Graphical User Interface; GUI)* običajno uporabljamo *kazalno napravo (angl. pointing device)*, na primer računalniško miško, risalno tablico, sledilno kroglico, ipd., s pomočjo katere izbiramo elemente računalniškega sistema in izvajamo želene akcije nad njimi. Značilni elementi grafičnega uporabniškega vmesnika so: okna, ikone, meniji, gumbi, itd. Dandanes jih pogosto lahko že izbiramo tudi neposredno na zaslonu računalnika (npr. tablični računalniki, mobilni telefoni).

Pri *govornih uporabniških vmesnikih (angl. voice user interface)* računalnik mora razpoznavati govorne ukaze in se na njih ustrezno odzvati. Tipični primeri uporabe takšnih računalniških sistemov so: informacije o voznih redih, rezervacije (npr. hoteli), nakup (vozovnice, vstopnice, ipd.), direktno bančništvo (npr. bankomat, ATM), upravljanje navigacijskega sistema in/ali klimatske naprave avtomobila, ipd. Operacijski sistem mora za govorno komunikacijo, za razliko od grafičnega uporabniškega vmesnika ali tudi kot dodatek k njemu, uporabniku nuditi še funkcije razpoznave in sintetiziranja govora (Slika 45).

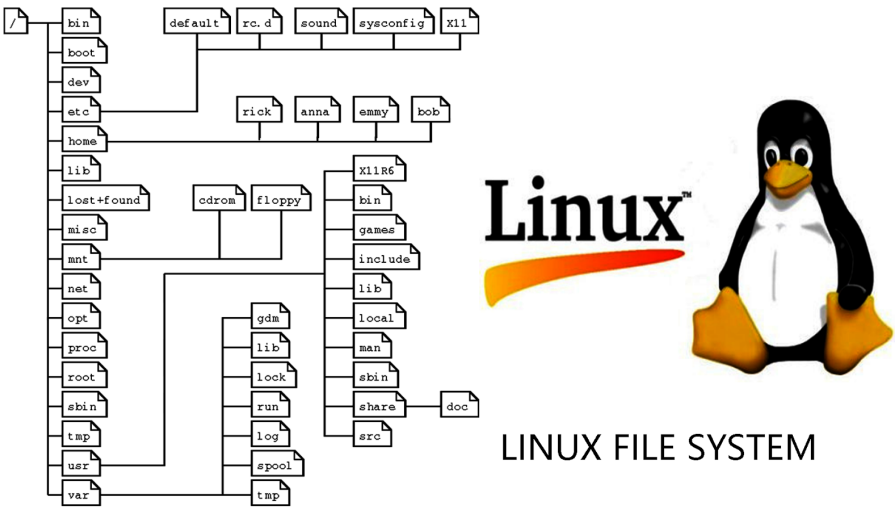
Govorni uporabniški vmesnik je najzahtevnejši (spol, narečja, način govora,...) in je zato doslej našel svojo uporabo le v namenskih aplikacijah. Z govorno komunikacijo lahko običajno krmilimo le posamezne funkcije aplikacijskih programov, ne pa tudi celotnega nabora funkcij operacijskega sistema.

*Delo z datotekami* (angl. file management) predstavlja tisti del operacijskega sistema, ki skrbi za podatke v sekundarnem pomnilniku. Vloga operacijskega sistema pri tem delu je, da zagotovi korektno in učinkovito shranjevanje podatkov na zunanje pomnilne medije in dostavo podatkov z njih. Tega smo se že dotaknili, ko smo govorili o upravljanju s spominom in navideznim pomnilniškim prostorom, vendar je v tistem delu šlo za razširitev primarnega pomnilnika – torej za podatke, ki jih ob izklopu računalnika praviloma izgubimo. Pri delu z datotekami upravljamo trajno shranjene podatke, ki se nahajajo na magnetnih in optičnih pogonih.




Slika 45: *Govorni uporabniški vmesnik*
Vir: lasten

Podatki v sekundarnem pomnilniku so organizirani v obliki *datotečnega sistema (angl. file system)*. Obstaja več različnih standardnih datotečnih sistemov (npr. FAT, NTFS, ext, itd.), ki med sabo praviloma niso kompatibilni, vendar sodobni operacijski sistemi vsebujejo module za delo z več datotečnimi sistemi, tako da prenašanje podatkov na različnih pomnilniških medijih ali celo med različnimi računalniškimi sistemi običajno ni problem. Datotečni sistemi logično shranjujejo datoteke (podatkovne zbirke) v drevesni strukturi imenikov (npr. Slika 46), da jih lahko hitreje najdemo, fizično pa so datoteke shranjene na določenem cilindru/sektorju/sledi in od tam jih tudi prečitamo, ko jih nalagamo za obdelavo v primarni pomnilnik. Programski moduli izvajajo preslikavo med logično in fizično lokacijo datotek na disku. Korenski imeniki posameznih datotečnih struktur predstavljajo t.i. *particije (angl. partition)*, ki lahko nosijo imena logičnih pogonov (npr. pri Windows A:, B:, C:, ...) ali pa so oštevilčene (npr. Linux /hda1 (IDE diski), /hda2, /sda1 (SCSI diski), sdb2,...).




Slika 46: *Primer hierarhije datotečnega sistema*
Vir: https://studyreadeducate.com/linux-file-system-shell-scriptsfileskernel/

*Datoteka (angl. (data)file)* je osnovna enota za shranjevanje podatkov v sekundarnem spominu (običajno na disku). Vsaka datoteka ima nekatere osnovne lastnosti: ime, končnico (karakterizira vsebino) in pot (lokacija v imeniški strukturi). Večina datotečnih sistemov loči dva osnovna tipa datotek: tekstovne (vsebujejo besedilo) in binarne (vsebujejo slike, glasbo, formatirano besedilo, programe, ...). Medtem ko za urejanje tekstovnih datotek praviloma ne potrebujemo posebnih programov (vsak operacijski sistem vsebuje preprost urejevalnik znakovnih datotek), za urejanje binarnih datotek praviloma potrebujemo aplikacijske programe, ki znajo delati s podatki v določenem formatu zapisa (predvajati/urejati glasbene datoteke, urejati dokumente, slike, ipd.). Za urejanje binarnih datotek torej praviloma potrebujemo dodatne programe, ki niso del operacijskega sistema. V večuporabniškem okolju imajo datoteke običajno še dodatne lastnosti: lastništvo datoteke (kateremu uporabniku pripadajo), do-

stopne pravice (branje, zapisovanje, izvajanje), dodatne informacije (arhiviranje), čas dostopa in spremembe datoteke (zaradi inkrementalnega arhiviranja) ter velikost (fizična velikost na disku).

Nalaganje in zagon programov seveda implicira prej opisane funkcije za delo z datotečnim sistemom – iskanje in prenos (delov) programske datoteke v delovni pomnilnik, upravljanje pomnilnika (upravljanje programov in podatkov v delovnem pomnilniku ter njihovo (sprotno) shranjevanje na disk) ter v končni fazi še upravljanje s procesi (dodeljevanje procesorja procesom aplikacijskih programov v izvajanju). Nekatere programe lahko zaženemo sami preko ukazov ukazne vrstice, s klikom na ikono ali z izrecno zahtevo računalniku, po potrebi pa programi za izpolnjevanje zahtevanih funkcij lahko sami zaženejo tudi druge programe in med sabo izmenjujejo podatke, ki so potrebni za obdelavo njihovih ukazov.

Zaradi oddaljenega dostopa do podatkov in računalniških resursov, pa tudi zaradi distribuirane obdelave podatkov je postala *funkcionalnost mreže in dostopa do interneta* pomemben del operacijskega sistema. S krmiljenjem mrežnih perifernih naprav je povezana dodatna vrsta gonilnikov. Dodatno pa je z upravljanjem omrežij povezanih še več sistemskih uporabnih programov, ki implementirajo ustrezne komunikacijske protokole in omogočijo enolično identifikacijo računalnikov v mreži. Več o tem smo podali v poglavju o *računalniški komunikacijski opremi* (Poglavje 4).

### 5.2.2 Sistemski uporabni programi

Med sistemsko programsko opremo razen operacijskega sistema prištevamo še druge programe, ki so običajno del t.i. *distribucij operacijskih sistemov*, kjer v paketu razen OS prejmemo še t.i. *uporabne programe (angl. utility programs)*. V grobem jih lahko razvrstimo v naslednje kategorije:

1. *Uslužnostni programi (angl. utilities)* opravljajo različne koristne funkcije (npr. prevajanje programov v različnih programskih jezikih, varnostno kopiranje podatkov, protivirusno zaščito, komprimiranje podatkov, itd.)

2. *Nadzorniki performans (angl. performance monitors)* so programi, ki omogočajo nadzor in nastavitve računalniškega sistema s ciljem normalnega (optimalnega) delovanja

3. *Nadzorniki varnosti (angl. security monitors)* so programi, ki spremljajo in omogočajo nadzor uporabe računalniškega sistema, z namenom, da bi optimizirali izrabo in onemogočili neavtorizirano izkoriščanje resursov

4. *Aplikacijski strežniki (angl. application servers)* predstavljajo programsko opremo, ki "prinaša" aplikacijo na računalnik klienta (večinoma preko HTTP protokola) in omogoča njeno delovanje preko Interneta (spletne aplikacije)

5. *Posredniška programska oprema (angl. middleware)* pomaga različnim aplikacijskim programom pri izmenjavi podatkov in skupnem delovanju

Med sistemsko programsko opremo prištevamo še *programska razvojna orodja in okolja (angl. software development tools and enivironments)*, kamor sodijo prevajalniki programskih jezikov in pomožna programska orodja, ki programerjem olajšajo delo. Sodobna programerska orodja običajno vključujejo vizualna *integrirana razvojna okolja (angl. Integrated Development Environment, IDE)*, ki programerjem pomagajo zasnovati programe, jih prevesti v izvedljivo obliko za ciljno platformo, optimizirati njihovo zmogljivost in *pomnilniški odtis (angl. memory footprint)*, pa tudi minimizirati možnost/pogostost pojavljanja napak pri njihovem razvoju. Sem sodijo:

1. *Grafična programska okolja (angl. Graphical Programming Interfaces)* integrirajo načrtovanje grafičnih uporabniških vmesnikov s programiranjem programskih funkcij in običajno vključujejo tudi dostop do podatkovnih baz

2. *Programski urejevalniki (angl. Programming Editors)* so običajno del *integriranih razvojnih okolij*, lahko pa so tudi samostojni; v osnovi so namenjeni kreiranju, urejanju in shranjevanju izvornih programov, lahko pa tudi izboljšajo njihovo berljivost s pomočjo *označevanja sintakse (angl. syntax highlighting)* in ustreznim zamikanjem programskih ukazov, ki bolje odraža strukturo programa

3. *Razhroščevalniki (angl. Debuggers)* so programi, ki omogočajo programerjem najti mesto programske napake s sledenjem skozi izvajanje programa in/ali spremljanjem sprememb vrednosti ustreznih spremenljivk

Za razvoj programske opreme so bile razen omenjenih integriranih razvojnih okolij razvite tudi metodologije, ki preko smernic in metod podpirajo sistematičen pristop k izdelavi programske opreme – *programsko inženirstvo (angl. software engineering)*. Programerska orodja, ki podpirajo omenjene metode, s kratico imenujemo *CASE orodja (angl. Computer-Aided Software Engineering tools)*. Predstavljajo nadgradnjo integriranih razvojnih okolij pri razvoju in vzdrževanju programske opreme – načrtovanje, urejanje, prevajanje, povezovanje, razhroščevanje, spremljanje verzij, ustvarjanje varnostnih kopij, itd.).

**Primer**: UML CASE orodja tipično vključujejo

1. Orodja za modeliranje podatkov (angl. data modeling tools)

2. Orodja za avtomatsko generiranje kode (angl. code generation tools) in

3. Integrirana razvojna okolja za posamezne programske jezike in ciljne operacijske sisteme

## 5.3 Aplikacijska programska oprema

Kot uporabnik se z operacijskim sistemom srečujemo večinoma le posredno – uporabljamo ga za zagon aplikacijskih programov (aplikacij), ki nam omogočajo izvrševanje različnih nalog s pomočjo računalnika. Glede na vrsto teh nalog ločimo različne vrste aplikacijskih programov.

### 5.3.1 Splošna aplikacijska programska oprema

Gre za programsko opremo, razvito pod COTS ali odprto-kodno licenco, namenjeno splošni uporabi oz. različnim uporabnikom. Običajno med splošno aplikacijsko programsko opremo prištevamo:

1. Spletne brskalnike
2. Programe za e-pošto, takojšnja sporočila, Web-log ali Blog programe
3. Programe za obdelavo besedil in namizno založništvo
4. Programe za izdelavo preglednic in predstavitev
5. Upravljavce osebnih podatkov
6. Programe za skupinsko delo...

*Spletni brskalniki* (angl. browser) so programi, namenjeni brskanju (iskanju informacij) po spletu. Postali so univerzalna programska platforma tudi za spletne aplikacije. Tipični predstavniki takšnih programov so na primer: Internet Explorer, Edge, Chrome, Opera, Safari, Mozilla Firefox,...

*E-poštni programi* so programi za pošiljanje/prejemanje e-poštnih sporočil po internetu. V osnovi gre pri elektronski pošti za tekstovna sporočila. Za pošiljanje hipermedijskih sporočil in binarnih priponk je bil razvit standard *MIME (Multipurpose Internet Mail Extension)*. E-poštni programi praviloma omogočajo komunikacijo tako z Intranetom kot z Ekstranetom. To so na primer: Outlook Express, Mozilla Thunderbird, Microsoft Outlook, Lotus Notes,...

S pojavom socialnih omrežij se je razmahnila uporaba t.i. programov za *takojšnja sporočila (angl. Instant Messaging, IM)*. Ti programi omogočajo trenutno (takojšnjo) komunikacijo preko e-sporočil, ki so tipično krajša od e-pošte in neformalna (kot npr. SMS (angl. Short-Message-Service) v GSM omrežjih).

Za prenos govornih in video sporočil ti programi uporabljajo *telefonijo preko internetne povezave (angl. Voice over IP, VoIP)*. Primeri takšnih programov so: IRC, Empathy, Skype, Signal, Viber, WeChat,...

Sodelovanje podpirajo tudi *Web-log ali blog programi*. Gre za *spletne programe*, ki nam olajšajo oblikovanje in hrambo lastnih spletnih strani v obliki osebnega dnevnika in tipično vsebujejo sveže informacije o neki temi ali sklopu tem, ki jih raziskujemo in lahko zanimajo druge. V sodobnem času ti spletni dnevniki lahko vsebujejo tudi video posnetke – zato večkrat govorimo tudi o *video-logih oz. vlogih*.

*Programi za obdelavo besedil (angl. word processors)* nam omogočajo ustvarjanje, urejanje, pregledavanje in izpis dokumentov. Običajno tu ne gre le za tekstovna besedila, ampak za formatirane multimedijske ali hipermedijske dokumente, ki so shranjeni v binarni obliki. Tipični predstavniki so: Microsoft Word, Lotus WordPro, OpenOffice Writer,...

*Programi za namizno založništvo (angl. desktop publishing)* nam omogočajo izdelavo materialov, ki izgledajo kot profesionalno oblikovani dokumenti, pripravljeni za izpis/tisk ali objavo na spletu. Tipični predstavniki so: Adobe PageMaker, Microsoft Publisher, QuarkXPress,...

*Elektronske preglednice (angl. spreadsheets)* so programi, ki nam omogočajo hrambo številskih podatkov in avtomatizirane izračune ter grafične predstavitve le-teh. Podatki in navodila za izračune so shranjeni v celicah delovnih listov, razdeljenih na vrstice in stolpce (podobno kot v delovnem pomnilniku, ki prav tako hrani podatke in ukaze). Tipični predstavniki so: Microsoft Excel, Lotus 1-2-3, OpenOffice Calc,...

*Predstavitveni programi (angl. presentation managers)* so programi, ki omogočajo multimedijske predstavitve, vključujoč grafiko, slike, animacije in video posnetke. Tipični predstavniki so: Microsoft PowerPoint, Lotus Freelance, OpenOffice Impress,...

*Upravljavci osebnih podatkov (angl. Personal Information Manager, PIM)* so programi za končne uporabnike, ki so namenjeni večji produktivnosti in lažjemu sodelovanju. Gre za elektronsko različico rokovnikov oz. terminerjev, ki lahko hranijo podatke o strankah, sestankih, opravilih, časovnice,... Tipični predstavniki so: Lotus Organizer, Microsoft Outlook, Evolution,...

Večji produktivnosti in intenzivnejšemu sodelovanju pri projektih so namenjeni tudi *programi za skupinsko delo (angl. Groupware)*. Ti programi podpirajo delovne skupine pri sodelovanju na nalogah in nudijo poenoteno e-pošto, forume, podatkovne baze, medsebojne video-konference, ipd. Tipični predstavniki so: Lotus Notes, Novell GroupWise, Microsoft Exchange, Microsoft Teams,...

### 5.3.2 Uporabniška programska oprema

Namenske aplikacijske programske opreme ne bomo posebej opisovali, saj gre po eni strani za specifične programe, ki služijo točno določenemu namenu in so "pisani na kožo" manjši skupini uporabnikov, po drugi strani pa je njihova zgradba z uporabniškega vidika običajno enostavno dojemljiva. Omogočajo avtomatizacijo določenih nalog oz. funkcij teh uporabnikov znotraj organizacije. Njihova uporaba je običajno omejena na vnos podatkov, njihovo hranjenje, procesiranje ter oblikovanje prikazov rezultatov v obliki ekranskih slik ali pisnih poročil. Tipični predstavnik, ki ga študenti dobro poznate je AIPS. Čeprav je študentska populacija številčna, vseeno gre za uporabniško programsko opremo, namenjeno le vam in samo vaši izpitni evidenci.

### 5.3.3 Programska okolja

Uporaba v tipskih poslovnih okoljih in aplikacijah je povzročila povezovanje aplikacijskih programov v *programska okolja (angl. application framework)*. To so *zbirke aplikacijskih programov*, ki jih povezujejo naslednje lastnosti:

1. skupna funkcionalnost
2. podobni uporabniški vmesniki
3. vgrajene možnosti za enostavno medsebojno izmenjavo podatkov

Tipične predstavnike programskih okolij iz skupine splošne aplikacijske programske opreme smo že omenili. Sem sodijo na primer programska okolja, ki združujejo programe za avtomatizacijo pisarniškega poslovanja – t.i. "Office" programski paketi.

Uporabniška programska oprema za vodenje poslovanja ima običajno prav tako omenjene lastnosti programskih okolij. Sem sodijo na primer ERP sistemi, študentske evidence oz. AIPS, ipd.

Zlasti pri poslovni programski opremi je težko potegniti mejo ločnico med COTS in uporabniško programsko opremo, saj večkrat gre za komercialno programsko opremo, ki jo ponudniki konfigurirajo (ukrojijo) za potrebe končnega uporabnika.

## 5.4 Spletni programi in servisi

Od nastanka Interneta se je pojavila naraščajoča potreba po učinkoviti spletni predstavitvi vsebin. Zaradi potrebe po stalnem posodabljanju in aktualizaciji vsebin ter dvosmerni komunikaciji, so bili razviti t.i. spletni programski jeziki. Gre za interpretirane programske jezike, katerih ukazi se interpretirajo ob prikazu spletnih strani.

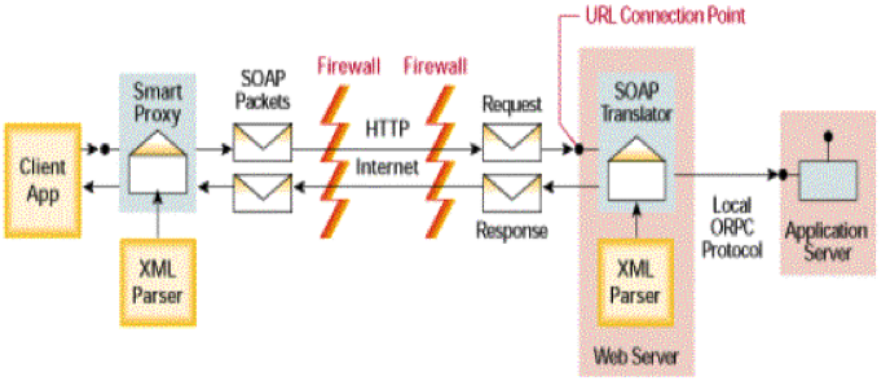
Dokumenti za splet so tekstovne datoteke, ki so kodirane na poseben način, ki enolično definira njihovo vsebino in strukturo (glava, naslov, odstavek,...). To strukturo vsi spletni brskalniki interpretirajo na enak način zato govorimo o interpretiranih programskih jezikih. Vsi aktualni jeziki za oblikovanje spletnih dokumentov imajo svoj izvor v *jeziku HTML* (HyperText Markup Language). Ker tu v osnovi ne gre za programe v klasičnem smislu besede, ampak za

hipermedijske dokumente, pogosto uporabljamo tudi pojem *označevalni jeziki (angl. markup language)*. Ker pa lahko v hipermedijsko besedilo vstavimo tudi kodo drugih *skriptnih programskih jezikov (angl. scripting programming language)*, kot sta a primer JAVA script in PHP, ki so interpretirani, jih vseeno uvrščamo v širšo skupino interpretiranih programskih jezikov.

Svetovni splet nam omogoča izmenjavo različnih podatkov na Internetu. Spletne strani so pripravljene v označevalnih jezikih, ki jih *spletni brskalniki (angl. browserji)*, znajo interpretirati. To so na primer različice HTML jezika, uveljavljajo pa se tudi njegove razširitve XHTML (razširitev z XML), VRML (navidezna resničnost), ipd. Za prikaz dinamičnih spletnih strani so dodatno v uporabi različni skriptni programski jeziki JSP, ASP, Perl, CGI skripte, JAVA Script, ipd., ki nam omogočajo (kontekstno odvisen) prikaz vsebin na zahtevo. Pri tem kontekst predstavlja vrsto in identiteto odjemalca, številko seje, ipd.

Medtem ko s HTML, XHTML, VRML ipd. označevalnimi jeziki tvorimo formatirane hipermedijske dokumente, nam JAVA script, PHP, ipd. spletni programski jeziki omogočajo zajem, procesiranje in posredovanje informacij preko spleta. V primeru PHP (*Hypertext Preprocessor*) gre za skriptni jezik, katerega ukaze preprosto vgradimo na ustrezna mesta v HTML dokumentu. Njihova interpretacija nam omogoča izvajanje ukazov (npr. delo s podatkovno bazo) na spletnem strežniku. Pravimo da *spletni strežnik (angl. Web server)* implementira določen *spletni servis (Web service)*. Podobno velja za JAVA script ukaze, vendar njihova obdelava poteka na strani brskalnika. Pri tem se lahko tudi sklicujemo na določen spletni servis, ki mu prenesemo parametre povpraševanja lokalnega brskalnika (npr. povpraševanje po vremenu pri vremenskem spletnem strežniku na določeni lokaciji). Obe varianti imata prednosti in slabosti – zato se glede uporabe odločamo na podlagi spletne aplikacije, ki jo želimo implementirati.

Za fleksibilen način izmenjave informacij preko spleta je bilo treba oblikovati format, ki ne bo odvisen od predstavitve podatkov v tej ali oni podatkovni bazi sodelujočih spletnih strežnikov, pa bo vseeno nedvoumno določal vsebino podatkovnih sporočil med njimi. Najbolj razširjen format *XML (Extensi-*




Slika 47: *Izmenjava EDI sporočil preko SOAP protokola*
Vir: Nandrajog (2001)

*ble Markup Language),* ki je postal standard za izmenjavo vseh vrst informacij preko spleta, je definiral in vzdržuje World Wide Web Consortium (W3C). Zraven novejšega *JSON (JavaScript Object Notation)* formata je najširše uporabljen standard za *elektronsko izmenjavo podatkov (angl. Electronic Data Interchange, EDI)* po spletu.

Kaj so torej *spletni servisi*? To so programske komponente, zasnovane na spletni in objektni paradigmi programiranja ter tehnologijah za elektronsko povezovanje različnih uporabnikov in platform. Zaradi boljšega medsebojnega delovanja spletnih servisov je bil zasnovan projekt *UDDI (Universal Description, Discovery and Integration)*. Gre za repozitorij procesov in pripadajočih spletnih servisov. Podpira *SOAP protokol (Simple Object Access Protocol oz. Service Oriented Architecture Protocol)* za izmenjavo XML sporočil v računalniških omrežjih – neko vozlišče (klient) pošlje zahtevo drugemu vozlišču (strežnik), ki mu nato odgovori z rezultatom povpraševanja (Slika 47) na enak način.

Ker gre pri spletnih aplikacijah običajno za tipske konfiguracije (operacijski sistem, spletni strežnik, brskalnik, podatkovna baza, spletišče), ki morajo med sabo tesno sodelovati, se je pojavil pojem t.i. *spletne platforme (angl. Web platform)*, kot je na primer LAMP (Linux-Apache-MySQL-PHP). Zainteresirani bralec lahko širšo razlago in povezave na dodatne informacije dobi na spletu.

## 5.5 Programski jeziki, algoritmi, podatkovne strukture

Sedaj je že čas, da povemo nekaj več tudi o *programskih jezikih (angl. programming language)*. Omenili smo že *strojni jezik (angl. machine code)*, sestavljen iz zaporedij 1 in 0 za ukaze, ki jih procesorji lahko neposredno interpretirajo, in njegovo človeško-berljivo (angl. human-readable) različico – *zbirni jezik (angl. assembly language)*.

Sodobni programi so prekompleksni, da bi jih lahko sprogramirali zgolj v zbirnem jeziku, katerega kompleksnost je s kompleksno strukturo in možnostmi sodobnih procesorjev prav tako še rasla. Zato sodobne programe pišemo v *visokonivojskih programskih jezikih (angl. high-level programming language)* in jih nato prevedemo v strojni jezik za izvedbo na konkretnem procesorju oz. platformi.

Nekatere (zlasti spletne) aplikacije ne potrebujejo prevajanja in jih lahko operacijski sistem in brskalnik enolično interpretirata na katerikoli platformi – te lahko zapišemo v interpretiranih skriptnih in označevalnih programskih jezikih.

**Razvoj programskih jezikov** Programski jeziki so razdeljeni v generacije, kot so se razvijali:

1. strojni jezik (1. generacija) – ukazi sestavljeni iz 0 in 1

2. zbirni ali simbolični jezik (2. generacija) – ukazi predstavljeni z zbranimi logičnimi simboli (npr. “ADD, MOVE,. . . ”)

3. visokonivojski programski jeziki:

   (a) 3. generacije:

      i. strukturni (Fortran, Basic, Pascal, C, Cobol, idr.)

      ii. objektni (C++, Java, C#)

   (b) 4. generacije:

      i. povpraševalni jeziki (SQL, DL Query, SPARQL,. . . )

   (c) 5. generacije:

      i. še niso v celoti razviti (umetna inteligenca, naravni jezik,. . . )

4. skriptni jeziki:

   (a) JavaScript, PHP, Perl, VBScript, Python

5. označevalni jeziki:

   (a) HTML, XHTML, XML, VRML

Visoko-nivojski programski jeziki so bili zasnovani zaradi naraščajoče (strukturne) kompleksnosti aplikacijskih programov. Gre za programske jezike, ki jih s pomočjo prevajalnikov prevedemo v zbirni oz. strojni jezik (običajno gre za dvostopenjski postopek prevajanja in povezovanja v izvedljivo celoto). Vsak ukaz višjega programskega jezika se torej prevede v več ukazov zbirnega jezika skladno z uporabljeno različico ukaza, tipom parametrov in kontekstom, v katerem je uporabljen. Kot smo že omenili, gre tukaj zraven obvladovanja funkcionalne kompleksnosti tudi za obvladovanje strukturne kompleksnosti aplikacij. V ta namen je bil najprej razvit strukturni način programiranja, nato pa še objektni način programiranja, skladno z naravo problemov oz. aplikacij, ki jih je mogoče z njimi realizirati.

Skriptni in označevalni jeziki so bili razviti za potrebe spletnih aplikacij in jih ne moremo uvrstiti v že omenjene generacije, saj so se razvili posebej zaradi potrebe po čim boljši predstavitvi statičnih vsebin (označevalni jeziki) in dinamičnih vsebin (skriptni jeziki) spletnih strani.

Zaradi potrebe po shranjevanju kompleksnih podatkovnih struktur so razvili podatkovne baze. Vzporedno z visokonivojskimi programskimi jeziki so zato nastajali povpraševalni jeziki, ki so tudi lahko strukturirani, in se podobno kot skriptni jeziki interpretirajo v določenem programskem okolju (npr. upravljavca podatkovne baze, baze znanja, ipd.). Eden najeminentnejših predstavnikov povpraševalnih jezikov za delo s podatkovnimi bazami je SQL (Structured Query Language). Ukaze SQL lahko izvajamo iz programov, napisanih v višjih programskih ali skriptnih jezikih, preko *aplikacijskega programskega vmesnika (angl. Application Programming Interface, API)* programskega upravljavca s podatkovno bazo.

Razvoj programskih jezikov ni zastal, ampak se razvija še naprej v smislu razumevanja govornih ukazov – naravnega jezika. Pri programiranju gre vedno za to, da želi človek računalniku prenesti navodilo za izvedbo določene naloge. Glavna težava pri programiranju je dovolj natančno in nedvoumno formulirati ta navodila, da bi jih računalnik lahko pravilno interpretiral. Ker je pri tem najbolj problematična dvoumnost človeških navodil, saj je vezana na kontekst, raziskave potekajo v smeri razumevanja nenatančno formuliranih ukazov. Določen napredek je bil že dosežen na primer pri krmiljenju multimedijskih sistemov v avtomobilih za udobje in asistenco pri vožnji (t.i. info-tainment sistemi).

**Programski jezik** Sedaj že imamo dovolj jasno sliko o tem, kaj programski jezik je, da lahko postavimo definicijo:

***Programski jezik*** *omogoča pisanje programov, ki jih lahko izvajamo na računalniku.*

Zaradi velike raznolikosti strojne opreme, zlasti procesorjev, računalniški programi v splošnem niso prenosljivi med različnimi tipi računalnikov oz. platformami (tip procesorja, operacijski sistem, spletna platforma). Do neke mere so računalniški programi v izvorni obliki prenosljivi, medtem ko so njihovi prevajalniki za vsako platformo različni.

Programski jezik sestavljajo naslednje komponente:

1. sintaksa (določa ukaze, podatkovne tipe)
2. semantika (določa pravila uporabe ukazov)
3. programske knjižnice (dostopne preko ukazov iz njihovih API)

Zlasti pri programskih knjižnicah pride do izraza raznolikost platform. Omejitev pri prenosljivosti programov pogosto predstavlja odsotnost programskih knjižnic za določene platforme, ali pa omejitve v njihovi rabi, ki lahko izvirajo tudi iz omejitev strojne opreme.

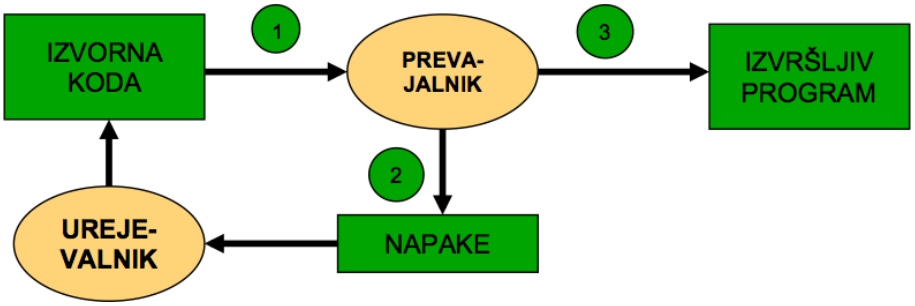
Programske jezike v splošnem delimo na:

1. prevajane (izvedba v strojnem jeziku) in
2. interpretirane (izvedba po izvorni kodi)

Prevajanje praviloma zahteva dva programa:

1. *prevajalnik* (angl. compiler) in
2. *povezovalnik* (angl. linker)

Slika 48 prikazuje postopek prevajanja od *izvorne kode (angl. source code)* programa do izvedljivega oz. *izvršljivega (angl. executable) programa*. Izvorno kodo najprej uredimo (1), nakar sledi prva faza prevajanja, kjer nas




Slika 48: *Postopek prevajanja programa*
Vir: lasten

prevajalnik opozori na morebitne pravopisne (sintaktične napake). Te popravljamo z urejanjem izvorne kode (2) dokler le-ta ni sintaktično pravilna. Druga faza prevajanja (3) k naši prevedeni kodi doda še (reference na) kodo programskih knjižnic, ki jih uporablja naš program. Dobimo izvršljiv program, ki ga lahko zaženemo na računalniku. Običajno pri tem naletimo še na logične (semantične) napake, ki jih moramo prav tako popraviti s popravljanjem/dopolnitvami izvorne kode. Šele nato je naš program uporaben *aplikacijski program* oz. *aplikacija*.

Programske napake delimo na:

1. pravopisne oz. sintaktične napake
2. logične oz. semantične napake
3. izjeme (angl. exception) oz. predvidljive napake v delovanju
4. nepredvidljive napake (angl. error) v delovanju

Za razliko od prevajanja, kjer izvorno kodo prevajamo v strojni jezik računalnika, interpretirane izvorne programe ukaz po ukaz izvajajo *interpreterji*. Predpogoj za to je seveda sintaktično pravilen izvorni program. SQL, Shell-script (ukazni jezik operacijskega sistema), JAVA script in PHP (programska

jezika spletnih aplikacij) so primeri interpretiranih programskih jezikov. Iz teh primerov lahko povzamemo, da gre pri tem običajno za posamezne ukaze ali krajše programe, ki v okolju operacijskega sistema, spletnega brskalnika ali sistema za upravljanje baze podatkov izvedejo neko specifično nalogo.

**Algoritem** Pojem algoritma smo že spoznali. Glede na programske jezike, ki smo jih obravnavali malo prej, lahko rečemo, da gre za urejeno zbirko ukazov, s katerimi implementiramo določeno nalogo – aplikacijo. Glede na vrsto aplikacije je seveda ta zbirka različna – gre za preslikavo "kaj-kako" – algoritem predstavlja navodila računalniku, kako naj izvede zadano nalogo.

Vsak algoritem ima naslednje lastnosti:

- vhodi: ima vhodne parametre (ni obvezna lastnost)
- izhodi: ima rezultat(e)
- določnost: je natančno in enoumno določen
- končnost: se konča v končnem številu korakov v vseh primerih
- izvedljivost: se ga da izvesti

Od tod bi lahko izpeljali naslednjo definicijo:

***Algoritem*** *je končni nabor ukazov, ki ob izvajanju opravi določeno nalogo v končnem številu korakov.*

Sklepamo lahko, da vsak računalniški algoritem realizira neko elementarno funkcijo informacijskega sistema, ki smo jo predstavili z enačbo $I = T(V)$, kjer $V$ predstavlja vhodne podatke, $I$ izhodne rezultate $T$ pa transformacijsko funkcijo.

Glavna problema pri snovanju algoritma sta:

1. Kako zasnovati algoritem za problem, ki ni trivialno rešljiv?
2. Kako izbrati najboljšo rešitev, kadar je možnih rešitev več?

Običajno kompleksni problem razdelimo na delne (princip *deli in vladaj*), jih rešimo ločeno in rešitve sestavimo v končno rešitev. Kadar je možnih rešitev več, moramo po pregledu najdenih rešitev oceniti njihovo koristnost in izbrati najboljšo.

Algoritme lahko izrazimo/zasnujemo na različne načine:

- naravni jezik: težko zadostimo zahtevam, ki izvirajo iz lastnosti algoritma
- grafični opis: diagrami poteka, funkcijski bloki, ipd., so nazorni, vendar potrebujejo interpretacijo
- psevdokoda: pregledna, vendar je običajno ne moremo izvesti
- programski jezik: programi, izvedljivi na računalniku

Osnovne krmilne strukture, ki jih najdemo v vsakem programskem jeziku so:

- funkcija (predefinirana ali uporabniška),
- zaporedje (sekvenca),
- izbira (alternativa) in
- ponavljanje (iteracija) ukazov.

V nadaljevanju bodo najprej podane v psevdokodi, ki je dovolj nazorna za predstavitev konceptov in dovolj formalna, da jo je mogoče enostavno prevesti v sintakso vseh višjih programskih jezikov. Za ilustracijo bodo posamezni koncepti nato predstavljeni še v programskem jeziku Python.

*Zaporedje* ali sekvenca opravil ponazarja aktivnost oz. proces, ki poteka linearno (po vrstnem redu korakov):

**begin**
Korak 1
Korak 2
Korak N
**end**

**Primer**: izračun $C = \sqrt{(A^2 + B^2)}$ (Python)

```
#Knjižnica matematičnih funkcij
import math

#Izračun 3. stranice enakostraničnega trikotnika
def c_triangle(a,b):
        x=pow(a,2)
        y=pow(b,2)
        z=x+y
        return math.sqrt(z)

c=c_triangle(5,3)
print('Stranica c:',c)
```

*Izbira* ponazarja alternativne aktivnosti, ki glede na trenutno stanje pogojev usmerijo potek izvajanja procesa:

**if** pogoj **then**
Alternativni koraki 1
**else**
Alternativni koraki 2
**end**

**Primer**: ocenjevanje izpitov (Python)

```
#Preprosta ocenjevalna funkcija
def opravil(ocena):
    if ocena > 90:
        print("odlično")
    elif ocena > 50:
        print("uspešno")
    else:
        print("neuspešno")

#Prvi klic --> "odlično"
opravil(91)
#Drugi klic --> "uspešno"
opravil(51)
#Tretji klic --> "neuspešno"
opravil(45)
```

*Ponovitev* ponazarja ponavljajoče se aktivnosti, kjer se določeno (zaporedje) korakov oz. proces izvaja dokler je izpolnjen pogoj:

**while** pogoj **do**
Koraki iteracije
**end**

**Primer**: vsota števil v tabeli (Python)

```
#Izpis vsote števil v tabeli
tabela = [1,2,3,4]
i = 0
vsota = 0
while i < 4:
  vsota += tabela[i]
  i += 1
print(vsota)
```

Medtem ko lahko algoritme za preproste naloge, kot so bile naštete, zapišemo ”iz glave” in smo pri tem dokaj prepričani, da bodo tudi pravilno delovali, to pri kompleksnejših nalogah ne drži. Zato moramo v sklopu programskega inženirstva algoritme podrobno analizirati,da ugotovimo njihove lastnosti. Analiza algoritmov obsega:

- preverjanje pravilnosti (verifikacija, validacija),
- preverjanje optimalnosti (ali je možno algoritem zasnovati preprosteje ali ceneje),
- analiza prostorske in časovne zahtevnosti (koliko časa in pomnilniškega prostora potrebujemo za izvedbo nekega algoritma).

*Verifikacija* preveri, ali algoritem deluje pravilno? Pri tem preverjamo prenosno funkcijo ($T$) na osnovi vstopnih in izstopnih pogojev ($V : I$) za vsako zaporedje ukazov.

*Validacija* preverja, ali algoritem opravi to, za kar je namenjen. Pri tem se sklicujemo na specificirano funkcionalnost, na osnovi katere je nastal.

*Ocenjevanje zahtevnosti* algoritmov pomeni ocenjevanje količine računalniških resursov, potrebnih za rešitev problema po izbranem postopku. Pri tem prostorska zahtevnost predstavlja količino potrebnega pomnilnika, časovna pa število računskih operacij, ki bodo potrebne za izvedbo algoritma. Časovna $T(n)$ in prostorska zahtevnost $S(n)$ sta običajno funkciji količine podatkov. Vrsta teh funkcij določa hitrost naraščanja zahtevnosti s številom podatkov:

1. polinomska in logaritemska zahtevnost predstavljata t.i. *lahke probleme*
2. eksponentna zahtevnost predstavlja t.i. *težke probleme*

Medtem ko za polinomske algoritme praktično ni omejitev v količini obdelanih podatkov oz. je računalniška obdelava v tem primeru najbolj smotrna, so algoritmi z eksponentno zahtevnostjo učinkoviti samo pri manjših količinah

podatkov. Za večje probleme si moramo tu pomagati z metodo *deli in vladaj* ter ustrezno *hevristično funkcijo*, ki nam bo pomagala izbirati koristne rezultate, saj bi sicer hitro presegli računalniške kapacitete, ki so nam na voljo.

**Podatkovna struktura** Struktura podatkov bistveno vpliva na algoritme za njihovo obdelavo – predvideni algoritmi definirajo strukturo podatkov. Kot smo že povedali je podatek nosilec informacije. Gre za primerek iz množice elementov, ki določa neko lastnost entitete v algoritmični obdelavi. Podatkovni tip se nanaša na vrsto podatkov, ki jih hranimo, obdelujemo in posredujemo med obdelavo. Poznamo vgrajene (npr. znaki, cela in realna števila, polja, ipd.) in sestavljene podatkovne tipe (npr. seznami, drevesa, grafi, ipd.). Podatkovna struktura ni le nabor podatkov, temveč podaja tudi odnose med njimi, definirane z naborom dovoljenih operacij (npr. inicializacija, vnos, sprememba, brisanje, ipd.). Je abstrakcija informacij iz realnega sveta.

***Podatkovna struktura*** *je množica zalog vrednosti D, ki obenem določa tudi njen posamični element d, nabor funkcij F nad njimi in nabor aksiomov A, ki veljajo nad njimi. Trojica (D,F,A) določa podatkovni element d.*

Njen matematični ekvivalent so diskretne strukture, odvisno od rigoroznosti njene definicije pa algebre, grupe in kolobarji. Večinoma so podatkovne strukture v računalništvu definirane bolj ohlapno, gre pa za isti princip – podatki, operacije in omejitve, ki definirajo njihovo pravilno uporabo.

Formalna definicija podatkovne strukture:

**structure** ime (seznam parametrov, ki jo definirajo)
(* komentar *)
**begin**
**declare**
seznam definiranih funkcij
**where**
aksiomi, ki jim funkcije zadoščajo
**end**

**Primer**: podatkovna struktura Tabela

**structure** tabela (indeks, vrednost)
**begin**
**declare**
PRIPRAVI(tabela) --> tabela
VRNI(tabela,indeks) --> vrednost
VSTAVI(tabela,indeks,vrednost) --> tabela
**where**
VRNI(PRIPRAVI(A),i) ::= napaka
VRNI(VSTAVI(A,i,x),j) ::= if i=j then x else VRNI(A,j)
**end**

**Primer**: uporaba podatkovne strukture Tabela (Python)

```
#Knjižnica z definicijo tabele
from tabulate import tabulate

#PRIPRAVI(tabela)
tabela = [["Ime", "Priimek", "Starost"],
          ["Jože", "Novak", 39],
          ["Marija", "Janež", 25],
          ["Jana", "Bo", 28]]

#VRNI(tabela)
print(tabulate(tabela, headers="firstrow",
                tablefmt="grid", showindex=True))

#VSTAVI(tabela,3,["Jane", "Doe", 28])
tabela[3]=["Jane", "Doe", 28]

#Izpiši VRNI(tabela,3)
print(tabela[3])
```

Čeprav se zdi, da je navedena tabela najbolj uporabna podatkovna struktura za hrambo in obdelavo najrazličnejših podatkov, pa tudi kot osnova podatkovnih zbirk podatkovnih baz in skladišč, to ni vedno tako. V logistiki obstajajo številne aplikacije, kjer imamo razpršene podatke, ki so med sabo povezani na najrazličnejše načine. Šele z algoritmi nato lahko poiščemo njihove povezave, na podlagi katerih lahko pridemo do ustreznih rešitev oz. zaključkov. Dober primer so algoritmi za iskanje optimalne dostavne poti, iskanje najboljšega dobavitelja, ipd. V tem primeru so grafi, kot dinamična podatkovna struktura, primernejši in tudi učinkovitejši. Več informacij o zasnovi algoritmov in ustreznih podatkovnih strukturah lahko zainteresirani bralec najde na Internetu, mi se pa bomo konkretnih primerov dotaknili pri obravnavi ustreznih tematik.

S tem zaključujemo poglavje o računalniški programski opremi. Za zasnovo učinkovitih računalniških programov so enako pomembni učinkoviti algoritmi kot tudi ustrezne podatkovne strukture. Kako jih zasnovati, si bomo ogledali v poglavju o *računalniški organizacijski opremi* (Poglavje 7).

# 6 Računalniška podatkovna oprema (DW)

Da bi jih lahko preiskovali, obdelovali in posredovali, moramo vse podatke, kot so števila, črke, valute, datumske oznake, ločila, itd., shraniti v računalniškem spominu. V ta namen jih je potrebno ustrezno zakodirati. Ta proces bi lahko poimenovali *digitalizacija*, saj gre za spreminjanje podatkov iz izvirne analogne v digitalno obliko. Seveda pa se *digitalizacija* s tem ne konča. V sklopu *računalniške organizacijske opreme* načrtujemo računalniško strojno, komunikacijsko in programsko opremo, ki naši organizaciji omogoča implementirati ustrezne funkcije računalniško podprtih LIS za zajemanje, hrambo, obdelavo in posredovanje teh podatkov.

Povedali smo že, da so vsi podatki v računalniku predstavljeni kot binarna števila, saj jih računalnik tako najlažje obdeluje/hrani. Obseg števil, ki jih lahko hkrati obdelamo je odvisen od dolžine računalniške ”besede”. Daljša, kot je, večji je obseg oz. natančnost podatka, ki ga predstavlja. Količina podatkov, ki jih obdelujemo, vpliva na prej omenjeno prostorsko zahtevnost obdelave in s tem velikost računalniškega spomina, ki ga za hrambo/obdelavo podatkov potrebujemo. Fizično te podatke običajno hranimo v že omenjenih podatkovnih shrambah podatkov na nivoju organizacije, logično pa jih združujemo v:

- podatkovnih zbirkah (datotekah), ki hranijo surove podatke ali elektronske dokumente,
- podatkovnih bazah (povezanih datotek), ki omogočajo preiskovanje povezanih podatkov v več datotekah in
- podatkovnih skladiščih, ki hranijo časovno urejene zbirke podatkovnih baz.

Medtem ko so podatkovne zbirke osnovni elementi datotečnega sistema vsakega operacijskega sistema in smo jih že obravnavali, bomo podatkovne baze in skladišča, ki organizirano hranijo večje količine medsebojno povezanih po-

datkov in so osnova podatkovne shrambe LIS, obravnavali v višjih letnikih, kjer se bomo podrobneje posvetili njihovi zgradbi, funkcionalnosti in povezavam podatkov, ki jih hranijo.

## 6.1 Digitalizacija besedil in številskih kod

ASCII (angl. "ask-key") kodiranje, je bila prva računalniška abeceda, ki je omogočala kodiranje vnesenih simbolov iz računalniške tipkovnice. Zaradi velikega števila svetovnih jezikov so ta standard razširili z nacionalnimi kodirnimi tabelami. Več informacij lahko najdete na spletu (npr. ASCII). Različici ASCII kodiranj ISO-8859-2 oz. Windows 1250, ki smo ju zlasti uporabniki MS Windows operacijskega sistema srečevali kot izbrani kodirni tabeli za centralno-evropske jezike, vse bolj nadomešča UTF-8 kodiranje, ki zaradi univerzalne uporabnosti in kompatibilnosti z ASCII kodiranjem postaja dominantno kodiranje dokumentov za izmenjavo podatkov. Več informacij o tem kodiranju prav tako lahko najdete na spletu (npr. UTF-8).

V logistiki obdelujemo še razne druge podatke (npr. črtne, RFID kode), ki jih prav tako moramo ustrezno zakodirati, da bi jih lahko hranili v računalniškem spominu, obdelovali in izmenjevali. Gre za govoreče (številske) kode, ki imajo natančno določeno strukturo in omogočajo enolično identifikacijo njihovih nosilcev. Slednja lastnost je za logistiko ključnega pomena in je vplivala na razvoj ustreznih kod za širšo (globalno) rabo. Te kode se hranijo, obdelujejo in prenašajo skupaj z nosilci, ki jih označujejo (npr. paketi, zabojniki, izdelki, njihove komponente, itd.). Zraven identifikacije teh nosilcev, pa omogočajo tudi preiskovanje podatkov, ki so z njimi povezani (npr. navodila za uporabo, tehnična dokumentacija, ipd.).

## 6.2 Digitalizacija zvoka in slike

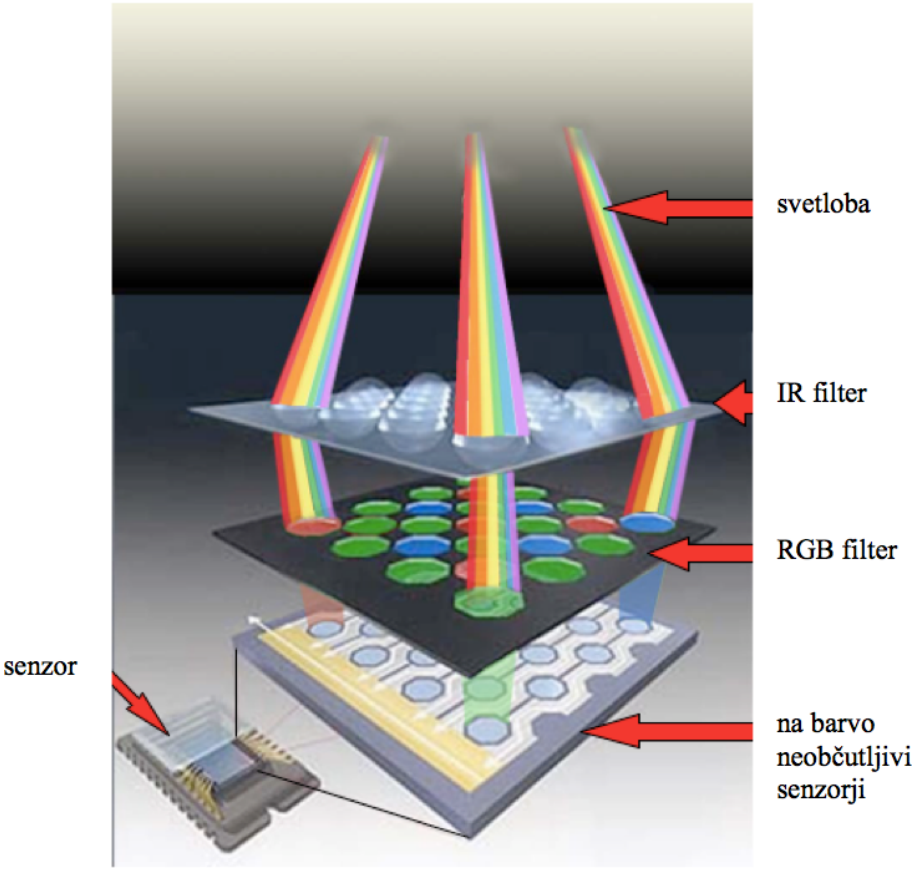
Zvok digitaliziramo s pomočjo posebne – *zvočne kartice* (angl. soundcard). Pri tem kot senzor lahko uporabimo mikrofon ali pa na njen vhod priključimo kakšno drugo analogno napravo, ki oddaja električno valovanje. Mikrofon na primer pretvarja nihanje zračnega tlaka v električno valovanje. Oddajati mora električno karakteristiko, ki je čim bolj verodostojna predstavitev izvirnega zračnega valovanja. Zato mora biti enako občutljiv za vse frekvence, sicer prihaja do linearnega popačenja (dodajanja harmoničnih frekvenc določenim zvokom). Kakovost digitaliziranega zvoka je odvisna od dolžine besede za zapis višine tonov in frekvence vzorčenja (npr. stereo CD audio predvideva 16-bitno ločljivost vzorčenja zvokov po kanalu in frekvenco vzorčenja 44,1 kHz). Po pretvorbi lahko digitalni zvočni zapis obdelujemo – filtriramo, urejamo in dodajamo digitalne učinke. Digitalne zvočne podatke lahko shranimo na različne pomnilne medije (npr. CD, spominske kartice, disk, ipd.). Ob predvajanju zvoka poteka obratna pretvorba zvočnega zapisa – digitalno-analogna pretvorba omogoča tvorbo električnih signalov, ki zanihajo opno v priključenih zvočnikih ali slušalkah. To povzroči nihanje zračnega tlaka ter s tem tvori zvok, ki ga lahko ljudje zaznamo, če se nahaja v naši slišni frekvenci (16 Hz – 16 kHz).

Sliko ali dokument lahko digitaliziramo s pomočjo digitalnega fotoaparata oz. kamere ali optičnega čitalnika. Digitalni fotoaparat ima namesto filma vgrajen poseben senzor tipa CCD (Charged Coupled Device) ali CMOS (Complementary Metal Oxide Semiconductor). Kakovost slike je odvisna od ločljivosti senzorja (števila točk na palec; angl. dots per inch) in števila predstavljivih barv (velikosti besede, ki pomeni odtenek barve oz. barvno globino, podobno kot pri frekvenčnem razponu zvoka), pomembne pa so tudi druge komponente: optika oz. lečje, svetlobna občutljivost senzorja, algoritmi za izračun barvnih nians, ipd.). Ko so svetlobni žarki skoncentrirani na senzorju, ta pretvori svetlobo v električno napetost, algoritem za pretvorbo le-te v slikovne točke pa ustvari digitalno sliko (Slika 49). Princip je enostaven – močnejša kot je svetloba, ki zadene foto-diodo, večja je napetost. Barvo določimo tako, da sve-

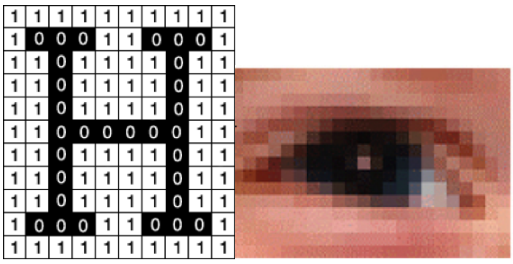
tlobo, ki vdre skozi objektiv v kamero in je običajna bela svetloba, ki vsebuje vse valovne dolžine, filtriramo. Ob filtriranju skozi RGB (Red-Green-Blue) filter, dobimo informacijo o intenzivnosti posamezne komponente filtriranega žarka svetlobe na točki senzorja in s tem podatek o barvi.

Pri CCD senzorjih slikovno informacijo preberemo po vrsticah in po najmanjših celicah senzorja – *slikovnih točkah* oz. *pikslih* (angl. pixel = picture element), zato je čas zajemanja slike nekoliko daljši, vendar je postopek zelo natančen. CMOS senzorji namesto konvertiranja svetlobnih valov na ločenem čipu, fotone prevedejo v elektrone s takojšnjim procesiranjem. Z uporabo sistema ojačevalcev in pretvornikov so ti senzorji hitrejši kot CCDji, vendar spremenljiva učinkovitost pretvornikov in ojačevalcev vpliva na dodajanje šuma sliki. Čeprav tudi večina CMOS senzorjev uporablja RGB filtre za določanje barv, obstaja izboljšava *Foveon* (Sigma), ki izkorišča lastnosti samega silikona za filtriranje barv. Predvsem zaradi hitrosti zajemanja slike, so CCD senzorje v osnovi tipično uporabljali v digitalnih fotoaparatih, CMOS pa v digitalnih video-kamerah. Zaradi številnih izboljšav enih in drugih trenutno stroge ločnice v uporabi ni, so pa kakovostni CCD senzorji tipično nekoliko dražji od CMOS senzorjev. CCD senzorji so našli ciljno področje uporabe v visokoperformančnih aplikacijah kot so npr. elektronski mikroskopi, teleskopi, ipd., medtem ko večina komercialnih digitalnih fotoaparatov in kamer uporablja CMOS senzorje.

*Ločljivost* (angl. resolution) slike je izražena s številom slikovnih pik na palec (dpi – dots per inch) ali številom vrstic na palec (lpi – lines per inch; mera, ki se uporablja npr. pri optičnih čitalnikih). Izbira ločljivosti je povezana z velikostjo datoteke v kateri bo slika shranjena. Z velikostjo slike in gostoto rastra se povečuje poraba pomnilnika, ki je potreben, da sliko shranimo v digitalni obliki. Izbira ločljivosti je odvisna od želenega rezultata. Za objavo na internetu ima prevelika ločljivost slike običajno za posledico počasno nalaganje spletne strani s sliko preko internetnega brskalnika. Večja ločljivost zahteva tudi bolj dolgotrajno osvetljevanje slike pri izdelavi reprodukcije. Na drugi




Slika 49: *Digitalna zaznava slike*
Vir: lasten




Slika 50: *Črno-bela in barvna slika*
Vir: lasten

strani premajhna ločljivost poslabša kakovost slike ob reprodukciji. Skozi prakso so se oblikovale tipske vrednosti za priporočeno ločljivost, glede na namen uporabe – za tisk 300 dpi, za splet 72 dpi, itd.

Podatki o barvi svetlobnih pik se nahajajo v nizu bitov fiksne dolžine. Število bitov, uporabljenih za eno piko, predstavlja barvno globino slike. Večja kot je barvna globina, več različnih barv lahko prikažemo na sliki – bolj je slika kontrastna, večja pa je tudi potrebna količina pomnilnika. Podobno kot pri ločljivosti so se tudi tukaj oblikovale tipske vrednosti glede na namen (Slika 50): črno/bela – 1 bit, sivinska lestvica (angl. grayscale) – 8 bitov, osnovna barvna (angl. color) – 8 bitov, hi-color – 16 bitov, true-color – 24 bitov, itd.

## 6.3 Črtne kode in RFID

Glede na zgoraj obravnavano tematiko je v okviru digitalizacije primerno omeniti obe najbolj pogosti tehnologiji označevanja, ki se v logistiki uporabljata za avtomatizirano identifikacijo artiklov, pa tudi za sledenje artiklov/pošiljk. Prva predvideva optično odčitavanje dvobarvnih (običajno črno/belih) etiket, druga pa radijsko identifikacijo. Obe zahtevata uporabo ustreznih perifernih naprav s senzorji za optično branje črtnih ter elektromagnetno odčitavanje RFID oznak.

Trenutno stanje pri označevanju proizvodov v industriji kot tudi pri označevanju pošiljk izkazuje prednostno uporabo dobro uveljavljenih *EAN/UPC črtnih kod* iz več razlogov (Tabela 4). Glavni je ekonomski – te oznake je poceni vzdrževati in odčitavati ter zadoščajo za enolično identifikacijo proizvodov. Tehnologija tiskanja in odčitavanja teh oznak je dozorela in se pocenila do te mere, da lahko vse potrebno najdemo že v lokalni diskontni trgovini (Slika 51). Tiskanje črtnih kod na embalažo praktično izniči njihovo ceno. Slabost črtnih kod je njihova obraba, ki povzroči slabšo berljivost. Prav tako informacije, ki jo hranijo, ne moremo preprosto posodabljati. Če to želimo, moramo na podlagi obstoječe praviloma natisniti novo kodo, ta pa mora nadomestiti (prekriti) staro. Za razliko od ročnega čitalca na Sliki 51 fiksni čitalci z zrcali omogočajo, da usmerjanje čitalca na črtno kodo ni več potrebno, in jih lahko v vidnem polju čitalca hitro odčitamo ne glede na orientacijo. Sledljivost artiklov je mogoče relativno preprosto doseči preko elektronske izmenjave podatkov (EDI) med v splet povezanimi informacijskimi sistemi akterjev v oskrbovalni verigi – s sistematičnim spremljanjem in združevanjem kod od dobaviteljev komponent do prodajalcev na drobno.

Regulativa *GS1* določa naslednjih 10 korakov za oblikovanje standardne črtne kode (vir: GS1):

1. Določitev identifikacijske številke (predpone) podjetja
2. Določanje identifikacijskih številk izdelkov (v skladu s proizvodnim programom)
3. Izbor načina tiskanja (glede na vsebino oznake – statična/dinamična)
4. Izbor primarnega okolja odčitavanja (POS, v distribuciji, posebna okolja)
5. Izbor simbologije (format črtne kode)
   (a) Prodajna mesta EAN (GS1 – 8 ali 13, DataBar)


Slika 51: *POS terminal in črtna koda EAN-13*
Vir: lasten

(b) Spremenljivi podatki (GS1 – 128, DataBar, DataMatrix)

(c) Na Embalaži (ITF-14)

6. Izbor velikosti črtne kode
7. Oblikovanje besedila črtne kode (“človeško berljivi” podatki na oznaki)
8. Izbira barve črtne kode
9. Določanje pozicije črtne kode na artiklu
10. Načrt kakovosti črtne kode

RFID oznake (Tabela 5) podobno kot črtne kode enotno označujejo artikle. Tipična RFID oznaka vsebuje črtno identifikacijsko kodo in dodatne informacije na vgrajenem čipu velikosti bucikine glave. Omogoča tudi zapisovanje sledilnih (angl. tracking) informacij. Za razliko od črtne kode optično odčitavanje tu ni potrebno, omogoča pa tudi branje več oznak naenkrat. Poznamo pasivne

Tabela 4: Primerjava črtnih kod

| **Lastnosti** | **Linearne (1D) črtne kode** | **Matrične (2D) črtne kode** |
|---|---|---|
| **Kapaciteta** | Do 128 znakov | Do 4296 znakov |
| **Odčitavanje** | Optično | Optično |
| **Trajnost** | Papirnate ali plastificirane etikete | Papirnate ali plastificirane etikete |
| **Učinkovitost odčitavanja** | 1 koda naenkrat | 1 koda naenkrat |
| **Branje/Zapisovanje** | DA/NE | DA/NE |
| **Varnost podatkov** | Ni podprta | Omejena z aplikacijo |
| **Cena** | 1-10 centov | 1-10 centov |

in aktivne RFID oznake, ki se ločijo po tem, da imajo slednje vir napajanja, kar poveča njihovo območje zaznavanja. T.i. pol-aktivne oznake se aktivirajo le po potrebi (npr. ob pritisku na gumb daljinca za odpiranje garažnih vrat). Aktivne oznake neprestano javljajo svojo prisotnost sprejemnikom, kar omogoča proaktivno ukrepanje (npr. odklepanje vrat avtomobila, priprava stroja za naslednjo proizvodno fazo, obvestilo o prispelem transportu,...). Čeprav RFID prav tako kot črtna koda v osnovi omogoča identifikacijo, ima zaradi svoje tehnične izvedbe številne prednosti, ki povečujejo njeno uporabnost. Omogoča popolnoma avtomatizirano odčitavanje oznak in s tem zmanjšuje stroške dela. Omogoča odčitavanje oznak, ki se gibljejo, in s tem omogoča njihovo sledenje. Zmožnost osveževanja shranjenih podatkov omogoča sledljivost skozi oskrbovalno verigo, pa tudi avtomatsko zgodovino manipulacij (npr. prekladanja, vzdrževanja, servisiranja, ipd.). Služi lahko tudi kot potujoča podatkovna shramba, kar zmanjšuje potrebo po hranjenju in prenašanju velikih količin podatkov pri in med akterji v oskrbovalni verigi. Ti podatki so neodvisni od njihovih informacijskih sistemov in se torej nenamerno ne morejo izgubiti. RFID omogoča tudi trajno označevanje virov – za čas življenjske dobe nosilca podatkov. Hitrost odčitavanja RFID oznak je bistveno večja kot pri črtnih kodah, zlasti ob hkratnem odčitavanju več oznak. Seveda pa tudi tehnologija RFID ni brez težav, saj je podvržena elektromagnetnim motnjam in napakam

Tabela 5: Primerjava RFID kod

| **Lastnosti** | **RFID** |
|---|---|
| **Kapaciteta** | Več tisoč znakov |
| **Odčitavanje** | Radijsko (EM) |
| **Trajnost** | Plastificirane etikete s čipom |
| **Učinkovitost odčitavanja** | Več kod hkrati |
| **Branje/Zapisovanje** | DA/DA |
| **Varnost podatkov** | Omejena s količino podatkov |
| **Cena** | 10-20 centov (pasivne oznake), do 100 EUR (aktivne oznake) |

pri odčitavanju, ki lahko nastanejo zaradi odbojnih materialov. RFID sistemi delujejo v ultra kratkovalovnem (860-900 MHz) in mikrovalovnem (2,4 GHz) frekvenčnem območju, kjer je motenj še najmanj.

Z globalnim standardom GS1 EPC Gen2 (ISO/IEC 18000-6:2013) je bil uveden tehnološki standard, ki določa komunikacije med RFID oznakami in sprejemniki. Podobno kot pri črtnih kodah GS1, EPCglobal standardi povezujejo tehnologijo RFID in sistem označevanja EPC posameznega artikla (izdelek, logistična enota, lokacija, osnovno sredstvo, vračljivo sredstvo, dokument, ...) z namenom neposredne, avtomatske identifikacije ter spremljanja artiklov in logističnih enot po oskrbovalnih verigah.

Z *EPCglobal* standardi je opredeljeno (vir: GS1):

- Katere podatke zapisujemo v RFID oznako?
- Kako zapisujemo podatke v RFID oznako?
- Kako RFID oznake komunicirajo z RFID bralno/pisalno napravo?
- Kakšni so programski vmesniki med RFID bralno/pisalno napravo in informacijskimi sistemi?
- Kako se izvaja izmenjava podatkov med poslovnimi partnerji v oskrbovalni verigi?

Tabela 6: Uporaba tehnologij označevanja

| **Tehnologija označevanja** | **Aplikacija** |
|---|---|
| Črtna 1D (R-oznake) | Enostavni artikli, namenjeni maloprodaji ali komponente večjih izdelkov |
| Črtna 2D (R-oznake) | Storitve (npr. UPS, letalski promet) in večji proizvodi, namenjeni maloprodaji, ki zahtevajo sledenje |
| RFID razreda 1 (pasivne, R-oznake) | Artikli, ki zahtevajo masovno identifikacijo, kontrola pristopa |
| RFID razreda 2 (pasivne, RW-oznake) | Artikli, ki zahtevajo vodenje pedigreja |
| RFID razreda 3 (pol-aktivne, RW-oznake) | Kontrola pristopa z dodano sledilno informacijo |
| RFID razreda 4 (aktivne, RW-oznake) | Sledenje sredstev in oseb v zaprtih prostorih – lokacijske aplikacije |
| RFID razreda 5 (avtonomne, RW-oznake) | Sledenje sredstev in oseb v zaprtih prostorih in na prostem – lokacijske aplikacije in interoperabilnost |

EPCglobal standardi so tudi osnova za GDSN (Global Data Synchronization Network). Le-ta omogoča avtomatiziran zajem in izmenjavanje matičnih podatkov o izdelkih in pakiranjih ter podjetjem omogoča centralno in globalno upravljanje teh podatkov ter tako tvori skupno podatkovno skladišče za vse partnerje v oskrbovalni verigi.

Odgovor na vprašanje, kakšno bo označevanje in sledenje izdelkov v prihodnosti, ni enoličen. Kako torej v konkretnem primeru izbrati najprimernejšo obliko označevanja? V pomoč je lahko Tabela 4, ki navaja tehnologije označevanja s tipičnimi področji uporabe. Medtem, ko se 1D črtna koda razvija na primer za označevanje izdelkov, ki so tako majhni, da jih v preteklosti sploh ni bilo mogoče označiti (npr. Databar) in se razvijajo nove 2D kode, ki bodo omogočale ne le korekcijo napak ob odčitavanju, temveč tudi kriptiranje podatkov (npr. Datamatrix), zori tudi tehnologija RFID. Med 2D črtnimi kodami se je najbolj uveljavila QR koda, ki jo lahko preberemo kar s pametnim mobilnim telefonom. Še več, čitalec jo je sposoben prečitati z ekrana mobilnega

telefona, kar s pridom izkorišča npr. letalska industrija pri registraciji potnikov s t.i. e-ticketi. Na drugi strani imamo cel spekter RFID in sorodnih (npr. Near Field Communication, NFC) oznak.

Standardizacija pri RFID še ni tako dorečena kot pri črtni kodi. Prav tako bo treba premostiti tehnološke omejitve RFID, jih prilagoditi področjem uporabe in na ta način povečati zanesljivost identifikacije. Varnost in zaupnost bosta pri RFID našli svoje mesto, ko bo tehnologija v masovni uporabi in bodo aplikacije narekovale različne stopnje varnosti, se pa že razvija. Vsekakor je RFID na pohodu in mnogi med nami razne oblike RFID že bolj ali manj zavedno uporabljamo, na primer, s tem ko z abonentsko kartico Slovenskih železnic izkažemo, da smo plačali prevoz na določeni relaciji, registriramo svoj prihod na delo, si privoščimo jutranjo kavico na delikomatu ali pa daljinsko odklenemo svoj avtomobil.

# 7 Računalniška organizacijska oprema (OW)

*Računalniška organizacijska oprema (angl. Orgware)* je povezovalni element LIS. Pomeni predvsem krepitev zmogljivosti različnih institucionalnih akterjev, ki so vključeni v proces uvajanja novih tehnologij. Združuje vse metode, ukrepe, predpise in specifikacije s katerimi časovno in funkcijsko usklajujemo delovanje ostalih komponent LIS: strojne, programske, komunikacijske opreme in osebja. V LIS uvaja računalniško podprto logistično inženirstvo – planiranje virov, aktivnosti/operacij – in ustrezne organizacijske spremembe, ki vplivajo na poslovno logiko organizacije. *Poslovna logika (angl. business intelligence)* je bistvenega pomena za vsak informacijski sistem. Ta koordinira delovanje *osebja (angl. liveware, LW)*, ki navsezadnje, a ne nazadnje, kot glavni akterji predstavljajo tudi najpomembnejšo komponento vsakega informacijskega sistema.

V tem poglavju se bomo najprej posvetili planiranju operacij, ki nam pomaga specificirati vire in funkcije informacijskega sistema ter njihove medsebojne povezave s ciljem definicije procesov v organizaciji. V nadaljevanju se bomo pomudili še pri planiranju aktivnosti, ki koordinira delovanje osebja. Seveda tu ne moremo predstaviti celotnega spektra različnih metod in specifikacij, ki se v konkretnih okoljih uporabljajo kot organizacijska oprema, bomo pa celovito predstavili delokrog in vse relevantne vidike te komponente LIS, da bodo bralci pridobljeno znanje lahko uporabili, nadgradili in povezali z vsebinami sorodnih in povezanih predmetov. Nazadnje se bomo v naslednjem poglavju (Poglavje 8) dotaknili še tistega dela organizacijske opreme, ki predstavlja ukrepe za zagotavljanje varnosti informacijskih sistemov in zaupnosti podatkov, ki jih hranijo. Ker ti ukrepi naslavljajo vse dele LIS in niso samo organizacijske narave, so predstavljeni v ločenem poglavju.

## 7.1 Planiranje operacij

Omenili smo že metode programskega inženirstva, ki se nanašajo na razvoj programske kode. Gre za tehnike, ki se uporabljajo predvsem pri specifikaciji posameznih aplikacijskih programov oz. programskih komponent informacijskih sistemov, ki jim pripadajo. Zajemajo algoritme delovanja ter informacijske vire, ki jih potrebujejo. V ta namen se bomo tukaj posvetili metodam logističnega inženirstva, ki so najbolj pogoste pri specifikaciji funkcionalnosti LIS. Prednost teh metod je grafična predstavitev, ki je po eni strani dovolj nazorna, po drugi pa dovolj rigorozna, da omogoča razvijalcem programske opreme na osnovi tako podanih načrtov zasnovati algoritme in na njihovi osnovi programe, ki bodo implementirali planirane funkcije oz. komponente (LIS) v skladu s podanimi specifikacijami.

### 7.1.1 Miselni diagrami

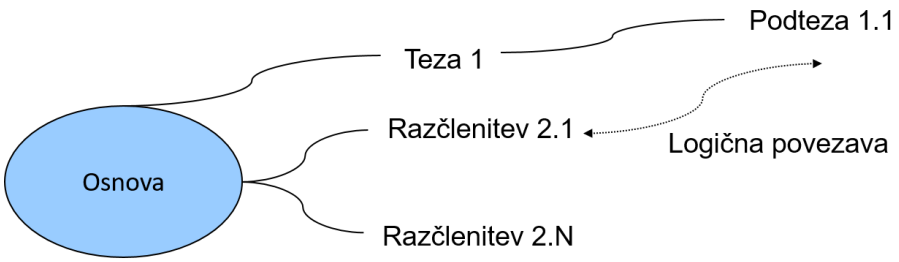
Ob snovanju se običajno najprej skoncentriramo na novo (informacijsko) rešitev in jo obravnavamo iz vseh zornih kotov. Pri tem skušamo čim bolj celovito razgraditi idejno rešitev in jo umestiti v organizacijo in njeno okolje, oceniti medsebojne vplive in posledice ter oblikovati ”kaj-če” scenarije, povezane z njeno implementacijo. V primeru, da se odločimo novo rešitev sprovesti, konceptualnemu sledi logično planiranje. Izvedbeni načrt našo idejno rešitev postavi na časovno os in v realne okvire, ki izhajajo iz razpoložljivih virov (tako materialnih kot nematerialnih). Izvedba plana ima za rezultat fizični sistem, ki predstavlja opredmetenje naše idejne rešitve v skladu z njeno zasnovo in izvedbenim načrtom. V našem primeru se bomo skoncentrirali na implementacijo informacijskega sistema. Dobra praksa narekuje, da hkrati z novo rešitvijo vzpostavimo tudi strategijo trajnostnega delovanja in mehanizme upravljanja kakovosti.

Idejno rešitev lahko učinkovito vizualiziramo s pomočjo *miselnih diagramov (angl. mind chart)*. Le-ti predstavljajo mentalni postopek upodabljanja misli in imajo širši spekter uporabe. V poslovnem svetu so v uporabi predvsem pri

strateškem planiranju za predstavitev in snovanje poslovnih situacij ter ustreznih reakcij nanje. Začnemo s tem, da v oval sredi delovnega lista zapišemo našo idejo oz. koncept, ki bi ga želeli obravnavati. Če ugotovimo, da gre za sestavljen koncept, ga lahko razbijemo na več pod-konceptov – narišemo dodatne ovale, ki jih z neusmerjenimi linijami povežemo z osnovnim. Le-te lahko obravnavamo tudi ločeno, če je to mogoče in bi nam njihova podrobna obravnava preveč zameglila osnovno sliko. Nasploh je boljša strategija narisati en sam diagram (angl. "big picture") na katerem je vse, kar je relevantno za naš koncept. Nato se skoncentriramo na vse, kar je z njim povezano. Misli, ki se nam porajajo okoli koncepta, nizamo na neusmerjene linije, ki jih rišemo kot veje iz ovalov. Le-te so lahko poljubno goste in v primeru izpeljanih ali razčlenjenih misli porajajo nove veje vzporedno ali kot nadaljevanje obstoječih.

Pomembno pri miselnih diagramih je, da gre za (skupinski) postopek, ki je dokaj hiter in neselektiven pri izbiri predstavitve miselnih vzorcev – lahko jih predstavimo besedilno ali s simboli. Zaradi hitrosti postopka pogosto govorimo o *viharjenju možganov (angl. brain-storming)*, ki je ključna sestavina metode. Večkrat vrišemo koncepte tudi brez povezav ali pa predvidimo prazne povezave, ki jim šele naknadno dodelimo koncepte. Logične povezave med koncepti ponazorimo s poimenovanimi črtkanimi linijami z obojestranskimi puščicami. *Urejanje misli* (čiščenje diagrama) običajno sledi, ko je postopek zaključen oz. je miselni diagram v glavnem "nared". Glavne gradnike miselnih diagramov ponazarja Slika 52.

**Primer:** Naše podjetje želimo opremiti s sodobnim upravljavskim informacijskim sistemom. Odpiramo prodajalno ProInstal z ogrevalno tehniko (radiatorji, ventili, peči, ...) in že poznamo nekaj svojih najpomembnejših dobaviteljev (npr. THS, Mavi, Merkur,...). Naš prodajni program sestavljajo peči za centralno kurjavo in aluminijasti radiatorji znamk Aklimat, Radson in Vogel-Noot. Seveda se želimo potruditi že kar v začetku in organizirati svoj informacijski sistem tako, da bo naše poslovanje čimbolj tekoče in transparentno.

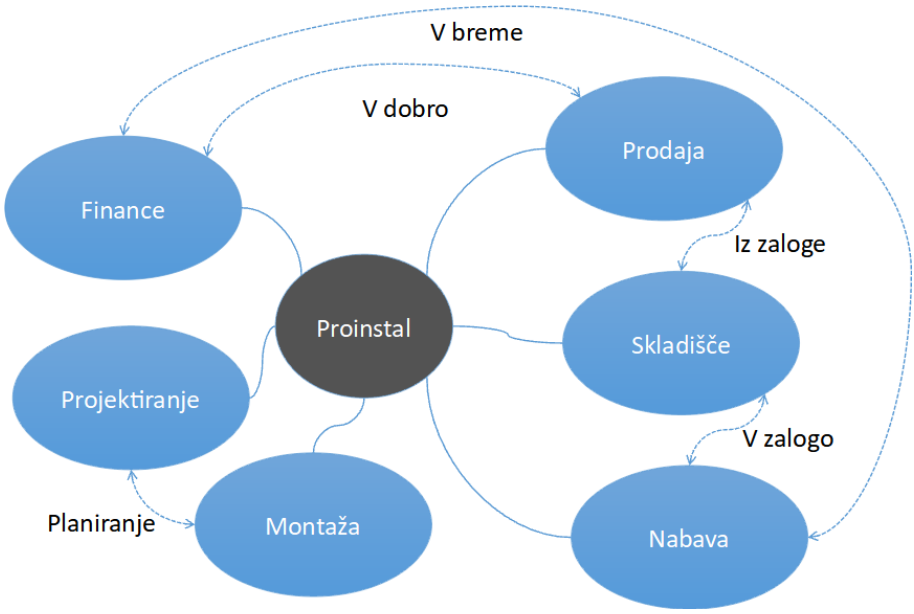



Slika 52: Zgradba miselnega diagrama
Vir: lasten

Ker nimamo veliko začetnega kapitala, smo se odločili delati z odprto-kodnim ERP IS Odoo. Na prvem sestanku se srečamo kar vsi in po oddelkih naredimo načrte, kako bomo pristopili k delu:

1. nabava naredi seznam dobaviteljev in zbere cenike
2. prodaja analizira trg in ugotavlja cenovni razpon
3. skladišče organizira skladiščne kapacitete, tako da bomo izdelke enostavno shranjevali in našli – zaenkrat planiramo samo eno skladišče na lokaciji prodajalne z možnostjo kasnejšega odpiranja novih lokacij po istem zgledu
4. finance vzpostavijo osnovno evidenco – blagajno, transakcijski račun ter računa preko katerih potekata nakup in prodaja izdelkov
5. projektiva načrtuje zadolžitve zaposlenih in oblikuje načrt dela v obliki montažnih projektov

V skupinah se vživimo v posamezne vloge, debatiramo vsebine in oblikujmo miselne diagrame za skupno obravnavo. Diagrame na koncu združimo v en sam velik načrt, ki bo vodilo našega obnašanja pri poslovanju. Miselni diagram, s katerim začnemo sestanek z vodstvenimi delavci posameznih oddelkov, sestoji iz glavne teme in več pod-tem, ki sovpadajo z oddelki (npr.

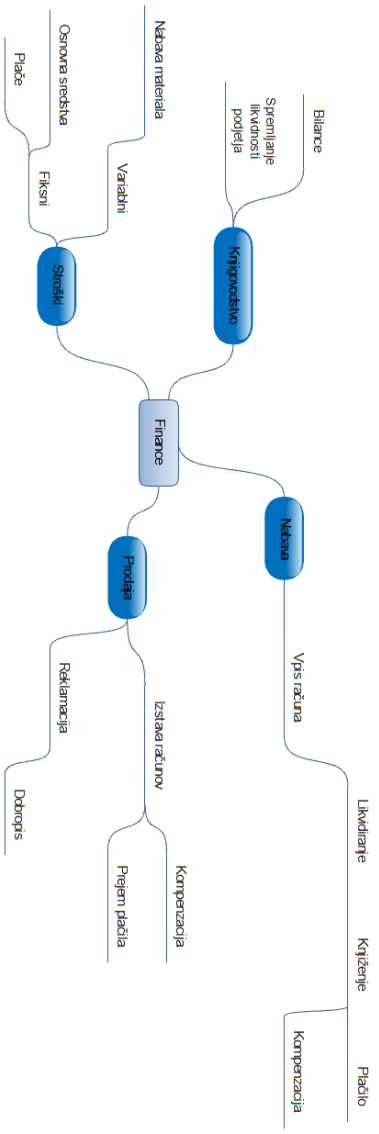



Slika 53: Miselni diagram ProInstal
Vir: lasten

Slika 53). Vodje oddelkov na podlagi poznavanja strategije podjetja zasnujejo lastne diagrame okoli teh pod-tem. Kot primer na Sliki 54 navajamo miselni diagram vodje finančnega oddelka s koncepti, o katerih je on razmišljal.

### 7.1.2 Odločitvene tabele

Kot vidimo, si z miselnimi diagrami lahko ustvarimo dober pregled informacijskega sistema podjetja z vsemi relevantnimi entitetami in medsebojnimi razmerji, ne moremo pa z njimi modelirati aktivnosti. Večkrat smo soočeni s situacijo, ko moramo za posamezne situacije oblikovati pravila, t.i. *poslovno logiko (angl. business rules management systems, BRMS)*, ki nam pomagajo pri ukrepanju ob določeni kombinaciji predpogojev.




Slika 54: Miselni pod-diagram ProInstal
Vir: lasten

Za opis ”kaj-če” pogojev so primerne *odločitvene tabele (angl. decision tables)*. S pomočjo odločitvenih tabel modeliramo kombinacije pogojev in ukrepov, ki nam olajšajo odločanje v konkretnih situacijah ter realizacijo ustreznih programskih orodij, saj predstavljajo organizacijsko logiko ključnih poslovnih odločitev. Imajo preprosto strukturo, ki jo v splošnem sestavljajo štirje kvadranti (Tabela 7).

Tabela 7: Zgradba odločitvene tabele

| Seznam pogojev | Vrednosti pogojev |
|---|---|
| Seznam ukrepov | Ukrepi glede na vrednosti pogojev |

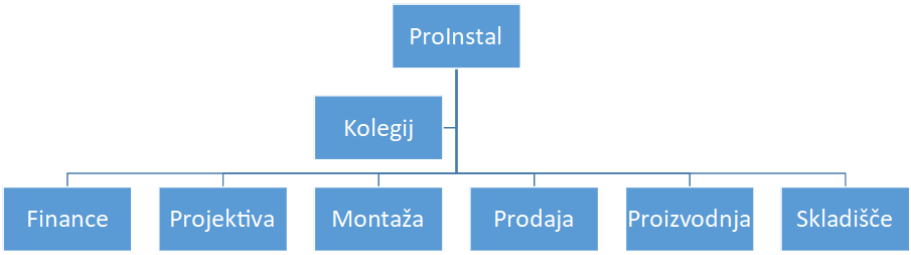
**Primer:** Kot primer poslovne logike navajamo odločitveno tabelo za cikel obdelave naročila podjetja ProInstal (Tabela 8).

Tabela 8: Poslovno pravilo poteka naročila

| Pravila / Pogoji | | Naročilo | Načrt | Material | Montaža | Plačilo |
|---|---|---|---|---|---|---|
| | Naročilo izdano | ne | da | da | da | da |
| | Načrt izdelan | ? | ne | da | da | da |
| | Material naročen | ? | ? | da | da | da |
| | Material na zalogi | ? | ? | ? | da | da |
| | Montaža potrjena | ? | ? | ? | ? | da |
| | Montaža izvršena | ? | ? | ? | ne | da |
| | Naročilo plačano | ? | ? | ? | ? | ne |
| Aktivnosti | | | | | | |
| | Predstavitev | x | | | | |
| | Načrtovanje | | x | | | |
| | Priprava materiala | | | x | | |
| | Nalog za montažo | | | | x | |
| | Montaža | | | | x | |
| | Izvedba plačila | | | | | x |

### 7.1.3 Organizacijski diagrami

*Organizacijski diagram (angl. Organizational chart)* je grafična ponazoritev organizacijske strukture podjetja. Hierarhično prikazuje oddelke, delovne skupine in delovna mesta v organizaciji. Zanj se je udomačilo tudi ime *organigram*.




Slika 55: Organizacijska shema ProInstal
Vir: lasten

V organizacijskem diagramu modeliramo nosilce posameznih aktivnosti v okviru procesov organizacije do nivoja delovnega mesta – posameznega nosilca, ki opravlja svoj delokrog oz. delovni proces (npr. Slika 55).

### 7.1.4 Procesni diagrami

Medtem ko nam odločitvene tabele olajšajo formulacijo pravil, po katerih se lahko ravnamo ob odločitvah v določenih problemskih situacijah (npr. ob okvarah, odločitvah o nadaljnjih korakih glede na stanje procesa, ipd.), na drugi strani za specifikacijo poteka procesov potrebujemo njihovo formalno predstavitev. To nam omogočajo *procesni diagrami (angl. process chart)*, ki so sestavljeni iz procesov, podprocesov in posameznih aktivnosti znotraj leteh. Z njihovo pomočjo vizualiziramo sosledje procesov, identificiramo glavne in podporne procese ipd. (npr. Slika 56).

### 7.1.5 Diagrami poteka

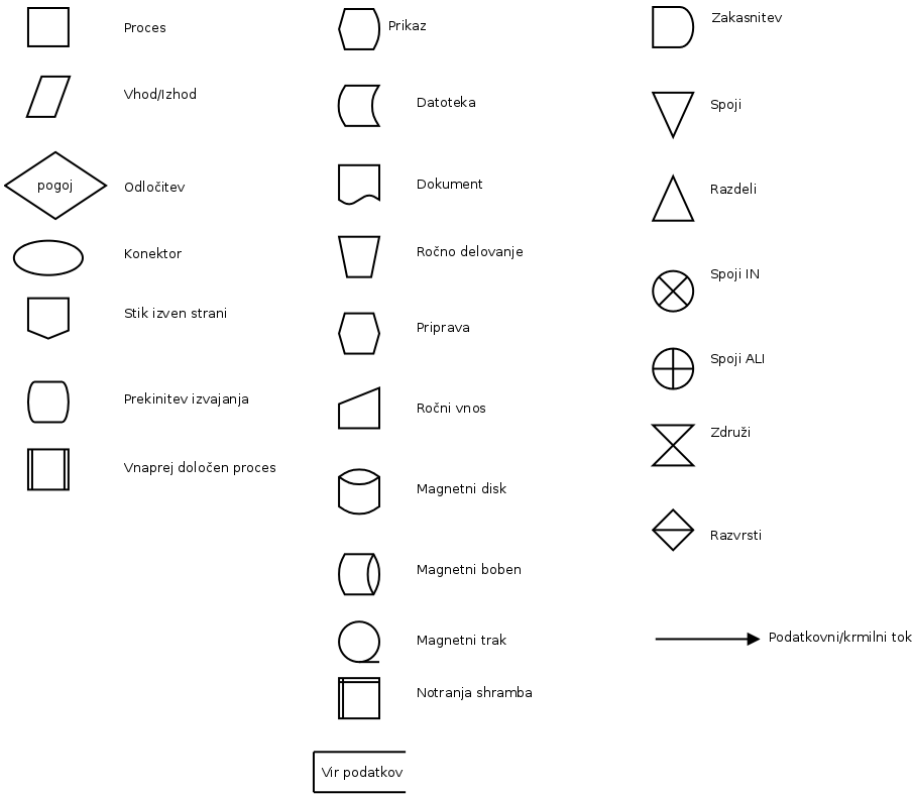
Procesni diagrami sicer omogočajo prikaz sosledja poslovnih procesov, vendar so pri tem vseeno precej omejeni. Procesnim diagramom predvsem manjkajo konstrukti za modeliranje podatkovnih, krmilnih tokov in podatkovnih nosilcev, ki so pri tem udeleženi. Prav tako iz njih ni mogoče razbrati nosilcev


Slika 56: Procesni diagram podjetja ProInstal
Vir: lasten

posameznih aktivnosti in odločitev, v skladu z organizacijsko strukturo podjetja. Specifikacijo poteka posameznih procesov omogočajo *diagrami poteka (angl. flowchart). Diagrami poteka (ISO 9001:2015)* so vizualna predstavitev zaporedja korakov (operacij) in odločitev, ki so potrebni za izvedbo procesa (Slika 57).

Za formalno predstavitev algoritmov, na katerih temeljijo poslovni procesi, so bili razviti *linijski diagrami poteka (angl. Swimlane diagram)* z razširjenim naborom konstruktov za predstavitev procesov, pogojev, podatkovnih in krmilnih tokov ter njihovih nosilcev. Takšno ime so dobili zaradi analogije s "plavalnimi stezami" znotraj katerih nosilci poslovnih aktivnosti izvajajo operacije poslovnega procesa.

Osnova linijskih diagramov poteka je dvodimenzionalni koordinatni sistem. Na eni od osi se nahajajo *steze (angl. lanes)*, ki predstavljajo *oddelke, delovna mesta ali posameznike*, ki sodelujejo v procesu. Na drugi osi modeliramo procesne aktivnosti in operacije s pomočjo konstruktov *diagramov poteka*. Ta




Slika 57: Konstrukti diagramov poteka
Vir: lasten

ponazoritev omogoča hitro identifikacijo nosilcev procesnih korakov in odločitev. Procesni in podatkovni tokovi so ponazorjeni s puščicami, ki ne razločujejo podatkovnih od krmilnih tokov, lahko pa (kjer je to primerno) eksplicitno modeliramo pretok (elektronskih) dokumentov.

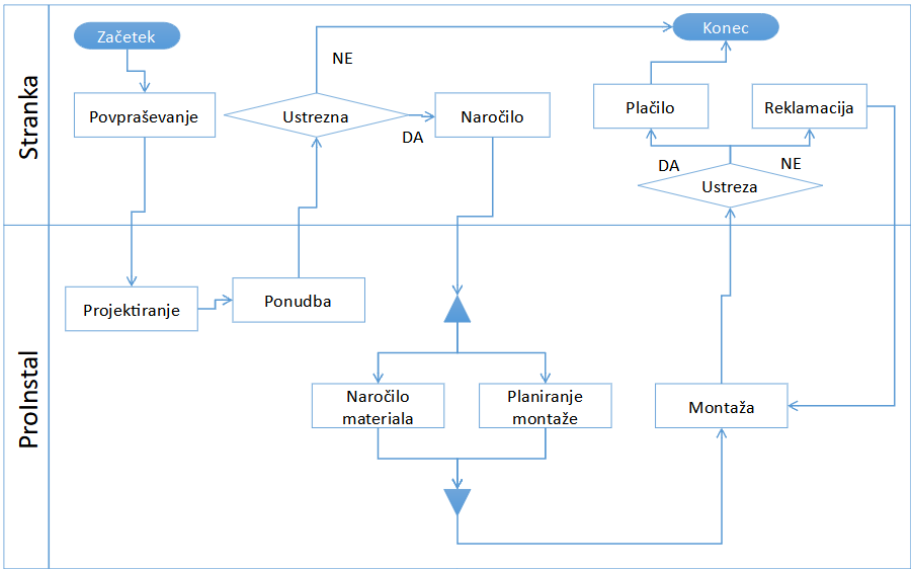
Swimlane diagrami sestojijo iz:

- konstruktov diagramov poteka:
  - aktivnosti,
  - konektorjev in
  - nosilcev (krmilnih) informacij ter
- nosilcev aktivnosti (”steze”).

Kljub pestrosti konstruktov je zlasti pri specifikaciji poslovnih procesov težnja ohranjati preprostost opisov, saj kompleksnost dodajata njihova številčnost in prepletenost. Zato se v omenjenem standardu ISO 9001 za specifikacijo poslovnih procesov uporabljajo zgolj osnovni konstrukti za predstavitev atomarnih operacij, odločitev, tokov in podatkov.

**Primer:** S pomočjo Swimlane diagramov zlahka vizualiziramo in na ta način odkrijemo morebitne pomanjkljivosti (npr. podvajanja, prekrivanja, ipd.) v poslovnem procesu (npr. Slika 58).

Tudi razvoj Swimlane diagramov se je nadaljeval v smislu standardizacije in širitve nabora konstruktov za prikaz poslovnih pravil. Da bi lažje obvladali dinamiko odvijanja procesov skozi čas, so težili tudi k razvoju ustreznih programskih orodij, ki bi omogočala modeliranje in simulacijo poslovnih procesov. Rezultat teh prizadevanj je bil razvoj BPMN diagramov, ki jih predstavljamo kot zadnje.




Slika 58: Prodajni proces podjetja ProInstal
Vir: lasten

### 7.1.6 BPMN diagrami

*Business Process Modeling Notation (BPMN)* je bila razvita leta 2002 na osnovi procesno-orientiranih specifikacijskih jezikov. Sestavljajo jo naslednji gradniki:

- objekti pretoka,
- objekti povezovanja,
- bazeni (angl. pool) in steze (angl. lane) za prikaz povezanih sistemov ali pod-sistemov ter
- simboli za prikaz preostalih pomembnih entitet (dogodki, podatkovni objekti, opombe, ipd.).

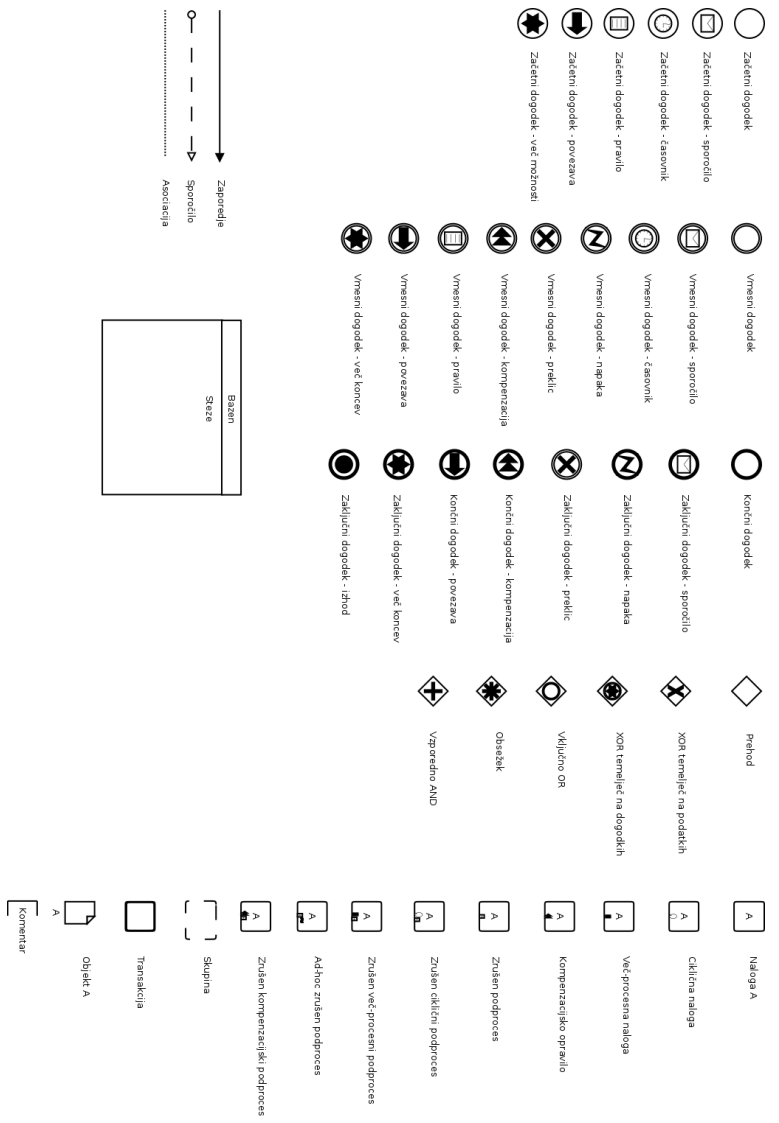
*Business Process Executing Language (BPEL)* omogoča simulacijo BPMN načrtov. S tem bi lahko odkrili tudi morebitne pomanjkljivosti in ozka grla v poslovnih procesih, vendar je zaradi njihove kompleksnosti in/ali pomanjkljivih načrtov to pogosto nemogoče. Konstrukti BPMN diagramov so našteti na Sliki 59. Razen nekoliko razširjenega nabora simbolov za prikaz dogodkov lahko opazimo predvsem, da BPMN uvaja bazene dodatno k stezam za prikaz fizično in/ali logično povezanih poslovnih enot (oddelkov, podružnic, ipd), pri čemer ohranja steze za prikaz delovnih področij posameznih oddelkov ali posameznikov znotraj teh enot.

**Primer:** Za primerjavo s Swimlane diagramom na Sliki 58, Slika 60 prikazuje prodajni proces podjetja ProInstal v BPMN notaciji.

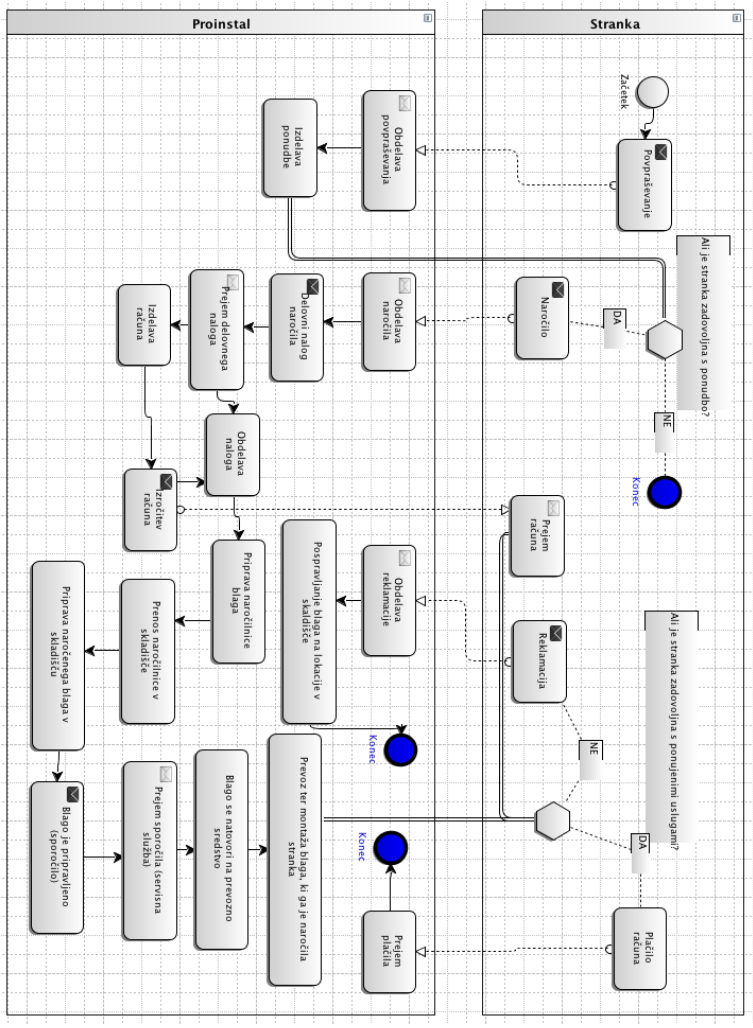
### 7.1.7 Infrastrukturni diagrami

Obravnavani formalizmi za opis poslovnih procesov so metodološka orodja inženiringa logističnih informacijskih sistemov za vizualizacijo in snovanje le-teh na različnih nivojih podrobnosti in iz različnih vidikov. Večinoma so podprti z ustreznimi *načrtovalskimi programskimi orodji (angl. Computer Aided Design, CAD)*, ki omogočajo njihovo konstruiranje, hranjenje, popravljanje in vključevanje v različne dokumente – npr. za predstavitve in projektno dokumentacijo. Nasploh so ali presplošni ali pa prekompleksni za formalno obravnavo in avtomatizirano obdelavo, so pa dobra osnova za zgodnje odkrivanje napak v načrtih in podrobno planiranje poslovnih procesov ter ustrezne informacijske podpore – logističnih informacijskih sistemov (LIS).

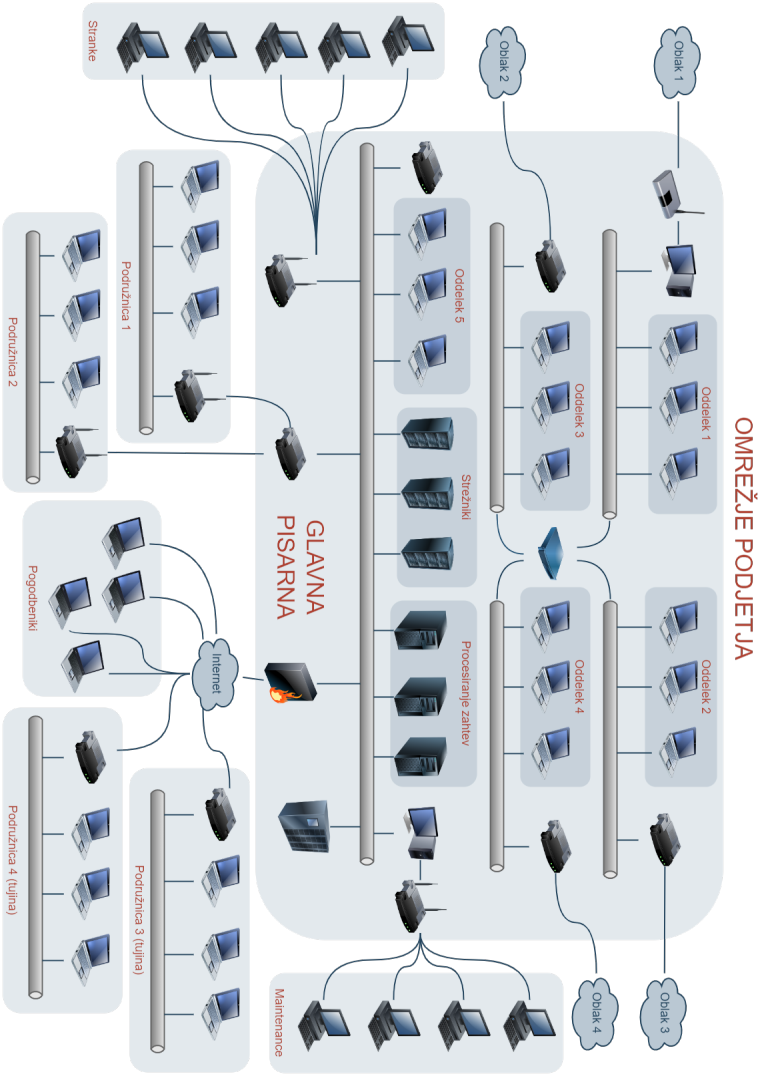
V nadaljevanju navajamo še nekaj tipov diagramov, ki se pogosto pojavljajo pri modeliranju informacijsko-komunikacijske infrastrukture (IKT). Vsebujejo konstrukte računalniške strojne (Slika 61), komunikacijske (Slika 62) in IoT opreme (Slika 63) na funkcionalnem nivoju. Služijo predvsem dokumentiranju konfiguracije oz. umestitve ustreznih infrastrukturnih komponent LIS v fizični prostor organizacije.




Slika 59: Konstrukti BPMN notacije
Vir: lasten




Slika 60: BPMN model prodaje ProInstal
Vir: lasten



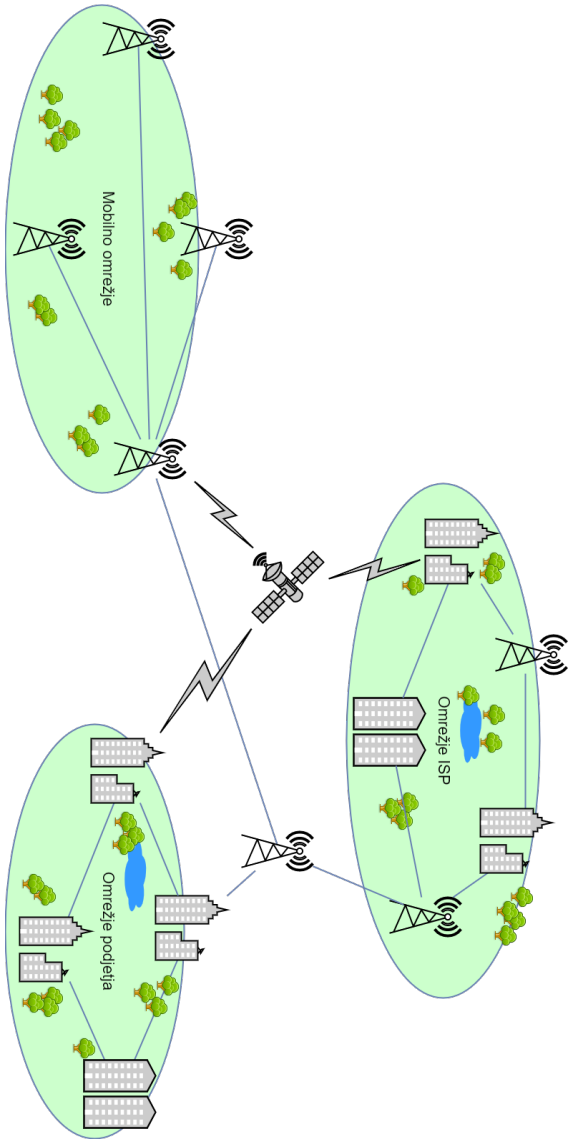

Slika 61: Informacijska infrastruktura organizacije
Vir: lasten, 2023

Večkrat moramo zaradi uvajanja novih infrastrukturnih komponent narediti načrt IKT opreme v podjetju (npr. Slika 61). Pri tem ne modeliramo natančne konfiguracije komponent, ampak se skoncentriramo na posamezne funkcionalne komponente (strežnike, delovne postaje, omrežje, požarne zidove, ipd.) in njihove medsebojne povezave (LAN, WAN, ipd.). Takšni diagrami so koristni, da uvidimo, kako bodo nove komponente vključene v obstoječ sistem, pa tudi zaradi vzdrževanja aktualnega načrta IKT opreme, ki nam lahko služi tudi kot osnova za načrtovanje vzdrževanja, posodobitev ter s tem povezanih stroškov.

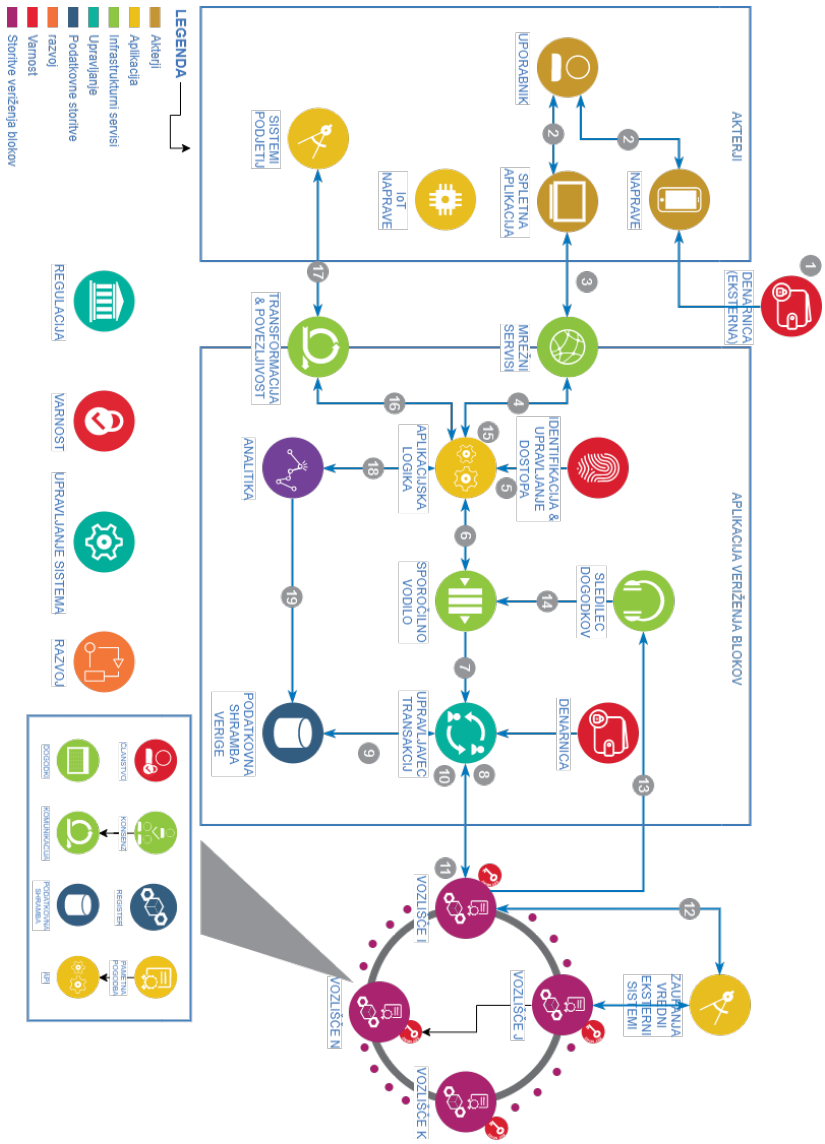
Zlasti omrežni operaterji, pa tudi vsi, ki se zanašajo na WAN omrežja morajo obvladati terminologijo in simbologijo naprav, ki so v teh omrežjih v rabi (Slika 62). Podobno kot pri načrtu IKT infrastrukture organzacij se tudi tu ne spuščamo v podrobnosti, ampak modeliramo predvsem povezave obstoječih z novimi funkcionalnimi enotami. Takšni diagrami so pomembni tudi iz vidika načrtovalcev in vzdrževalcev omrežne infrastrukture v bodočih pametnih mestih, saj jim pomagajo graditi in obvladovati to heterogeno omrežje, v katerem ima vsaka nova entiteta svoje mesto in vlogo.

V zvezi s pametnimi mesti se pojavljajo tudi aplikacije, ki vključujejo IoT naprave (npr. Slika 63). Za razliko od zgoraj naštetih diagramov pri načrtovanju takšnih aplikacij ne načrtujemo samo IKT opreme, ampak uvajamo funkcionalne programsko-aparaturne sklope, ki imajo svojo vlogo – implementirajo določen servis – v aktualni aplikaciji. Koristno za vse načrtovalce tovrstnih aplikacij je poznavanje ustrezne simbolike, ki sicer ni standardizirana, vendar je pričakovati, da bo to sledilo njeni širši uporabi.

Pozoren bralec bo ugotovil, da so le redki od pričujočih formalizmov dejansko standardizirani, kar ob hitri rasti in razvoju IT ni presenetljivo. Pogosto se je že zgodilo, da je bila določena notacija zastarela še preden je njen standard doživel ustrezno zrelost. Zato je tukaj dopustna določena ”svoboda” v izražanju, dokler so načrtovane komponente enolično razpoznavne. Po drugi strani dobro poznavanje določene notacije omogoča tudi enostavno privzemanje standardne notacije, ko je le-ta na voljo.




Slika 62: Telekomunikacijsko omrežje
Vir: lasten, 2023




Slika 63: IoT aplikacije
Vir: lasten, 2023

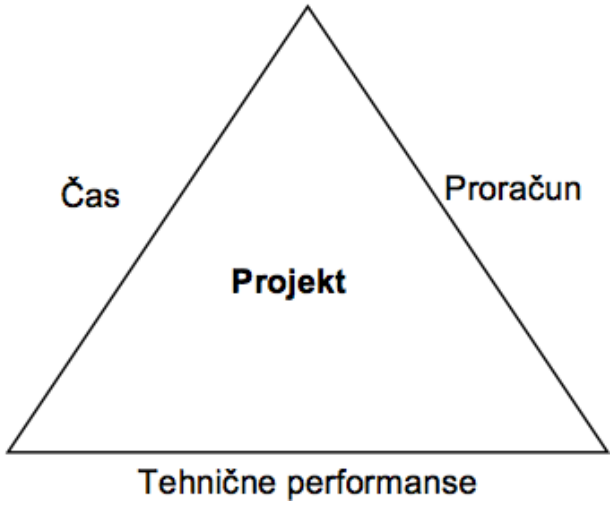
## 7.2 Planiranje aktivnosti

Običajni organizacijski model pri planiranju aktivnosti v poslovnem svetu je projektni model, ki omogoča največjo fleksibilnost združevanja zaposlenih pri opravljanju skupnih nalog – mrežno organiziranost. Kot sodelavci v projektu lahko nastopajo tudi projektni partnerji – zunanji sodelavci ali zaposleni partnerskih podjetij. Gre za posameznike in organizacije, ki so izkazali interes (investirali vire – ljudi ali sredstva) v projekt kot celoto ali katero od njegovih aktivnosti. Ker ima njihov interes za posledico izvedbo neke aktivnosti ali odsotnost le-te, je pomembno, da se njihove vloge zavedamo in jo pri načrtovanju upoštevamo.

Da bi bolje razumeli koncepte in postopke planiranja (projektnih) aktivnosti, se bomo z njimi pobližje seznanili. Še prej pa bomo predstavili osnovne mejnike v okviru življenjskega cikla projekta:

1. Konceptualna faza: določitev ciljev in zadolžitev odgovornega nosilca projekta
2. Faza planiranja: odgovorni nosilec identificira sodelavce, izdela razčlenitev dela (WBS), grafikon odgovornosti (LRC) in časovnico projekta (Ganttov diagram)
3. Faza izvedbe: zbiranje podatkov, sredstev, implementacija, verifikacija in validacija ter analiza rešitve
4. Zaključna faza: evalvacija rezultatov projekta

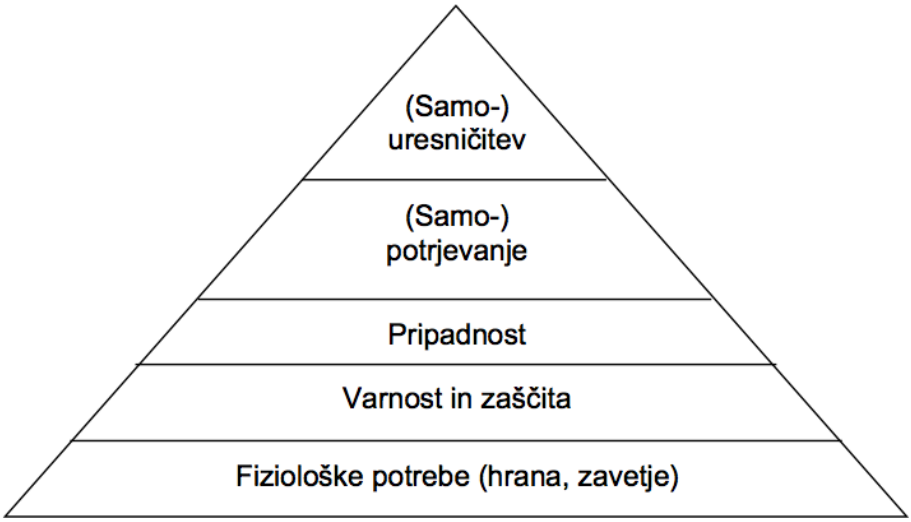
Pričakovane cilje oz. *tehnične performanse* (rezultata) projekta običajno specificiramo vnaprej. Te vplivajo predvsem na izbiro uporabljenih materialnih virov. Ko naredimo plan, *časovni potek* projekta spremljamo s pomočjo Ganttovih diagramov, kjer so vse aktivnosti združene na skupni časovni osi. *Projektni proračun* vključuje stroške vseh virov, ki so potrebni, da projekt izvedemo.




Slika 64: *Model projektnih parametrov*
Vir: lasten

Omenjene tri komponente so med sabo tesno povezane in tvorijo trikotni model projektnih parametrov (Slika 64). Optimalno stanje predstavlja pravokotni trikotnik, večja odstopanja pri kateremkoli parametru pa običajno pokvarijo simetrijo modela in vplivajo na druga dva parametra (npr. zamude bodo vplivale na stroške, ker bodo zahtevale dodatne vire za njihovo odpravo, ali pa bodo imele za posledico slabše tehnične performanse).

Faza izvedbe vključuje tudi motiviranje sodelavcev, ki v okviru projekta uresničujejo tudi svoje potrebe in ambicije. Potrebno jih je upoštevati, če želimo, da bodo svoje delo kakovostno in v rokih opravili. Maslow (Maslow, 1943) je zasnoval hierarhijo potreb sodelavcev na projektu (Slika 65), ki gradi na osnovnih potrebah in nadgradnji le-teh z željami in stremljenji. Višja stopnja motivacije se sklada z višjimi nivoji te hierarhije.




Slika 65: *Maslowa hierarhija potreb sodelavcev*
Vir: (Maslow, 1943)

### 7.2.1 Strukturirana razčlenitev dela

*Strukturirana razčlenitev dela (angl. Work Breakdown Structure, WBS)* je razporeditev projektnih aktivnosti v faze, aktivnosti, naloge in opravila. Pregledno predstavlja vse aktivnosti, ki jih bo potrebno izvesti za dosego ciljev projekta. Število nivojev je odvisno od kompleksnosti projekta ter samostojnosti sodelavcev pri izvedbi projektnih aktivnosti.

**Primer**: projekt izgradnje informacijskega sistema podjetja (Slika 66)

### 7.2.2 Linearni grafikon odgovornosti

*Linearni grafikon odgovornosti (angl. Linear Responsibility Chart, LRC)* za vsako osnovno aktivnost v projektu definira vlogo sodelujočih v projektu. Vsak sodelujoči lahko nastopa pri več aktivnostih in lahko ima več vlog. Osnovni tipi vlog so:

1. Formulacija problema
   1.1. Orientacija
   1.2. Opis problema in rešitve
   1.3. Cilji rešitve
2. Specifikacija sistema
   2.1. Opis arhitekture in funkcionalnih enot
   2.2. Opis strojnih komponent
   2.3. Opis omrežja
   2.4. Opis programskih modulov
      2.4.1. Opis vhodov in izhodov
      2.4.2. Opis funkcionalnosti
3. Analiza podatkov
   3.1. Zbiranje in urejanje podatkov
   3.2. Oblikovanje podatkovnega modela
   3.3. Zasnova podatkovne baze
4. Načrt strojne opreme
   4.1. Konfiguracija strežnikov
   4.2. Konfiguracija mreže
   4.3. Konfiguracija odjemalcev
5. Načrt programske opreme
   5.1. Načrtovanje primerov uporabe
   5.2. Načrtovanje podatkovnega modela
   5.3. Načrtovanje obnašanja in operacij
6. Vzpostavitev strojne opreme
   6.1. Zbiranje ponudb
   6.2. Izbira najboljšega ponudnika
   6.3. Instalacija in integracija opreme
7. Implementacija programske opreme
   7.1. Implementacija funkcionalnih modulov
   7.2. Implementacija uporabniških vmesnikov
   7.3. Implementacija vmesnikov do podatkovne baze
   7.4. Implementacija omrežnih vmesnikov
8. Verifikacija programske opreme
   8.1. Testiranje modulov
   8.2. Testiranje integracije
9. Validacija sistema
   9.1. Pregled specifikacije
   9.2. Pregled skladnosti načrta
   9.3. Testiranje skladnosti implementacije
10. Vzdrževanje sistema
   10.1. Načrtovanje vzdrževanja sistema
      10.1.1. Izdelava plana vzdrževanja za strojne komponente
      10.1.2. Izdelava plana vzdrževanja za programske komponente
      10.1.3. Predvidevanje sprememb in rasti IS
   10.2. Vzdrževanje sistema
      10.2.1. Odkrivanje in odprava napake
      10.2.2. Načrtovano vzdrževanje
      10.2.3. Načrtovana nadgradnja

Slika 66: *WBS za projekt IS*
Vir: lasten

**P** primarna odgovornost (angl. primary responsibility) odgovorna oseba za posamezno aktivnost, tako kot je odgovorni nosilec projekta odgovoren za cel projekt

**S** sekundarna odgovornost (angl. secondary responsibility) namestnik odgovorne osebe

**W** izvajalec (angl. work) posamezne aktivnosti je v manjših projektih običajno hkrati odgovorna oseba za aktivnost

**A** odobritev (angl. approval) je vloga, ki običajno pripade vodji aktivnosti oz. projekta – odločitev o tem ali je aktivnost oz. projekt ustrezno izveden

**R** pregled (angl. review) je vloga, ki jo večkrat dodelimo vodjem ali soodgovornim za posamezno aktivnost, lahko pa jo povezujemo tudi z odobritvijo na podlagi pregleda (nadzorna funkcija)

Glede na vrsto/kompleksnost projekta po potrebi lahko sodelavcem pripišemo dodatne vloge (npr. svetovalna, inšpekcijska, ipd.).

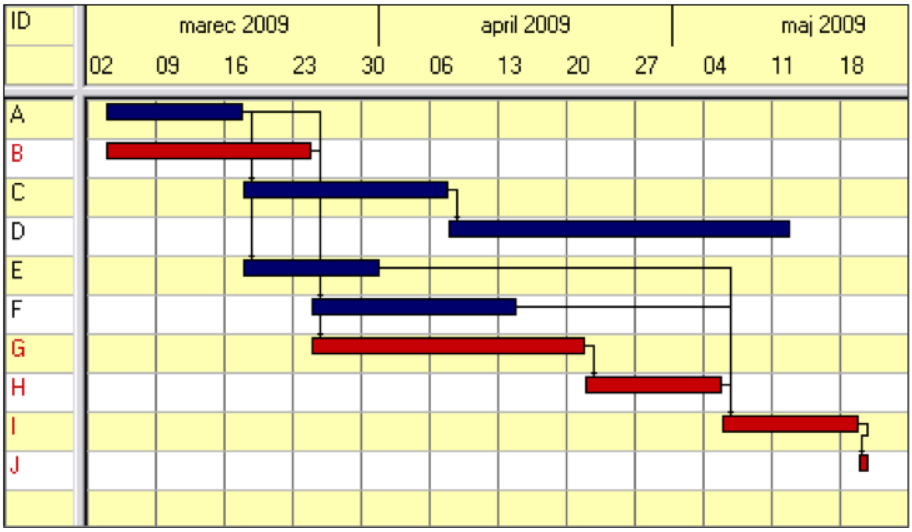
**Primer**: linearni grafikon odgovornosti za projekt izgradnje IS ob predpostavki manjšega tima, ki ga sestavljajo le vodja in dva strokovna sodelavca (Slika 67), pri čemer prvi pokriva predvsem strojno, drugi pa programsko opremo, vendar pri njuni integraciji sodelujeta.

### 7.2.3 Mrežni ali PERT diagram

Iz WBS in LRC sicer lahko razberemo aktivnosti znotraj projekta, vendar od tod ne moremo ugotoviti, kako so le-te med sabo povezane. Da bi jih lahko spravili na isto časovno os (kot na primer na Sliki 68), potrebujemo razen trajanja aktivnosti vsaj še podatek o njihovi medsebojni odvisnosti. Specifikacijo soodvisnosti med aktivnostmi nam omogoča metoda *PERT (Project Evaluation and Review Technique)*. Na podlagi metode PERT lahko specificiramo naslednje tipe soodvisnosti med aktivnostmi v projektu:

| | Vodja projekta | Sodelavec 1 | Sodelavec 2 |
|---|---|---|---|
| 1. Formulacija problema | | | |
| 1.1. Orientacija | P,W | S,R | |
| 1.2. Opis problema in rešitve | P,W | S,R | |
| 1.3. Cilji rešitve | P,W | S,R | |
| 2. Specifikacija sistema | | | |
| 2.1. Opis arhitekture in funkcionalnih enot | A | P,W | S,R |
| 2.2. Opis strojnih komponent | A | P,W | S,R |
| 2.3. Opis omrežja | A | P,W | S,R |
| 2.4. Opis programskih modulov | | | |
| 2.4.1. Opis vhodov in izhodov | A | S,R | P,W |
| 2.4.2. Opis funkcionalnosti | A | S,R | P,W |
| 3. Analiza podatkov | | | |
| 3.1. Zbiranje in urejanje podatkov | A | S,R | P,W |
| 3.2. Oblikovanje podatkovnega modela | A | S,R | P,W |
| 3.3. Zasnova podatkovne baze | A | S,R | P,W |
| 4. Načrt strojne opreme | | | |
| 4.1. Konfiguracija strežnikov | A | P,W | S,R |
| 4.2. Konfiguracija mreže | A | P,W | S,R |
| 4.3. Konfiguracija odjemalcev | A | P,W | S,R |
| 5. Načrt programske opreme | | | |
| 5.1. Načrtovanje primerov uporabe | A | S,R | P,W |
| 5.2. Načrtovanje podatkovnega modela | A | S,R | P,W |
| 5.3. Načrtovanje obnašanja in operacij | A | S,R | P,W |
| 6. Vzpostavitev strojne opreme | | | |
| 6.1. Zbiranje ponudb | P,W | | |
| 6.2. Izbira najboljšega ponudnika | P,W | | |
| 6.3. Instalacija in integracija opreme | A | P,W | S,R |
| 7. Implementacija programske opreme | | | |
| 7.1. Implementacija funkcionalnih modulov | A | S,R | P,W |
| 7.2. Implementacija uporabniških vmesnikov | A | S,R | P,W |
| 7.3. Implementacija vmesnikov do podatkovne baze | A | S,R | P,W |
| 7.4. Implementacija omrežnih vmesnikov | A | S,R | P,W |
| 8. Verifikacija programske opreme | | | |
| 8.1. Testiranje modulov | A | P,W | S,R |
| 8.2. Testiranje integracije | A | S,R | P,W |
| 9. Validacija sistema | | | |
| 9.1. Pregled specifikacije | A | P,W | S,R |
| 9.2. Pregled skladnosti načrta | A | S,R | P,W |
| 9.3. Testiranje skladnosti implementacije | A | P,W | S,R |
| 10. Vzdrževanje sistema | | | |
| 10.1. Načrtovanje vzdrževanja sistema | | | |
| 10.1.1. Izdelava plana vzdrževanja za strojne komponente | A | P,W | S,R |
| 10.1.2. Izdelava plana vzdrževanja za programske komponente | A | S,R | P,W |
| 10.1.3. Predvidevanje sprememb in rasti IS | P,W | S,R | S,R |
| 10.2. Vzdrževanje sistema | | | |
| 10.2.1. Odkrivanje in odprava napake | R | P,S,A | P,S,A |
| 10.2.2. Načrtovano vzdrževanje | R | P,S,A | P,S,A |
| 10.2.3. Načrtovana nadgradnja | R | P,S,A | P,S,A |

Slika 67: *LRC za projekt IS*
Vir: lasten




Slika 68: *Primer Ganttovega diagrama*
Vir: lasten

**konec-začetek** vzročno posledična zveza – predhodna aktivnost se mora zaključiti, da bi se lahko naslednja začela

**začetek-začetek** vzporedne aktivnosti – aktivnosti, ki se lahko istočasno začnejo oz. imajo iste predpogoje za izvedbo (npr., da so odvisne od zaključka istih aktivnosti)

**konec-konec** aktivnosti, ki se istočasno končajo – aktivnosti, katerih zaključek predstavlja nek skupen dogodek (npr. zaključek (faze) projekta)

V skladu z metodo PERT lahko povezanost aktivnosti grafično prikažemo z acikličnim usmerjenim grafom, ki mu večkrat rečemo kar *mrežni diagram (angl. net-diagram)*. Možna sta dva načina prikaza: z dogodkovnimi ali z aktivnostnimi mrežami.

Pri *dogodkovnih mrežah (angl. Activity On Arc, AOA)* so vozlišča dogodki, aktivnosti pa predstavljajo povezave med dogodki (Slika 69). Večkrat dogodke imenujemo tudi *miljski kamni (angl. milestone)*, ker označujejo pomembne mejnike v poteku projekta, v katerih se na osnovi realizacije odloča o nadaljnjih aktivnostih.

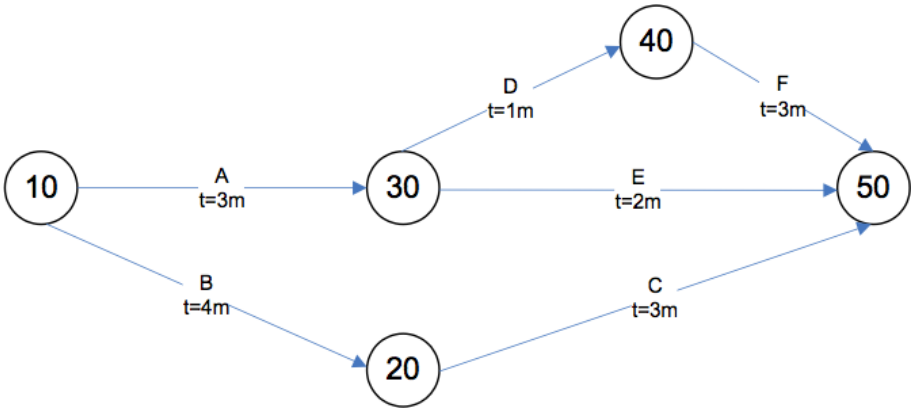
Dogodkovna mreža (Slika 69) je definirana z naslednjimi pojmi:

1. dogodki: 10, 20, 30,...
2. aktivnosti: A, B, C,... kjer
3. čas trajanja aktivnosti $t_{ij} \geq 0$,
4. *prirejeni viri $r_{ij} \geq 0$,
5. *stroški aktivnosti $s_{ij} \geq 0$,
6. lahko uvedemo tudi navidezne aktivnosti, kjer velja: $t_{ij} = 0, r_{ij} = 0, s_{ij} = 0$.

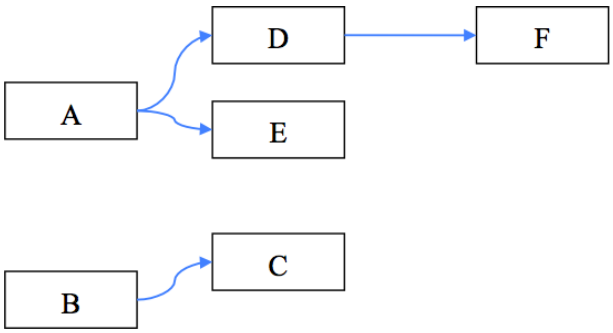
* Vsaki aktivnosti lahko razen trajanja priredimo tudi vire, ki jih potrebuje za svojo izvedbo (ljudi ali sredstva) in stroške, za primer, da želimo analizirati tudi ostale elemente projekta.

Pri *aktivnostni mreži (angl. Activity On Node, AON)* vozlišča predstavljajo aktivnosti, povezave pa določajo vrstni red aktivnosti. Aktivnostna mreža je definirana z naslednjimi pojmi (Slika 70):

1. aktivnosti: A, B, C,... kjer
2. čas trajanja aktivnosti $t_X \geq 0$,
3. prirejeni viri $r_X \geq 0$,
4. stroški aktivnosti $s_X \geq 0$.




Slika 69: *Primer dogodkovne mreže*
Vir: lasten




Slika 70: *Aktivnostna mreža za primer*
Vir: lasten

### 7.2.4 Mrežno planiranje

Aktivnosti projekta lahko na časovni osi razvrstimo v skladu z njihovim trajanjem in omejitvami s pomočjo metod kritične poti ali kritične verige. Običajno tehniki paroma uporabljamo v kombinaciji z metodo PERT, postopek pa imenujemo *mrežno planiranje (angl. network planing)*.

*Metoda kritične poti* (*angl. Critical Path Method*, *CPM*) je algoritem za razvrščanje projektnih aktivnosti in nam kot orodje omogoča učinkovito upravljanje projekta. Uporabna je v vsakem projektu s soodvisnimi aktivnostmi. Temelji na:

1. Strukturirani razčlenitvi dela (WBS)
2. Trajanjih aktivnosti
3. Odvisnostmi med aktivnostmi (PERT)

Od tod sestavimo model za CPM, ki omogoča izračun *kritične poti* skozi projektne aktivnosti. Ta nam določa trajanje projekta, ter *najzgodnejše/najkasnejše začetke/zaključke* posameznih aktivnosti.

**Določanje kritične poti** V projektu niso vse aktivnosti enako pomembne. Nekatere lahko zamujajo, ne da bi pri tem vplivale na čas dokončanja projekta. V vsakem mrežnem diagramu obstaja zaporedje aktivnosti, ki neposredno določa trajanje projekta. Tej poti pravimo *kritična pot (angl. critical path)*. Dolžina (trajanje) te poti je tisti čas, ki ga za projekt porabimo v vsakem (tudi najboljšem) primeru. Formalno je definirana na naslednji način:

***Kritična pot*** *je najdaljše nespremenljivo zaporedje aktivnosti, ki so vzročno-posledično povezane in njihovega trajanja po eni strani ne moremo skrajšati, po drugi pa bi vsaka zakasnitev aktivnosti s te poti pomenila tudi zakasnitev projekta.*

CPM nam pomaga določiti prioritete aktivnosti (ključne aktivnosti). Najvišje imajo seveda aktivnosti na kritični poti – tem po potrebi dodajamo vire, da tako zagotovimo njihovo pravočasno izvedbo. Nekritične aktivnosti imajo nižjo prioriteto, saj čas njihovega zaključka lahko variira.

**Časovna analiza dogodkovnih mrež** Elementi dogodkovne mreže imajo naslednjo strukturo (Slika 71):

**i** začetni dogodek aktivnosti i-j

**j** zaključni dogodek aktivnosti i-j

$t_{ij}$ čas trajanja aktivnosti i-j

$t_i^{(0)}$ najzgodnejši rok nastopanja dogodka i

$t_j^{(0)}$ najzgodnejši rok nastopanja dogodka j

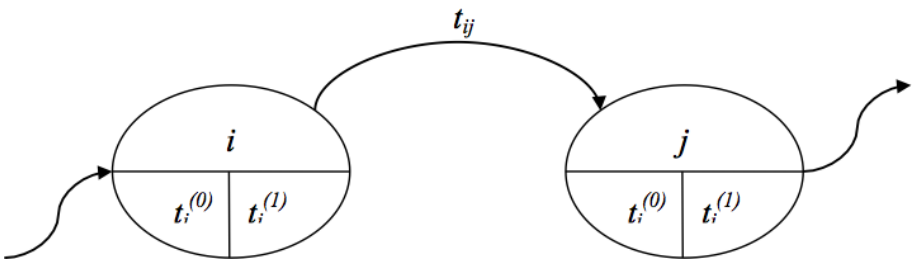
$t_i^{(1)}$ najkasnejši rok nastopanja dogodka i

$t_j^{(1)}$ najkasnejši rok nastopanja dogodka j

Poglejmo, kako bi izračunali kritično pot v dogodkovni mreži.

1. Za prvi dogodek velja, da je $t_1^{(0)}$=0, za zadnji n-ti dogodek pa $t_n^{(0)}$=$t_n^{(1)}$
2. Da bi izračunali najkrajši možni čas trajanja projekta, moramo po mreži progresivno izračunati najzgodnejše roke za izvedbo aktivnosti s pomočjo naslednje enačbe:

$$t_j^{(0)} = \max\left\{t_i^{(0)} + t_{ij}\right\}$$




*Slika 71: Element dogodkovne mreže*
Vir: lasten

3. Po mreži se premikamo od prvega dogodka (začetka) proti zadnjemu (zaključku) in na vsaki povezavi izvajalnemu času prištejemo trajanje aktivnosti, da določimo čas njenega zaključka; če v neko vozlišče vodi več poti, potem v tem vozlišču kot najzgodnejši začetek upoštevamo najkasnejši zaključek vseh aktivnosti, ki imajo ponor v tem vozlišču

4. Ko izračunamo najzgodnejše začetke dogodkov in s tem dobimo najkrajši čas trajanja projekta, lahko določimo tudi najkasnejše začetke dogodkov; te določamo retrogradno po enačbi:

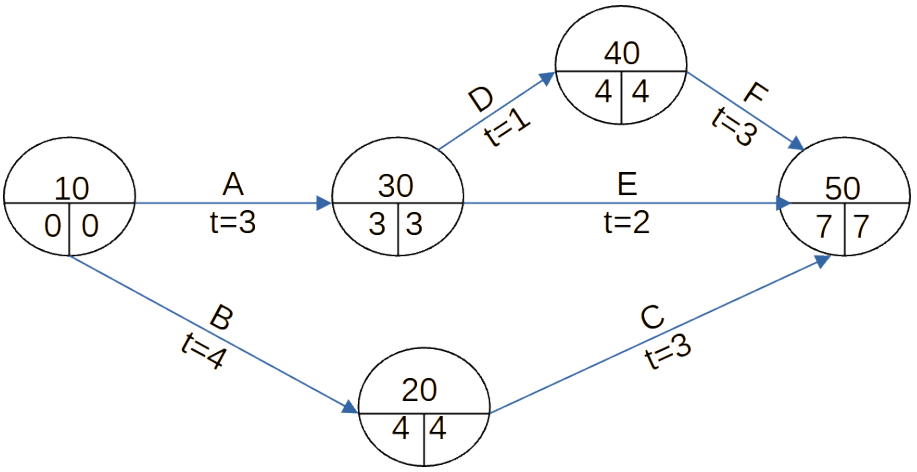
$$t_i^{(1)} = \min\left\{t_j^{(1)} - t_{ij}\right\}$$

5. Velja, da je $t_n^{(1)}$=$t_n^{(0)}$, kjer je $n$ število vseh dogodkov

6. V diagramu se vračamo po vseh možnih poteh od končnega dogodka proti začetku projekta, tako da v vozliščih od zaključkov odštevamo trajanja aktivnosti; kjer imamo zaradi več vstopnih poti v vozlišče, več različnih rezultatov, upoštevamo najzgodnejši čas

Kritična aktivnost je tista, za katero velja, da je:

- $t_i^{(0)} = t_i^{(1)}$




Slika 72: *Časovna analiza dogodkovne mreže*
Vir: lasten

- $t_j^{(0)} = t_j^{(1)}$
- $t_j^{(1)} = t_i^{(0)} + t_{ij}$

**Primer**: časovno analizo primera (Slika 69) podaja Slika 72. Rezultat analize:

- Imamo 2 kritični poti: (B,C), (A,D,F); $T_{min}$=7.
- Aktivnost E ni kritična in čas njenega začetka lahko variira 2 časovni enoti.

**Časovna analiza aktivnostnih mrež** Elementi aktivnostne mreže imajo naslednjo strukturo (Slika 73):

**X** oznaka aktivnosti

| **X** | | |
|---|---|---|
| $t(X)$ | $t_{ES}(X)$ | $t_{EF}(X)$ |
| $D(X)$ | $t_{LS}(X)$ | $t_{LF}(X)$ |

Slika 73: *Element aktivnostne mreže*
Vir: lasten

$t(X)$ čas trajanja aktivnosti X

$t_{ES}(X)$ najzgodnejši rok začetka aktivnosti X

$t_{EF}(X)$ najzgodnejši rok dokončanja aktivnosti X

$t_{LS}(X)$ najkasnejši rok začetka aktivnosti X

$t_{LF}(X)$ najkasnejši rok dokončanja aktivnosti X

$D(X)$ časovna rezerva oz. drsenje aktivnosti X

Določanje najzgodnejših rokov aktivnosti (Slika 73) poteka progresivno po naslednjem zaporedju:

1. Prvi aktivnosti A pripišemo najzgodnejši rok začetka aktivnosti $t_{ES}(A) = 0$ in izračunamo najzgodnejši rok konca aktivnosti $t_{EF}(A) = t_{ES}(A) + t(A)$

2. Najzgodnejši rok začetka ostalih aktivnosti določimo po naslednji enačbi:

$$t_{ES}(V) = \max_{U \in P} \{t_{ES}(U) + t(U)\}$$

   kjer je $P$ množica vseh neposrednih predhodnikov aktivnosti $V$

3. Najzgodnejši rok zaključka aktivnosti določimo na osnovi najzgodnejših rokov izvedbe teh aktivnosti $t_{EF}(X) = t_{ES}(X) + t(X)$

Najkasnejše roke nastopanja aktivnosti pa določimo zopet retrogradno in sicer:

1. Za zadnje aktivnosti Z velja: $t_{LF}(Z) = t_{EF}(Z) = t_{projekta}$
2. Nato določimo najkasnejši rok začetka zadnje aktivnosti Z: $t_{LS}(Z) = t_{LF}(Z) - t(Z)$
3. Najkasnejše roke dokončanja ostalih aktivnosti določimo po naslednji enačbi:
$$t_{LF}(U) = \min_{V \in N} \{t_{LF}(V) - t(V)\},$$
kjer je $N$ množica vseh neposrednih naslednikov aktivnosti $U$
4. Najkasnejše roke začetkov aktivnosti pa izračunamo po enačbi: $t_{LS}(X) = t_{LF}(X) - t(X)$

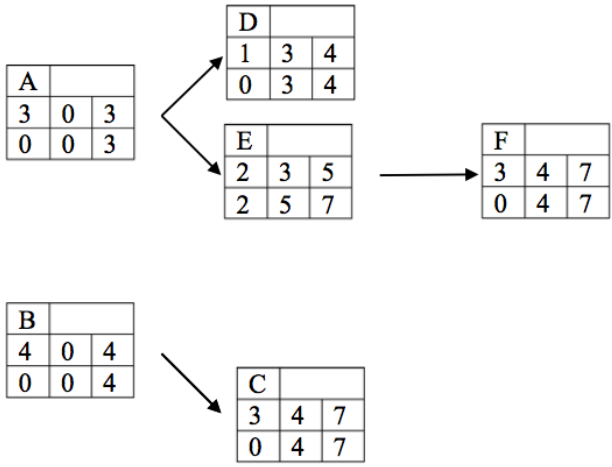
Časovno rezervo oz. drsenje aktivnosti X lahko izračunamo po enačbi:

$$D(X) = t_{LS}(X) - t_{ES}(X) = t_{LF}(X) - t_{EF}(X)$$

Če je drsenje *D(X) = 0*, je aktivnost X *kritična aktivnost*.

**Primer**: časovno analizo primera na Sliki 69 podaja Slika 74. Bolj pregledno jo lahko predstavimo tabelarično (Tabela 9).

Ker se projektni urnik lahko spreminja, nam CPM omogoča njegovo spremljanje med izvajanjem projekta. Na kateri koli točki lahko ponovno izdelamo in preverimo časovnico. Pri tem se lahko zgodi tudi, da zaradi večje zakasnitve kakšne nekritične aktivnosti, ali z njo povezanih aktivnosti, le-ta preide na kritično pot. Zato je koristno spremljanje izvajanja aktivnosti in preračunavanje kritične poti kadarkoli pride do zakasnitve kakšne aktivnosti.




Slika 74: *Časovna analiza aktivnostne mreže*
Vir: lasten

Tabela 9: *CPM analiza aktivnostne mreže na Sliki 74*

| Aktivnost | Trajanje | Predhodnik | ES | EF | LS | LF |
|---|---|---|---|---|---|---|
| A | 3 | - | 0 | 3 | 0 | 3 |
| B | 4 | - | 0 | 4 | 0 | 4 |
| C | 3 | B | 4 | 7 | 4 | 7 |
| D | 1 | A | 3 | 4 | 3 | 4 |
| E | 2 | A | 3 | 5 | 5* | 7 |
| F | 3 | D | 4 | 7 | 4 | 7 |

* nekritična aktivnost

**Časovna analiza mrež z omejitvami virov** V osnovi CPM omogoča le upoštevanje časovnih in precedenčnih omejitev aktivnosti. Metodo so razširili, da bi omogočili še upoštevanje odvisnosti aktivnosti od virov in jo združili v okviru *metode kritične verige* (*CCPM - Critical Chain Method*). S tem skušamo preprečiti zakasnitev projekta zaradi odsotnosti/pomanjkanja virov.

***CCPM*** *je metoda, ki ob upoštevanju zahtev aktivnosti po virih le-te razvrsti tako, da jim bodo v času izvajanja na voljo.*

***Kritična veriga*** *(angl. critical chain) je definirana kot zaporedje precedenčno in glede na zahtevane vire porazdeljenih aktivnosti, ki jih zaradi omejenih virov ne moremo izvršiti v krajšem času.*

Če so viri vedno na voljo v neomejeni količini, sta kritična pot in kritična veriga identični.

Poglavitne razlike med CCPM in CPM so:

1. upoštevanje (pogosto implicitnih) omejitev virov (npr. sodelujoči pri sicer neodvisnih aktivnostih so isti ljudje)

2. ne iščemo optimalne rešitve – zadošča "dovolj dobra" rešitev, saj je negotovost ocen trajanja aktivnosti mnogo večja od potencialne razlike med obema rešitvama

3. identifikacija in vstavljanje rezerv:

   (a) projektna rezerva

   (b) vhodna rezerva

   (c) rezerva virov

4. nadzorovanje porabe rezerv namesto opazovanja pravočasne izvedbe opravil

Načrt projekta izdelamo tako kot pri CPM – “z načrtovanjem od konca” določimo najkasnejše možne začetke aktivnosti. Za vsako aktivnost vodimo dva časa trajanja: po ”najboljši oceni” (50% gotovost) in ”varen” (90% ali 95% gotovost zaključka aktivnosti, odvisno od stopnje tveganja, ki si jo lahko privoščimo).

CCPM analiza:

1. vsaki aktivnosti dodelimo vire ter načrt izvajanja aktivnosti uravnotežimo z njimi, upoštevajoč čase trajanja po ”najboljši oceni”; najdaljše zaporedje aktivnosti z omejitvami virov, ki vodijo od začetka do konca projekta, identificiramo kot kritično verigo

2. rezerve uporabimo za določitev miljskih kamnov, kjer preverjamo stanje projekta glede na predvideno trajanje aktivnosti in izrabo virov

3. časovne rezerve projekta zberemo v rezervi na koncu projekta (projektna rezerva) kakor tudi na koncu vsakega zaporedja opravil, ki vodi v kritično verigo (vhodna rezerva)

4. končno določimo *začetni projektni plan (angl. baseline plan)*, kar nam omogoča tudi določitev in kasnejši nadzor projektnih financ, ter s tem ”fiksiramo kritično verigo”. Naknadne spremembe brez soglasja vodje projekta niso dovoljene, saj so vse predpostavke glede izrabe virov vezane na ta plan.

Prednosti CCPM so:

1. vire v kritični verigi izkoriščamo 100%, saj so bili planirani glede na ”najboljšo oceno”; zato se v izogib nepotrebnim zakasnitvam z njimi povezane aktivnosti izvajajo brez nepotrebnih prekinitev (npr. zaradi vzporednega izvajanja drugih aktivnosti)

2. sprotno pri miljskih kamnih nadzorujemo porabo rezerv; ker je nemogoče pričakovati pravočasno izvedbo vseh aktivnosti, lahko tukaj uvedemo nadzor in odločanje o korekcijskih mehanizmih (na primer dodajanje virov, alternativne aktivnosti, ki so glede na uporabo virov manj zahtevne, ipd.)

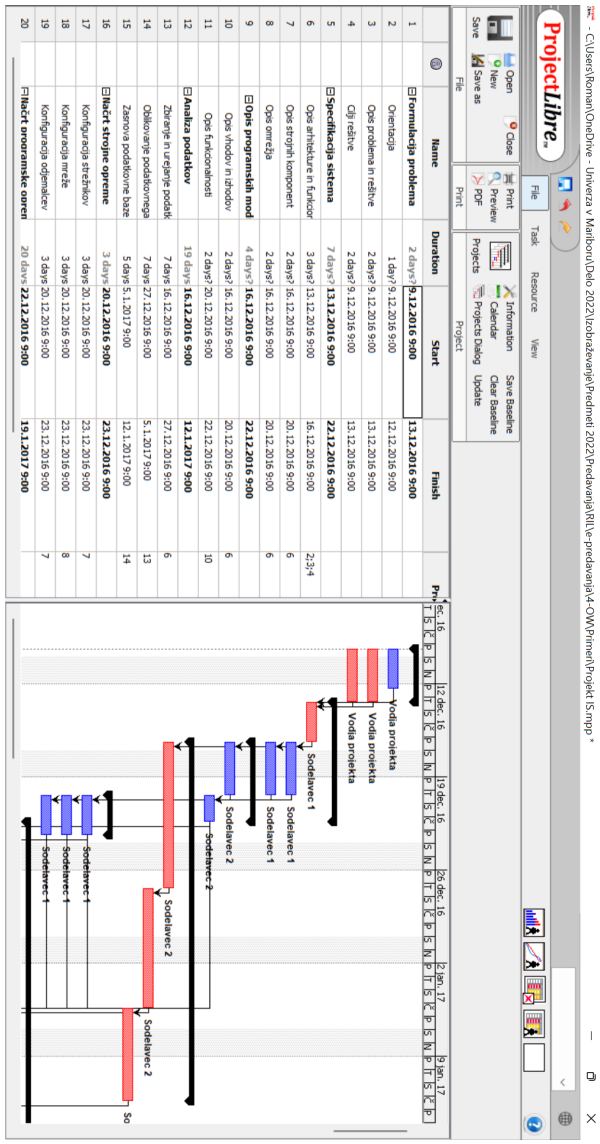
### 7.2.5 Časovnica

S *časovnico projektnih aktivnosti (Ganttovim diagramom)* specificiramo njihov začetek, trajanje in vrstni red. Med izvajanjem projekta omogoča pregled nad njegovim napredkom, pa tudi spremljanje stanja posameznih aktivnosti.

Ganttov diagram je vrstični palični diagram, ki predstavlja trajanje neke aktivnosti ali več povezanih aktivnosti (npr. faze in aktivnosti projekta, opravila v računalniku, postopki proizvodnega procesa, ipd.). Sestavimo ga tako, da narišemo X in Y os (kot za dvodimenzionalen koordinatni sistem). Na X-os nanesemo časovno lestvico z natančnostjo najkrajše projektne aktivnosti in v dolžini, ki približno ustreza predvidenemu trajanju projekta. Glede na področje uporabe, je časovna lestvica lahko zelo različna (npr. v logistiki običajno merimo ure, v proizvodnji minute ali celo sekunde, če pa merimo trajanje računalniških procesov ali komunikacij, pa so to običajno milisekunde). Imena oz. kratice postopkov (aktivnosti), ki so potrebni za izvedbo naloge (projekta) nanizamo na Y-os.

S tem, ko aktivnosti razporedimo po skupni časovni osi, ugotovimo predvideno skupno trajanje projekta. Glede na precedenčne odvisnosti med aktivnostmi in vire, ki jih le-te zahtevajo, lahko ta čas zmanjšamo s tem, da vzporedno izvajamo aktivnosti, ki so med sabo neodvisne.

Namen tega poglavja ni bil podrobno obravnavati upravljanje projektov, saj gre pri tem za širšo tematiko. Zainteresirani bralec lahko več informacij o tem pridobi v (Kerzner, 2003) in (Institute, 2003). Kljub temu inženirji logistike




Slika 75: *Primer Ganttov-vega diagrama za projekt IS*
Vir: lasten, zajem zaslona, 2023

zraven planiranja operacij, ki smo ga predstavili v predhodnem poglavju, potrebujejo tudi znanje iz planiranja aktivnosti, saj na ta način lažje konsistentno obvladujejo izvajanje projektov uvajanja in prenove informacijskih sistemov v logistiki.

## 7.3 Poslovna inteligenca

*Poslovna inteligenca (angl. Business Intelligence, BI)* zajema vse strategije in tehnologije, ki jih podjetja uporabljajo za podatkovne analize in upravljanje poslovnih informacij. Podprta je s *sistemi znanja in razvoja (angl. Knowledge Management System, KMS)*, ki v strukturi LIS predstavljajo tisti del, ki strokovnjakom za različna področja omogoča opravljati svetovalno in razvojno funkcijo na vseh nivojih upravljanja.

Sistemi znanja in razvoja obsegajo:

- Poslovno analitiko (BA)
- Sisteme za podporo odločanju (DSS)
- Inženirstvo na osnovi znanja (KBE)

Namen tega poglavja je izpostaviti vlogo računalniško podprtih sistemov znanja in razvoja ter seznaniti študente s tipskimi scenariji njihove uporabe za potrebe planiranja v logističnih aplikacijah.

### 7.3.1 Poslovna analitika

*Podatkovno rudarjenje (angl. Data Mining, DM)* je postopek iskanja anomalij, vzorcev in korelacij v večjih naborih podatkov, da bi lahko predvideli izide. Poslovna analitika, kot nadgradnja DM, predstavlja sistematičen postopek analize poslovnih podatkov, njihovega vrednotenja in vizualiacije s ciljem predvideti izide različnih poslovnih scenarijev.

*Poslovna analitika (angl. Business Analytics, BA)* je proces, ki na osnovi BI omogoča nov vpogled v poslovanje in boljše strateško odločanje za prihodnost.

Proces poslovne analitike zajema:

1. agregacijo podatkov (pred analizo morajo biti podatki zbrani, organizirani in filtrirani skozi planske ali transakcijske podatke)
2. podatkovno rudarjenje (preiskuje velike količine podatkov iz podatkovnih baz in jih obdeluje s pomočjo statističnih metod in strojnega učenja, da bi ugotovilo relacije in trende)
3. asociacije in kavzalnost (identifikacija predvidljivih "kaj-če" in "kaj-potem" kriterijev)
4. rudarjenje besedil (preiskovanje večjih količin nestrukturiranih besedil z namenom kvalitativne in kvantitativne analize)
5. napovedi (analizirajo zgodovinske podatke iz določenega obdobja, da bi lahko predvideli bodoče dogodke in obnašanje)
6. prediktivno analitiko (različne statistične metode za izdelavo modelov, ki na podlagi podatkov iz naborov identificirajo vzorce in izdelajo lestvico možnih izidov)
7. optimizacijo (na podlagi ugotovljenih trendov in napovedi izdelamo različne scenarije in uporabimo simulacijske metode za vrednotenje in izbiro najboljših)
8. vizualizacijo podatkov (vizualna predstavitev izidov v obliki tabel in grafikonov za hitro in pregledno analizo)

**Primer**: analiza prodaje

Imamo podjetje, ki prodaja več tipov izdelkov preko različnih prodajnih mest. Prodajne transakcije beležimo zaradi analize prodaje po izdelkih, prodajnih mestih in terminski porazdelitvi. Shranjujemo jih v CSV (Comma Separated Values) formatu, ki ga lahko obdelujemo z elektronskimi preglednicami.

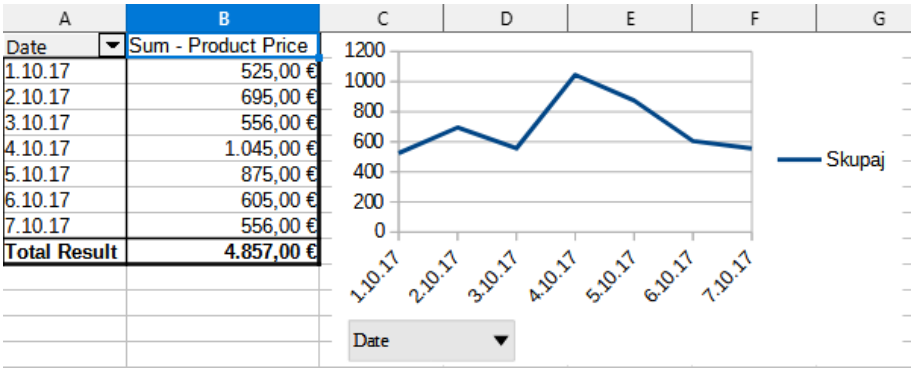
Zbrane transakcijske podatke (Slika 76) najprej filtriramo, da ugotovimo, ali so celoviti in pravilno formatirani. Šele nato se lahko lotimo njihove statistične obdelave, saj bi nam sicer manjkajoči podatki lahko pokvarili vzorec, napačno formatirani podatki pa povzročili računske napake.

Običajno želimo ustvariti smiselne vpoglede glede na zbrane "surove podatke", ki sami po sebi ne predstavljajo dodane vrednoti pri odločanju. V ta namen lahko ustvarimo *vrtilne tabele (angl. pivot table)*, ki nam omogočajo grupiranje podatkov okoli okoli izbranih atributov in njihovo statistično obdelavo. Najpreprostejši, vendar pogosto tudi najbolj uporabni, so vzorci, ki jih tvorimo na podlagi grupiranja okoli enega izbranega atributa:

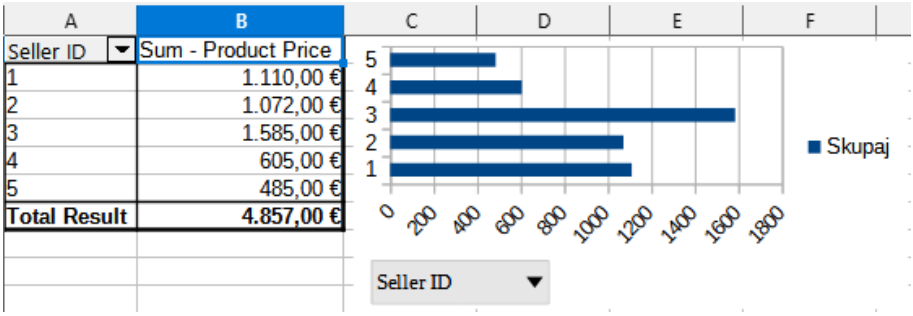
- Statistika prodaje po dnevih (Slika 77) nam nakaže gibanje prodaje v izbranem časovnem obdobju; na podlagi tega lahko sklepamo na sezonska nihanja in ustrezno planiramo zaloge
- Statistika prodaje po prodajnih mestih (Slika 78) nam pokaže katera prodajna mesta ustvarijo največ prometa
- Statistika prodaje po proizvodih (Slika 79) nam pove, kateri izdelki so najbolj iskani oz. predstavljajo največji delež v naši prodaji
- Pregledna tabela s podatki o prodaji po dnevih, prodajnih mestih, transakcijah in kupcih (Slika 80) nam omogoča boljši vpogled v strukturo naših podatkov; na podlagi takšne tabele lahko na primer retrogradno ugotavljamo, na kateri dan in na katerem prodajnem mestu je bila realizirana prodaja določenega izdelka določenemu kupcu

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| **Date** | **Seller ID** | **Customer ID** | **Transaction ID** | **Product ID** | **Product Price** |
| 1.10.17 | 1 | 12 | 1 | 101 | 195,00 € |
| 1.10.17 | 1 | 12 | 1 | 102 | 45,00 € |
| 1.10.17 | 1 | 12 | 1 | 103 | 35,00 € |
| 1.10.17 | 2 | 14 | 2 | 104 | 55,00 € |
| 1.10.17 | 2 | 14 | 3 | 101 | 195,00 € |
| 2.10.17 | 3 | 15 | 4 | 105 | 85,00 € |
| 2.10.17 | 3 | 15 | 4 | 101 | 195,00 € |
| 2.10.17 | 3 | 15 | 4 | 103 | 35,00 € |
| 2.10.17 | 3 | 16 | 5 | 104 | 55,00 € |
| 2.10.17 | 1 | 17 | 6 | 101 | 195,00 € |
| 2.10.17 | 1 | 17 | 6 | 102 | 45,00 € |
| 2.10.17 | 1 | 17 | 6 | 105 | 85,00 € |
| 3.10.17 | 2 | 18 | 7 | 106 | 35,00 € |
| 3.10.17 | 2 | 18 | 7 | 107 | 65,00 € |
| 3.10.17 | 2 | 18 | 7 | 108 | 86,00 € |
| 3.10.17 | 4 | 19 | 8 | 105 | 85,00 € |
| 3.10.17 | 4 | 19 | 8 | 101 | 195,00 € |
| 3.10.17 | 4 | 19 | 8 | 103 | 35,00 € |
| 3.10.17 | 4 | 19 | 9 | 104 | 55,00 € |
| 4.10.17 | 5 | 20 | 10 | 105 | 110,00 € |
| 4.10.17 | 5 | 20 | 10 | 106 | 125,00 € |
| 4.10.17 | 5 | 20 | 10 | 104 | 55,00 € |
| 4.10.17 | 5 | 20 | 10 | 101 | 195,00 € |
| 4.10.17 | 1 | 21 | 11 | 102 | 45,00 € |
| 4.10.17 | 1 | 21 | 11 | 105 | 85,00 € |
| 4.10.17 | 1 | 21 | 12 | 106 | 35,00 € |
| 4.10.17 | 3 | 12 | 13 | 103 | 35,00 € |
| 4.10.17 | 3 | 12 | 13 | 104 | 55,00 € |
| 4.10.17 | 3 | 12 | 13 | 105 | 110,00 € |
| 4.10.17 | 3 | 12 | 13 | 101 | 195,00 € |
| 5.10.17 | 1 | 22 | 14 | 107 | 35,00 € |
| 5.10.17 | 1 | 22 | 14 | 108 | 25,00 € |

Slika 76: *Transakcijski podatki o prodaji*
Vir: lasten, zajem zaslona, 2023

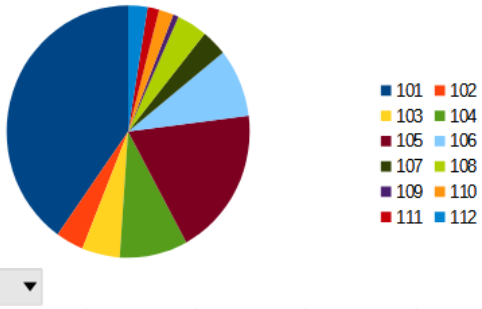



Slika 77: *Podatki o prodaji za pretekli teden*
Vir: lasten, zajem zaslona, 2023




Slika 78: *Podatki o prodaji po prodajnih mestih*
Vir: lasten, zajem zaslona, 2023

| Product ID | Sum - Product Price |
|---|---|
| 101 | 1.950,00 € |
| 102 | 180,00 € |
| 103 | 245,00 € |
| 104 | 440,00 € |
| 105 | 925,00 € |
| 106 | 425,00 € |
| 107 | 165,00 € |
| 108 | 197,00 € |
| 109 | 35,00 € |
| 110 | 95,00 € |
| 111 | 75,00 € |
| 112 | 125,00 € |
| **Total Result** | **4.857,00 €** |




Slika 79: *Podatki o prodaji po proizvodih*
Vir: lasten, zajem zaslona, 2023

| Date | Seller I | Customer I | Transaction I | Product | Vsota - Product Price |
|---|---|---|---|---|---|
| 1.10.17 | 1 | 12 | 1 | 101 | 195,00 € |
| | | | | 102 | 45,00 € |
| | | | | 103 | 35,00 € |
| | 2 | 14 | 2 | 104 | 55,00 € |
| | | | 3 | 101 | 195,00 € |
| 2.10.17 | 1 | 17 | 6 | 101 | 195,00 € |
| | | | | 102 | 45,00 € |
| | | | | 105 | 85,00 € |
| | 3 | 15 | 4 | 101 | 195,00 € |
| | | | | 103 | 35,00 € |
| | | | | 105 | 85,00 € |
| | | 16 | 5 | 104 | 55,00 € |
| 3.10.17 | 2 | 18 | 7 | 106 | 35,00 € |
| | | | | 107 | 65,00 € |
| | | | | 108 | 86,00 € |
| | 4 | 19 | 8 | 101 | 195,00 € |
| | | | | 103 | 35,00 € |
| | | | | 105 | 85,00 € |
| | | | 9 | 104 | 55,00 € |

Slika 80: *Pregled prodaje za pretekli teden*
Vir: lasten, zajem zaslona, 2023

### 7.3.2 Sistemi za podporo odločanju

*Sistem za podporo odločanju (angl. Decision Support System, DSS)* je interaktiven sistem, ki pomaga odločevalcem uporabiti podatke in modele za reševanje ne-strukturiranih ali delno-strukturiranih problemov. Pri tem lahko uporablja tudi metode podatkovnega rudarjenja, vendar je tu poudarek na modelu, ki podpira odločanje na osnovi rezultatov analiz. V ta namen se lahko poslužujemo različnih sistemov za obdelavo podatkov – od elektronskih preglednic do sistemov za upravljanje baz podatkov – odvisno od tega, kako so podatki shranjeni in strukturirani. Za preprostejšo uporabo se lahko poslužimo tudi programskih orodij, ki za nas zbirajo in strukturirano prikazujejo zbrane podatke iz zbirk podatkov našega informacijskega sistema (npr. Metabase, Power BI). V tem primeru govorimo o *ekspertnih systemih (angl. Expert System, ES),* saj gre za aplikacijske programe oz. okolja, ki učinkovito podpirajo reševanje problemov na specializiranih problemskih področjih, ki zahtevajo ekspertno znanje in spretnosti.

Modeli strukturiranja problemov, ki jih uporabljajo sistemi za podporo odločanju, so različni. Zanje je značilno, da na skupni imenovalec za potrebe odločanja združijo skupine različnih kritrijev (tako po vsebini kot po formatu). Tukaj bomo predstavili samo najpreprostejšega – večparametrsko odločanje – ker se pogosto uporablja predvsem pri nabavnih odločitvah, ki spremljajo infrastrukturne projekte implementacije informacijskih sistemov. Vrste podatkov, ki jih obdelujemo na podlagi izbranih kriterijev so:

- kvantitativni (količine, vrednosti, performančni podatki, ipd., ki omogočajo količinsko primerjavo inačic vzorca)
- kvalitativni (besedilni podatki, ki omogočajo odločanje o tem, v kolikšni meri so izpolnjeni določeni kriteriji in kvalitativno primerjavo inačic vzorca)
- binarni (”da/ne” podatki, ki omogočajo odločanje o tem ali posamezne inačice vzorca izpolnjujejo določen kriterij ali ne)

**Večparametrsko odločanje** *Večparametrsko odločanje (angl. Multi Criteria Decision Making, MCDM)* je pod-disciplina operacijskih raziskav, ki se ukvarja z analizo več (potencialno) nasprotujočih si kriterijev v poslovnem in vsakdanjem življenju.

Večparametrski odločitveni model (VPOM) sestavljajo:

- Kriteriji (izbor relevantnih parametrov, na podlagi katerih primerjamo inačice iz vzorca podatkov)
- Uteži (pripišemo jih posameznim atributom iz izbora parametrov glede na pomembnost)
- Funkcija koristi (angl. utility function) (vrednotenje naborov kriterijev, ki nam vrne enolično vrednost, ki omogoča primerjavo inačic med sabo)
- Podatki (predstavljajo inačice rešitev, ki jih med sabo primerjamo)

Postopek večparametrskega odločanja:

1. Predstavimo inačice: $V_i(P_{i,1}; P_{i,2}; ...P_{i,n}); i = 1..m$
2. Parametre normiramo tako, da izračunamo $p_{i,j}$ za vsak $P_{i,j}$ za vsak $j = 1..n$, pri čemer najprej izberemo maksimalni $P_{i,j}$ izmed vseh $i$ vzorcev za določen $j$:
   (a) $p_{i,j} = P_{i,j}/maxP_{i,j}$ če večja vrednost Pi,j predstavlja bolj koristno vrednost oz.
   (b) $p_{i,j} = 1 - P_{i,j}/maxP_{i,j}$ če nam manjša vrednost Pi,j predstavlja bolj koristno vrednost
3. Parametre utežimo: $x_{i,j} = p_{i,j} \cdot U_j$ za vsak $j = 1..n$, pri čemer sami določimo $U_j$ glede na naše preference (vsota $U_j$ mora biti 1 oz. 100%)
4. Izračunamo ocene inačic: $X_i = \sum x_{i,j}$ za vsako inačico $i = 1..m$ in vse njene parametre $j = 1..n$, da dobimo ocene naših inačic glede na utežitev parametrov

5. Izberemo najboljšo oceno: $Y = maxX_i$

**Primer**: izbor aktualno najboljše mobilne platforme z Android operacijskim sistemom za naše podjetje

V Tabeli 10 so najprej zbrani podatki o primerkih, ki smo jih identificirali kot ožji izbor. Te podatke razvrstimo po parametrih, ki smo jih izbrali kot relevantne za izbiro. V nadaljevanju zbrane podatke normiramo, da dobimo primerljive vrednosti, jih utežimo, da vrednostim, ki so za nas pomembnejše, damo večjo vrednost, ter jih seštejemo, da dobimo ocene primerkov.

Končni rezultat naše analize predstavlja zbirna tabela in grafikon na Sliki 81. Tu so zbrane in grafično prikazane ocene primerkov po kategorijah parametrov, izpostavljen pa je tudi primerek z najboljšo skupno oceno. Kot tudi v tem primeru, se pogosto izkaže, da to ni primerek, ki bi bil najboljši v vseh kategorijah, pač pa je v povprečju najboljši med vsemi primerki iz izbora glede na našo utežitev parametrov. To je tudi glavna dodana vrednost metode, saj bi nas izbor po posameznih kategorijah lahko zavedel.

### 7.3.3 Inženirstvo na osnovi znanja

*Inženirstvo na osnovi znanja (angl. Knowledge Based Engineering, KBE)*, kot nadgradnja *računalniško podprtega inženirstva (angl. Computer Aided Engineering, CAE)*, je metodologija, ki omogoča sistematično integracijo inženirskih izkušenj v načrtovanje sistemov. V ta namen kombinira različne metode, ki omogočajo ne samo načrtovanje in inženiring, ampak tudi oceno zmogljivosti in stroškov planiranih rešitev.

Metode inženirstva na osnovi znanja:

- Računalniško podprto *planiranje aktivnosti – vodenje projektov (angl. Project System, PS)*

Tabela 10: *Večparametrski odločitveni model za izbor mobilne platforme*

**VEČPARAMETRSKI ODLOČITVENI MODEL**
(Android Smartphone)

| Model | PARAMETRI | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Cena (€) | Ocena | Zmogljivosti | | | Lastnosti | | | Kamera |
| | | (1-10) | hitrost proc. (GHz) | RAM (GB) | razširitev pom. (GB) | teža (g) | velikost (mm^3) | kapaciteta bat. (mAh) | (1-10) |
| Huawei P10 Plus | 537 | 8,4 | 2,4 | 6 | 128 | 165 | 79500 | 3750 | 7 |
| Oneplus 5 | 441 | 8,7 | 2,45 | 8 | 128 | 153 | 83411 | 3300 | 9 |
| Nokia 8 | 580 | 9,1 | 2,5 | 4 | 64 | 160 | 89405 | 3090 | 7 |
| Galaxy Note 8* | 776 | 9,5 | 2,35 | 6 | 256 | 195 | 104533 | 3300 | 8 |

* priporočena cena proizvajalca

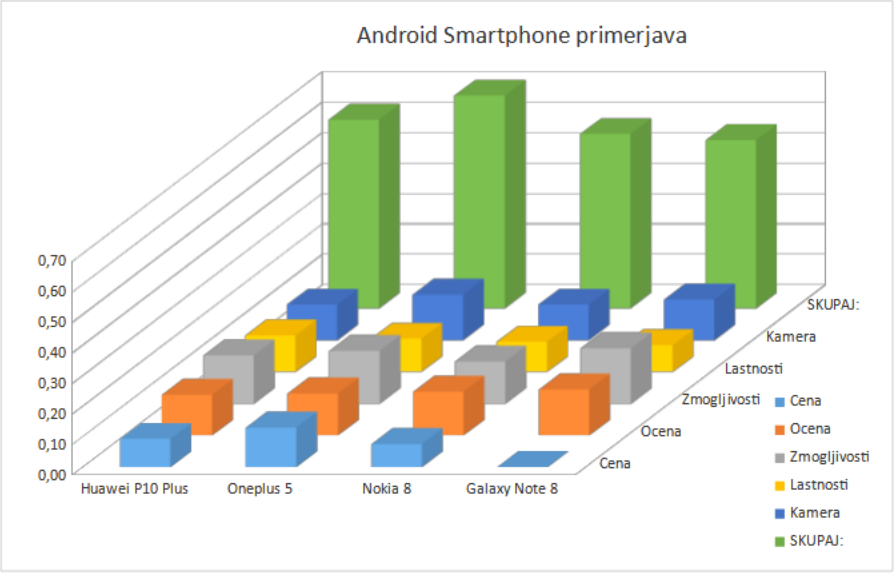
Vir www.testberichte.de

| | UTEŽITEV PARAMETROV | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Cena | Ocena | Zmogljivosti | | | Lastnosti | | | Kamera | |
| | | | hitrost proc. | RAM | razširitev pom. | teža | velikost | kapaciteta bat. | | |
| Utež | 30 % | 15 % | 10 % | 5 % | 5 % | 5 % | 5 % | 10 % | | 15 % |
| | | | | 20 % | | | 20 % | | | |

| Model | NORMIRANI PARAMETRI | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Cena | Ocena | Zmogljivosti | | | Lastnosti | | | Kamera | |
| | | | hitrost proc. | RAM | razširitev pom. | teža | velikost | kapaciteta bat. | | |
| Huawei P10 Plus | 0,31 | 0,88 | 0,96 | 0,75 | 0,50 | 0,15 | 0,24 | 1,00 | 0,78 | |
| Oneplus 5 | 0,43 | 0,92 | 0,98 | 1,00 | 0,50 | 0,22 | 0,20 | 0,88 | 1,00 | |
| Nokia 8 | 0,25 | 0,96 | 1,00 | 0,50 | 0,25 | 0,18 | 0,14 | 0,82 | 0,78 | |
| Galaxy Note 8 | 0,00 | 1,00 | 0,94 | 0,75 | 1,00 | 0,00 | 0,00 | 0,88 | 0,89 | |

| Model | KONČNA OCENA PARAMETROV | | | | | | | | | SKUPAJ: |
|---|---|---|---|---|---|---|---|---|---|---|
| | Cena | Ocena | Zmogljivosti | | | Lastnosti | | | Kamera | |
| | | | hitrost proc. | RAM | razširitev pom. | teža | velikost | kapaciteta bat. | | |
| Huawei P10 Plus | 0,09 | 0,13 | 0,10 | 0,04 | 0,03 | 0,01 | 0,01 | 0,10 | 0,12 | 0,62 |
| Oneplus 5 | 0,13 | 0,14 | 0,10 | 0,05 | 0,03 | 0,01 | 0,01 | 0,09 | 0,15 | 0,70 |
| Nokia 8 | 0,08 | 0,14 | 0,10 | 0,03 | 0,01 | 0,01 | 0,01 | 0,08 | 0,12 | 0,57 |
| Galaxy Note 8 | 0,00 | 0,15 | 0,09 | 0,04 | 0,05 | 0,00 | 0,00 | 0,09 | 0,13 | 0,55 |

Vir: lasten, zajem zaslona, 2023




| ZBIRNA TABELA OCEN PO SKUPINAH PARAMETROV | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Cena** | **Ocena** | **Zmogljivosti** | **Lastnosti** | **Kamera** | **SKUPAJ:** |
| Huawei P10 Plus | 0,09 | 0,13 | 0,16 | 0,12 | 0,12 | 0,62 |
| Oneplus 5 | 0,13 | 0,14 | 0,17 | 0,11 | 0,15 | 0,70 |
| Nokia 8 | 0,08 | 0,14 | 0,14 | 0,10 | 0,12 | 0,57 |
| Galaxy Note 8 | 0,00 | 0,15 | 0,18 | 0,09 | 0,13 | 0,55 |

| Izbira | 0,70 | Oneplus 5 |
|---|---|---|

Slika 81: *Izbor optimalne mobilne platforme*
Vir: lasten, zajem zaslona, 2023

- Računalniško podprto *načrtovanje (angl. Computer Aided Design, CAD)*, *proizvodnjo (angl. Computer Aided Manufacturing, CAM)* in *robotizirano proizvodnjo (angl. Computer Integrated Manufacturing, CIM)*
- Računalniško *modeliranje, simulacija in analiza (angl. Simulation Modeling and Analysis, SMA)*
- Računalniško podprto *podrobno planiranje proizvodnje (angl. Master Production Scheduling / Master Resource Planning; MPS / MRP)*

Življenjski cikel upravljanja znanja/izkušenj zajema njihovo:

1. Identifikacijo: ugotavljanje neskladja z zaželenim stanjem v proizvodnem procesu kot posledica slabo definiranega proizvoda ali procesa
2. Zajemanje: izkušnje so shranjene
3. Analizo: vzročna analiza zajete izkušnje, da bi definirali primerno strategijo odprave neskladja in ponovne uporabe izkušnje v izogib novim neskladjem
4. Shranjevanje: izsledki iz analize so arhivirani; na ta način so izkušnje shranjene
5. Iskanje in pridobivanje: izkušnjo poiščemo in pridobimo
6. Uporabo: element izkušnje uporabimo
7. Ponovno uporabo: ta faza zaključi cikel upravljanja izkušenj in začne novega

**Primer**: izboljšava zmogljivosti proizvodnje (Amdahlov zakon)

V proizvodnji imamo problem s pretočnostjo proizvodne linije, ki bi ga lahko rešili z vzporednim procesiranjem vhoda na dodatnih proizvodnih celicah. $P1 = 0,2$ (20% vhoda) lahko procesiramo na celici, ki je 10 krat ($N1 = 10$)

hitrejša od osnovne celice, $P2 = 0,4$ (40% vhoda) lahko izvajamo na celici, ki je 5 krat hitrejša od osnovne celice ($N2 = 5$), preostanek vhoda pa procesiramo na osnovni celici ($P3 = 1 - (P1 + P2) = 0,4$; $N3 = 1$). Kakšna je možna pospešitev glede na aktualni čas procesiranja vhoda?

Najprej izračunamo nov čas procesiranja vhoda $T'$ glede na predpostavke:

$$T' = P1/N1 + P2/N2 + P3/N3 = 0,02 + 0,08 + 0,4 = 0,5$$

Od tod lahko sklepamo na pospešitev:

$$S = 1/T' = 2$$

Razlaga: tudi če uporabimo bistveno hitrejše procesne komponente, ki pa ne morejo prevzeti večjega dela vhodnega bremena (npr. zaradi tipa obdelovancev ali drugih karakteristik proizvodnega procesa), pospešitev ni sorazmerna.

V našem primeru smo uporabili zgolj Amdahlov zakon, da smo ugotovili teoretično kapaciteto sistema po prenovi. Bolj natančne podatke in analizo zmogljivosti ob variabilni strukturi vhoda, pa tudi ob spremembah poti v proizvodnem postopku, bi dobili s simulacijo in razvrščanjem proizvodnih aktivnosti. Glede na to, da je to predmet predmetov v višjih letnikih te vsebine tukaj samo omenjamo in zainteresiranega bralca napotimo na ustrezne učne vsebine predmetov *Informacijska podpora logističnim sistemom in procesom* ter *Integracije logističnih informacijskih sistemov*.

# 8 Varnost informacijskih sistemov

V tem poglavju se bomo seznanili s področjem *varnosti informacijskih sistemov* oz. *informacijske varnosti*. Gre za področje, katerega pomen s širjenjem informacijske tehnologije na vsa področja življenja in ob naraščajočem pomenu zaupnosti shranjenih in po internetu posredovanih podatkov stalno narašča. S hitrim tehnološkim razvojem informacijskih tehnologij ter hitrim sprejemanjem in uvajanjem novih tehnologij naloga zagotavljanja varnosti informacijskih sistemov stalno pridobiva na pomenu, njena kompleksnost pa se hkrati stalno povečuje.

V zvezi z informacijsko varnostjo se pojavlja vprašanje etičnih izzivov pri uporabi informacijske tehnologije. Zakaj etičnih? Poenostavljeno rečeno, etika predstavlja pravilno oz. družbeno sprejemljivo obnašanje. Neetično ravnanje ne pomeni nujno nelegalnega delovanja. Zato se je že zgodaj v poslovnem svetu uveljavila dobra praksa uvajanja t.i. etičnih kodeksov. Uporaba IT je pred podjetja postavila naslednje etične izzive:

1. zasebnost (pridobivanje, hramba in distribucija osebnih podatkov posameznikov)
2. pravilnost podatkov (avtentičnost, zanesljivost in točnost pridobljenih in procesiranih podatkov)
3. avtorske pravice (lastništvo in vrednost informacij)
4. dostop do podatkov (kdo ima dostop do podatkov, v kakšnem obsegu in ali je dostop plačljiv)

*Informacijska zasebnost* pomeni pravico, da določimo kdaj in do kakšne mere lahko pridobivamo in prenašamo informacije o posamezniku; zato nas bo s tem v zvezi zanimalo naslednje:

1. Kje so podatki shranjeni?

2. Ali so podatki točni?
3. Ali lahko netočne podatke spremenimo?
4. Kako spreminjamo podatke?
5. Kako dolgo se naši podatki hranijo?
6. Kdo ima dostop do naših podatkov?
7. Ali je zagotovljena varnost podatkov?
8. Pod kakšnimi pogoji se delijo naši podatki?
9. Komu in pod kakšnimi pogoji so posredovani naši podatki?

*Kodeks zasebnosti* v podjetju je nabor ukrepov, ki naj zagotovijo varnost podatkov kupcev, dobaviteljev, uporabnikov in zaposlenih. Z mednarodnim vidikom zasebnosti se ukvarjajo organizacije, katerih informacijski kanali se razprostirajo izven meja ene države.

## 8.1 Informacijska varnost

Kaj torej je informacijska varnost oz. varnost informacijskega sistema?

***Varnost IS*** *je sklop aktivnosti in ukrepov, za zagotavljanje normalnega delovanja IS brez rušenja njegove integritete.*

*Varnost informacijskega sistema* zajema skupino ukrepov in aktivnosti, ki naj zagotovijo njegovo normalno delovanje, preprečijo zlorabo njegovih virov ter zagotovijo avtoriziran dostop do podatkov. Vsi ukrepi in aktivnosti za njeno zagotavljanje morajo potekati brez rušenja integritete informacijskega sistema. Večje kot je tveganje, bolj rigorozni zaščitni ukrepi so potrebni. Tveganje na podlagi katerega določamo stopnjo potrebne varnosti ocenjujemo glede na vrsto podatkov oz. vsebin, ki se hranijo ali distribuirajo ter oceno izvorov in oblik groženj tem vsebinam.

V okviru varnosti IS govorimo o:

1. varnosti podatkov
2. varnosti dostopa do podatkov
3. varnosti podpornih informacijskih tehnologij (IT)
4. varnosti komunikacij kot posebnega dela IT

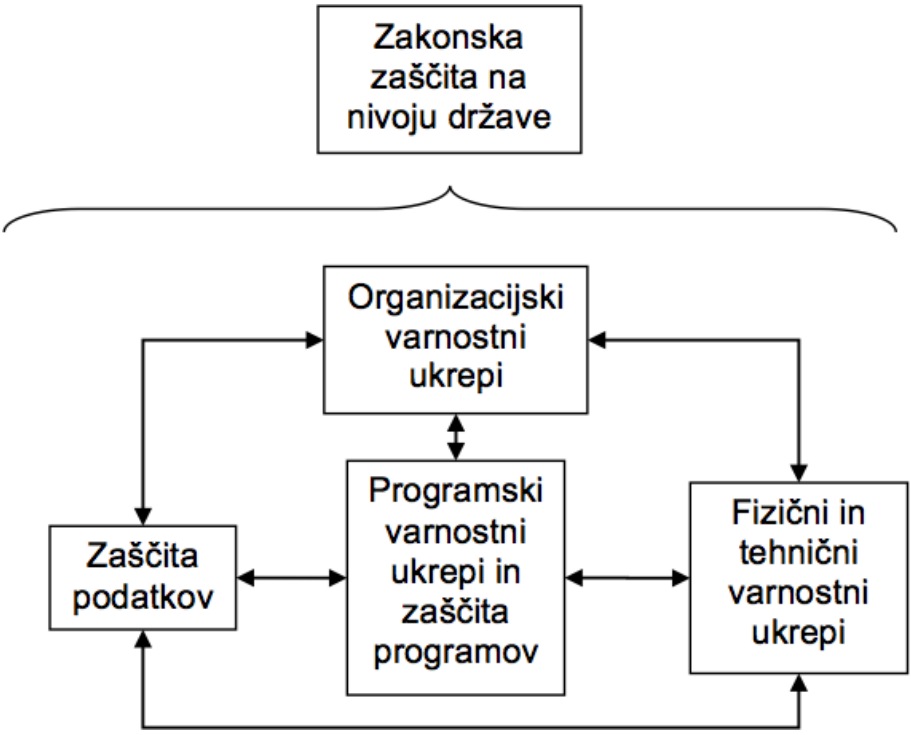
Kot je nujno dobro načrtovati IS, je nujno planirati tudi omenjene varnostne ukrepe. Le-ti cenovno ne smejo preseči vrednosti škode, ki bi lahko nastala kot posledica izgube vseh ali večjega dela podatkov. Da bi lahko planirali stopnjo varnosti, moramo poznati stopnjo tveganja. Cilj implementacije sistema varnosti IS je optimizacija njegovega delovanja glede na tveganje, ki mu je izpostavljen.

***Tveganje*** *izraža verjetnost izgube ali poškodovanja. Je verjetnost, da bodo aktivnosti imele negativne posledice.*

Tveganje ocenjujemo glede na naravo podatkov, ki jih hranimo in distribuiramo ter oceno izvora in oblike groženj njihovi varnosti:

1. način, na katerega jih država zakonsko ščiti
2. interes upravljavske strukture
3. njihova aktualnost in originalnost na lokalnem in globalnem nivoju
4. njihova pomembnost za formalno delovanje organizacije

Slika 82 shematsko prikazuje vse *nivoje zaščite IS*. Pri tem je treba upoštevati, da večje in/ali bolj frekventno tveganje predstavlja večjo stopnjo izpostavljenosti. Izvori varnostnih groženj so lahko naravni (nesreče), človeški (namerni in nenamerni), tehnični (okvare). Glede na sestavo računalniško podprtih IS (Poglavje 2.1) ugotovimo, da lahko vsaki komponenti IS pripišemo določeno stopnjo varnosti in varnostne ukrepe, ki se medsebojno dopolnjujejo.




Slika 82: *Struktura zaščitnih varnostnih ukrepov*
Vir: lasten

*Ukrepi zagotavljanja varnosti IS* zajemajo:

1. varnostno kopiranje,
2. protivirusna zaščita,
3. kriptiranje podatkov in
4. digitalno podpisovanje.

Vsak od naštetih ukrepov bo v nadaljevanju podrobneje predstavljen.

## 8.2 Zaščita podatkov pred izgubo

Sodobne organizacijske metode zaščite podatkov pred izgubo se nanašajo predvsem na varnostno kopiranje (npr. RAID polje, "backup" strežnik, prenosni mediji velikih kapacitet, ipd.). Princip je enostaven: podatki se hranijo še na drugi lokaciji/mediju in se v primeru izgube na primarni(em) lahko prenesejo nazaj in na ta način obnovijo porušeno integriteto IS.

Ločimo več vrst *varnostnega kopiranja (angl. backup copying)*:

1. *popolni (angl. full)* ustvari kopijo vseh datotek ne glede na to, ali so bile označene za arhiviranje
2. *diferencialni (angl. differential)* ustvari kopijo novih datotek in tistih, ki so bile označene za arhiviranje
3. *inkrementalni (angl. incremental)* ustvari kopijo posodobljenih datotek, ki so bile označene za arhiviranje

V praksi vedno najprej izvedemo popolno varnostno kopiranje. V nadaljevanju izvajamo diferencialno in inkrementalno varnostno kopiranje v predvidenih intervalih ob določenem času. Vzpostavimo torej urnik varnostnega kopiranja (npr. ob koncu delovnega dne, med malico, ob koncu tedna, ipd.). To so hkrati obnovitvene točke našega IS, ki predstavljajo zadnje integralno stanje IS.

## 8.3 Zaščita pred vdori in zlonamernim delovanjem

Zraven zaščite pred izgubo podatkov želimo zaščititi informacijske sisteme tudi pred nepooblaščenimi dostopi, ki imajo lahko za posledico razkritje zaupnih podatkov, škodljivo delovanje za integriteto informacijskega sistema ali celo sovražni prevzem nadzora nad informacijskim sistemom ter izsiljevanje podjetja za ponovno vzpostavitev njegovega normalnega delovanja. V ta namen na nivoju programske opreme uvajamo naslednje zaščitne ukrepe:

1. zaščita na nivoju operacijskega sistema
2. zaščita na nivoju uporabniške programske opreme
3. kriptiranje podatkov v komunikaciji
4. protivirusna orodja
5. protivohunska orodja
6. požarni zidovi

### 8.3.1 Zaščita na nivoju operacijskega sistema

Pri računalniško podprtih IS je običajno večuporabniško delo. Tu nastopata dve vrsti akterjev – administratorji in uporabniki. Vsak računalnik lahko ima več uporabnikov in administratorjev.

Administratorji so sodelavci, ki imajo v sistemu posebna pooblastila in praviloma upravljajo z dostopom do IS. V ta namen vsakemu uporabniku dodelijo identiteto – pristopno uporabniško ime in geslo. Prav tako lahko vsakemu uporabniku ali skupini uporabnikov dodelijo pravice (npr. za dostop do določenega nabora programov ali podatkov).

Višji nivo varnosti administratorji lahko dosežejo s:

1. pravilnim definiranjem uporabniških *pravic in privilegijev (angl. rights and privileges)*,
2. pravilno razporeditvijo uporabnikov v skupine,
3. konfiguracijo *uporabniške varnostne politike (angl. user security policy)* in
4. konfiguracijo *skupinske varnostne politike (angl. group security policy)*.

Vsi sodobni operacijski sistemi (Unix, Linux, Mac OS, Windows) omogočajo te ravni zaščite.

### 8.3.2 Zaščita na nivoju uporabniške programske opreme

Tako kot na ravni operacijskega sistema, lahko tudi na nivoju uporabniške programske opreme definiramo varnostno politiko. Pri tem pravice do uporabe posameznih uporabniških programov ali programskih okolij vežemo na uporabniški profil. Običajno se morajo uporabniki za uporabo takšnih programov identificirati – na primer preko vnosa uporabniškega imena in gesla. Prav tako jim na tej osnovi lahko dodelimo tri nivoje dostopa do podatkov:

1. vpogled v podatke IS
2. vnos in spreminjanje podatkov
3. brisanje podatkov

Podatki se pri brisanju z uporabniškim dostopom običajno fizično še ne brišejo, ampak premestijo v poseben del shrambe, do katerega imajo dostop le administratorji. Le-ti nato periodično (ob predhodnem soglasju skrbnikov podatkov) dokončno izbrišejo te podatke.

### 8.3.3 Kriptiranje podatkov

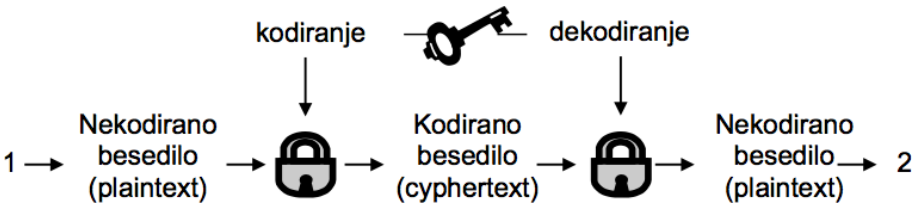
Kriptiranje nam omogoča zaščito posredovanih podatkov pred nepooblaščenim vpogledom. Običajno ga uporabljamo pri komunikaciji preko Interneta. Za razliko od Intranetov, kjer lahko dostop do podatkov učinkovito zaščitimo s prej omenjenimi mehanizmi, in Ekstranetov, kjer gre samo za dodaten nivo zaščite določenih podatkov, s poslovanjem preko Interneta nastane potreba po prenosu podatkov izven meja organizacije in s tem potreba po ustrezni zaščiti prenesene vsebine.

Kaj se lahko zgodi s podatki v komunikacijskem kanalu?

1. nekdo lahko prisluškuje
2. nekdo lahko komunikacijo prekine
3. nekdo lahko prestreže podatkovne pakete in jim spremeni vsebino
4. nekdo lahko generira neobstoječo/napačno vsebino in jo posreduje naprej kot naše sporočilo

Seveda gre v vseh naštetih primerih za neželeno dogajanje – zato ga želimo preprečiti. Čeprav prekinitve komunikacije ne moremo preprečiti, če je le-ta prekinjena zaradi odpovedi strojne in/ali programske opreme, lahko s šifriranjem (kriptiranjem) podatkov preprečimo vse ostalo. Predvsem tukaj mislimo na *prisluškovanje (angl. eavesdropping)* ter *prestrezanje in spreminjanje podatkov med prenosom (angl. man in the middle attack)*. Kriptografska zaščita podatkov zagotavlja dve osnovni funkciji:

1. integriteta prenesenih podatkov (na tehničnem nivoju ga zagotavlja TCP/IP protokol)
2. onemogočenje neavtoriziranega vpogleda ali spreminjanja podatkov med prenosom




Slika 83: *Simetrično kriptiranje*
Vir: lasten

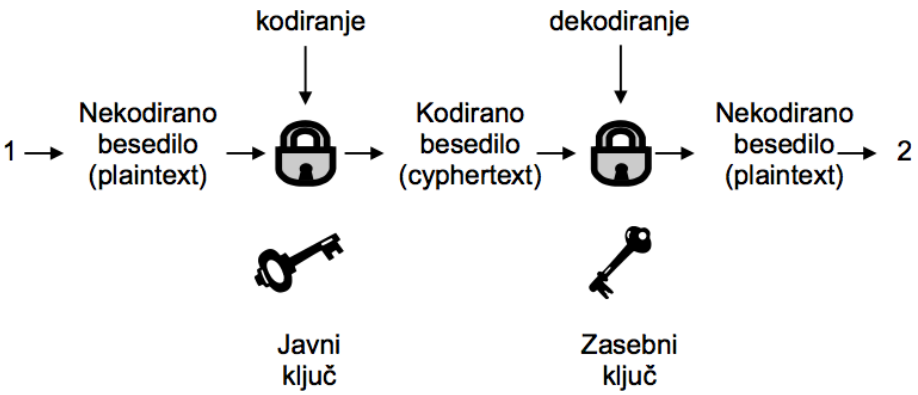
Predvsem drugo funkcijo lahko zagotovimo s kriptiranjem podatkov, ki se na ta način prenesejo kot nepoklicanim nerazumljiva sporočila, ki pa jih prejemnik po dekriptiranju lahko razbere v izvirni obliki. Kaj je torej kriptiranje (šifriranje) in kaj dekriptiranje (dešifriranje)?

***Kriptiranje*** *je postopek pretvorbe razumljivega sporočila v nerazumljivo.* ***Dekriptiranje*** *je postopek pretvorbe kriptiranega sporočila v razumljivo.*

Glede na način izvedbe postopka ločimo:

1. Simetrično kriptiranje – za oba postopka uporabljamo isti ključ (npr. Slika 83), ki ga morata uskladiti uporabnika pred izmenjavo podatkov
2. Asimetrično kriptiranje – za kriptiranje uporabljamo javni ključ, za dekriptiranje pa zasebni; princip zaščite je zasnovan na *infrastrukturi javnih ključev (angl. Public Key Infrastructure, PKI)* (npr. Slika 84)

Prednost simetričnega kriptiranja je njegova enostavnost, saj zadošča da uporabnika ključ izmenjata samo enkrat, da bi lahko v nadaljevanju varno komunicirala. Težava nastopi, če nekdo s prisluškovanjem prestreže ta ključ. Seveda je po tem ves trud s kriptiranjem zaman, saj je komunikacija med uporabnikoma s tem postala *ranljiva (angl. vulnerable)*.




Slika 84: *Asimetrično kriptiranje*
Vir: lasten

Logika asimetričnega kriptiranja je zasnovana na principu, da vsak, ki pozna uporabnikov javni ključ, lahko pošlje temu uporabniku kriptirano sporočilo, ki ga bo lahko pravilno dekriptiral samo on s svojim zasebnim ključem. Partnerja v komunikaciji lahko medsebojno izmenjata javne ključe in na ta način bosta z uporabo ustreznih ključev lahko varno obojestransko komunicirala. Ker je takšnih uporabnikov več, pa tudi zaradi možnega lažnega predstavljanja, so bili uvedeni zaupanja vredni strežniki (angl. Certificate Authority, CA), ki hranijo javne ključe partnerjev v komunikaciji in jih po varni povezavi posredujejo uporabnikom. Ti strežniki tvorijo prej omenjeno infrastrukturo javnih ključev (PKI) in hranijo ne samo javne ključe ampak tudi digitalne certifikate njihovih uporabnikov.

**Primer**: PGP šifriranje in podpisovanje

Pri PGP (”Pretty-Good-Privacy”) *šifrirana* sporočila so sestavljena po naslednjem postopku:

1. PGP najprej generira naključno število – t.i. *ključ seje (angl. session key)*,
2. nato sestavi sporočilo iz:

(a) sporočila zakodiranega z začasnim ključem in

(b) s prejemnikovim javnim ključem zakodiranega začasnega ključa.

Ta tehnika je pogosta pri varnem dostopu do spletnih strani. Tu moramo pogosto vpisati neko varnostno geslo ali na določen način premakniti miško, da generiramo omenjeni začasni ključ, ki se v času naše seje uporablja za šifriranje naših odhodnih sporočil spletišču.

PGP *dešifriranje* sporočil poteka po obratni poti:

1. prejemnik s svojim privatnim ključem dekodira začasni ključ in
2. uporabi le-tega za dekodiranje sporočila.

PGP omogoča tudi *podpisovanje* sporočil:

1. sporočilo podpišemo z lastnim privatnim ključem in oboje kodirano s prejemnikovim javnim ključem pošljemo prejemniku
2. ta po dekodiranju s svojim privatnim ključem uporabi naš javni ključ za kodiranje prejetega sporočila in s primerjavo s podpisom ugotovi, ali je sporočilo avtentično

Včasih nas bolj kot zaupnost vsebine zanimata *integriteta in avtentičnost* nekega sporočila. Enostavno bi bilo na primer prilepiti naš elektronski podpis tudi v katero drugo sporočilo in ne le na tisto, ki smo ga sami poslali. To lažje, kot s šifriranjem celotnega sporočila, dosežemo s t.i. *zgoščevalno funkcijo (angl. hash)* nad osnovnim besedilom, ki nam da njegov *povzetek (angl. digest)* – niz znakov, ki ga zakodiramo s privatnim ključem in predstavlja podpis za prejeti dokument. Prejemnik lahko postopek ponovi z našim javnim ključem in s primerjavo podpisov preveri ne samo avtentičnost sporočila, ampak tudi njegovo integriteto, saj bi vsaka sprememba sporočila imela za posledico

tudi drugačen povzetek. Če želimo zagotoviti še zaupnost vsebine sporočila, sporočilo in povzetek zakodiramo s prejemnikovim javnim ključem in na ta način zagotovimo, da ju bo lahko samo on pravilno dekriptiral.

**Primer**: Uporaba kriptiranja pri e-poslovanju

Po navedbah (Kričej, 2002) je vsem projektom, ki se izvajajo v okviru programa e-uprava poleg strateške usmeritve skupna tudi skrb za popolno varnost poslovanja. Zato je bil v sklopu Centra vlade za informatiko (CVI RS) vzpostavljen overovitelj digitalnih potrdil (SI*CA). Le-ta overja kvalificirana digitalna potrdila, za katera veljata najvišja stopnja varovanja in načelo tajnega kodiranja ter deluje v skladu z Zakonom o elektronskem poslovanju in e-podpisu (ZEPEP). V skladu z ZEPEP elektronskemu podpisu priznavamo enako veljavo kot lastnoročnemu podpisu. Politika delovanja overovitelja določa namen, delovanje in metodologijo upravljanja z digitalnimi potrdili, odgovornost overovitelja ter varnostne zahteve, ki jih mora izpolnjevati imetnik.

Na CVI RS predstavljata overovitelja dva izdajatelja kvalificiranih digitalnih potrdil (certifikatov):

1. SIGEN-CA za državljane in pravne osebe in
2. SIGOV-CA za upravo RS oz. zaposlene v javni upravi RS.

Oba sta mednarodno registrirana, medsebojno priznana ter tehnološko in zakonsko enakopravna. Digitalna potrdila oz. certifikate na splošno izdaja t.i. *Certification Authority (CA)*, ki vzdržuje t.i. *certificate server (cert)*. Le-ta običajno vzdržuje tudi podatkovno bazo javnih ključev - t.i. *PKI (Public Key Infrastructure)*, ki omogoča CA, da razen funkcij *cert* omogoča tudi verifikacijo certifikatov.

Z digitalnimi potrdili uporabniki šifrirajo podatke in dokumente digitalno podpisujejo. Šifriranje podatkov zagotavlja zaupnost in kontrolo dostopa do podatkov, digitalni podpis pa nedvoumnost identitete, nezatajljivost lastništva in

neokrnjenost podatkov v elektronski obliki. Naše digitalno potrdilo oz. certifikat je sestavljeno iz naše identifikacije, osebnih podatkov in podatkov o izdajatelju ter našega javnega in zasebnega ključa.

Pot do digitalnega potrdila za občane vodi preko upravne enote, kjer se moramo identificirati z osebnim dokumentom in vlogo lastnoročno podpisati. Čez nekaj dni po pošti prejmemo obvestilo z delom gesla, drugi del pa prejmemo na e-naslov, ki smo ga navedli v vlogi – na ta način je poskrbljeno za potrebno varnost. Nato lahko prevzamemo digitalno potrdilo v osebnem računalniku, ga shranimo na pametno kartico, (prenosni) disk ali drug primeren medij. *Pametna kartica* (angl. *Smart-Card*) je primernejša, ker nas kot osebna izkaznica lahko spremlja tudi na poti in se z njo lahko elektronsko identificiramo tudi na javno dostopnih informacijskih točkah (t.i. teletočke).

Z digitalnim potrdilom se lahko prijavimo v sistem e-uprave, ki nas na podlagi le-tega "prepozna". Na ta način lahko dostopamo do najobčutljivejših zbirk podatkov (javne evidence, CRP, ...). Seveda imamo dostop le to tistih podatkov, do katerih smo upravičeni, vendar na podlagi našega "podpisa" ni formalnih ovir za v pogled v vse lastne in registriranim uporabnikom dostopne podatke iz omenjenih registrov. Prav tako lahko po tej poti osvežujemo lastne podatke, oddajamo razne vloge, ipd. – ko z vnosom podatkov zaključimo dokument digitalno podpišemo in ga oddamo preprosto s "klikom na gumb". Naš digitalni certifikat nas enolično identificira kot uporabnika teh storitev in omogoča varen prenos podatkov in dokumentov med nami in spletnimi strežniki, ki jih hranijo. V perspektivi digitalizacije je predvidena nova osebna izkaznica, ki bo kot pametna kartica razen osebnih podatkov vsebovala še digitalni certifikat. S tem bo "digitalni državljan" pridobil avtoriziran dostop do vseh storitev e-uprave, e-zdravstva, itd.

### 8.3.4 Zaščita proti škodljivim programom

Razen zaščite podatkov je potrebna tudi zaščita proti škodljivim programom:

1. virusom
2. črvom (angl. worm)
3. trojanskim konjem
4. vohunskim (angl. spyware) programom ipd.

***Virusi*** *so destruktivni računalniški programi, ki imajo za cilj poškodovanje ali uničenje podatkov/funkcionalnosti programov na okuženem računalniku.*

Vsak računalniški *virus* ima naslednje komponente:

1. infekcija – programski del, ki omogoča širjenje virusa
2. nosilna komponenta (angl. payload) – predstavlja glavno aktivnost virusa (brisanje podatkov ali onemogočenje dostopa do programov)
3. prožilna funkcija (angl. trigger) – definira čas ali dogodek, ko se začne izvajati nosilna komponenta programa

Za obrambo pred okužbo z virusi in odstranjevanje le-teh iz sistema so bili razviti t.i. protivirusni programi. Njihov osnovni namen je preprečiti kopiranje virusa na računalnik. Če je do tega že prišlo, želimo preprečiti njihovo delovanje – izpolnitev pogoja prožilne funkcije. Viruse nevede najpogosteje aktiviramo sami z zagonom okužene datoteke, v prepričanju, da odpiramo kak dokument ali zaganjamo aplikacijski program, ki smo ga prejeli od ”zaupanja vrednega vira”.

Varnostne ukrepe, ki jih izvajajo protivirusni programi lahko delimo na:

1. preventivne (preprečitev okužbe – kopiranja virusne datoteke na računalnik s pravočasnim obveščanjem uporabnika) in
2. sanacijske ukrepe (izolacija virusa, da se ne bi izvedel, ter njegova neškodljiva odstranitev).

Največjo škodo lahko povzročijo t.i. zagonski (angl. boot) virusi – ti virusi inficirajo zagonski (*angl. boot sektor*) na disku računalnika in na ta način lahko preprečijo nalaganje operacijskega sistema. To ima seveda za posledico neuporabnost računalnika. Ker se zaženejo prvi, lahko tudi imobilizirajo morebitne protivirusne programe, da jih ne morejo pravočasno zaznati. Zato lahko preteče tudi dalj časa, ne da bi zaznali okužbo.

Pri načrtovanju ukrepov za zaščito pred virusi imamo več možnosti:

1. Organizacijski: onemogočanje instalacije neavtoriziranih programov in kopiranja vsebin z nepreverjenih lokacij in medijev; za preverjanje takšnih vsebin je priporočeno *zaganjanje programov v peskovniku (angl. sandboxing)* – uporaba v mrežo nepovezanega računalnika za testiranje sumljivih programov in vsebin
2. Nadzorni: uporaba protivirusnih orodij
   (a) Funkcija “on access scanning”
   (b) Avtomatsko ažuriranje baze (digitalnih sledi) virusov
   (c) Skeniranje vseh sumljivih datotek pred njihovim odpiranjem/kopiranjem
   (d) Protivirusna zaščita e-poštnega nabiralnika
   (e) Pogosta pomanjkljivost – baza virusov ni ažurna (manjka virusov ”digitalni podpis”; vsaka virusna datoteka ima dva dela – telo in podpis – naključno generiran ključ za kodiranje, s katerim je kodirano telo)
3. Sanacijski: računalnik je okužen, kar običajno zaznamo po upočasnjenem delovanju, manjkajočih podatkih, reducirana je funkcionalnosti določenih programov; temu običajno sledijo ukrepi v naslednjem vrstnem redu:
   (a) Odklop računalnika z omrežja
   (b) Zbiranje sumljivih medijev

(c) Odstranitev virusa (s programom ali ročno)

(d) Ponovna instalacija celotnega sistema (skrajni, najdražji a najzanesljivejši ukrep – pri tem potrebujemo varnostne kopije podatkov in programov iz obdobja pred okužbo)

Večina sodobnih protivirusnih programov nas učinkovito ščiti pred "črvi in trojanskimi konji". Le-ti predstavljajo največjo grožnjo omreženim računalnikom.

***Črv*** *je program, ki se širi avtomatsko preko omrežja in pri tem izkorišča šibke točke omreženega računalnika ter z njim povezanih računalnikov.*

Širjenje poteka s pošiljanjem paketov podatkov, običajno preko UDP protokola, saj črv ne pozna omrežja. Ko prejme odziv in se poveže z gostiteljskim računalnikom, se lahko prenese naprej in na ta način širi okužbo.

V osnovi črvi nimajo lastnosti virusov – se ne vežejo na določene datoteke/programe. Njihovega širjenja običajno ne zaznamo, dokler ne opazimo neželenega ali upočasnjenega delovanja omrežja. Razen klasičnih mrežnih črvov obstajajo še e-poštni črvi. Večkrat jih sploh ni potrebno pognati, saj se zaženejo sami E-poštni črvi se na primer sami prenesejo vsem naslovnikom iz adresarja okuženega računalnika (npr. VBS/OnTheFly@mm, kjer mm = "mass mailer" po klasifikaciji Wildlist baze virusnih vzorcev). Pojavljajo se tudi hibridni črvi, ki imajo hkrati še lastnosti virusov ali trojanskih konjev.

***Trojanski konji*** *so škodljivi programi, ki nimajo zmožnosti samostojnega razširjanja. Opravljajo drugo funkcijo od deklarirane (npr. lažni prijavni program posname geslo, odgovori na UDP sporočilo črva in omogoči njegovo širjenje).*

Uporabljajo jih "hekerji" za pridobivanje dostopa do tujih računalnikov. Trojanske konje najpogosteje distribuirajo piratski programi, prenašajo pa se tudi preko P2P (Peer-To-Peer) orodij (KaZaA, e–Mule, LimeWire ipd.).

**Primer**: naložimo predvajalnik glasbe, ki v ozadju odpre *omrežna zadnja vrata (angl. backdoor)* preko katerih lahko nekdo prevzame nadzor nad računalnikom.

Trojanski konji, ki izkoriščajo dostop na "zadnja vrata" so t.i.:

1. *RAT (angl. Remote Access Tools)* – orodja za popoln nadzor računalnika z oddaljene lokacije
2. *Zadnja vrata (angl. Backdoor)* – trojanski konji, ki omogočajo ustvarjanje nezaščitenih dostopov do računalnika z administratorskimi pooblastili s pomočjo RAT orodij

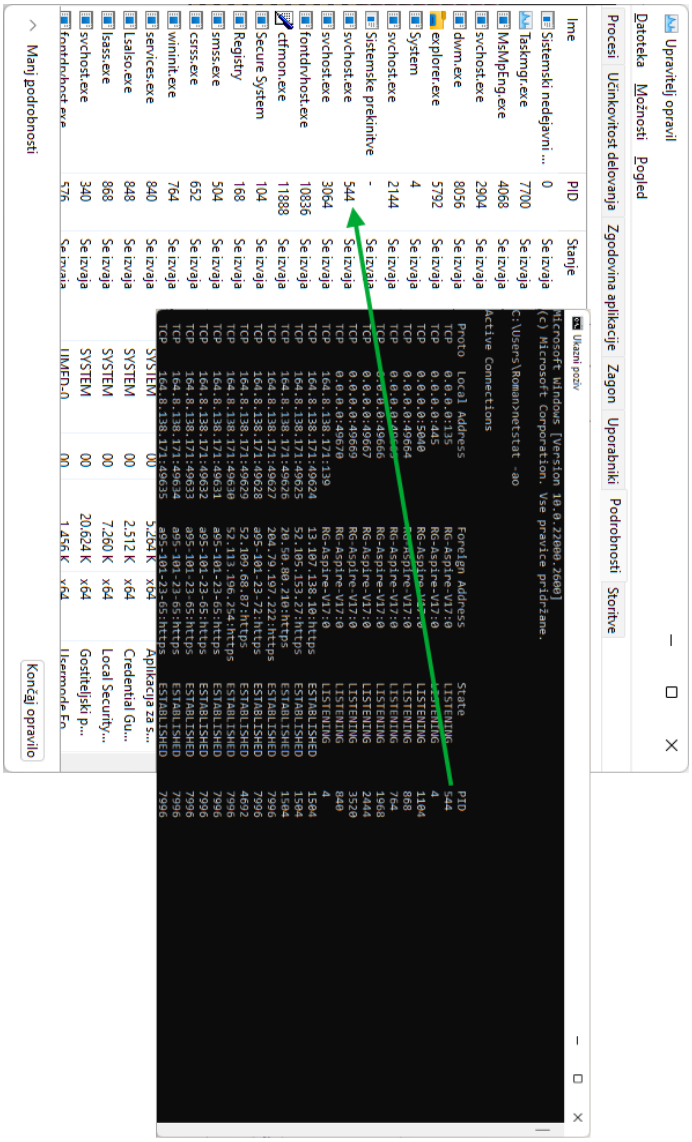
**Primer**: kako opazimo trojanskega konja (Slika 85)?

Če želite na računalniku z Windows operacijskim sistemom ugotoviti, kateri programi (procesi) so odprli TCP/UDP vrata in prisluškujejo na njih, lahko to lahko storite v konzoli z ukazom:

>**netstat –ao**

Da bi ugotovili, za katere programe gre, morate omogočiti prikaz PID (Process ID) v upravitelju procesov (Windows Task Managerju). Če na seznamu opazimo program, ki ni niti del operacijskega sistema, niti aplikacijski program, ki ga poznamo, gre najverjetneje za trojanskega konja.

Posebno nevarni so t.i. *izsiljevalski (angl. ransomware)* programi, ki v IS vstopijo kot črvi ali trojanski konji, nato pa prevzamejo administratorski nadzor nad IS, ki omogoča oddaljenemu napadalcu ekskluziven in popoln nadzor nad delovanjem IS. Običajno se takšni napadi manifestirjo tako, da niti administratorji niti uporabniki IS več nimajo dostopa do njegovih podatkov in funkcij in le-teh tudi ne morejo nadomestiti iz varnostnih kopij, saj napadalci običajno kriptirajo celotno programsko opremo IS, tako da jo lahko samo oni ponovno dekriptirajo.



Slika 85: *Preko omrežnih vrat in PID do trojanskih konjev*
Vir: lasten, zajem zaslona, 2023

Proti trojanskim konjem se lahko zaščitimo s pomočjo požarnih zidov (angl. firewall). Ob poskusu odpiranja vrat, nas program, ki implementira požarni zid vpraša, ali bomo to dovolili. Če smo dovolj pazljivi, bomo pri tem opazili trojanskega konja, še preden postane nevaren. Požarni zid preprečuje neavtoriziranim uporabnikom dostop do lokalnega omrežja in poskrbi, da se z okolico izmenjujejo le kontrolirane vsebine.

Razen omenjenih najnevarnejših programskih groženj je bilo razvitih še več vrst *škodljivih programov* (angl. *Malware – Malicious Software*):

1. *Adware* programi – omogočajo oglaševanje med deskanjem po spletu preko pojavnih oken (običajno se instalirajo s programi, za katere smatrate, da so pod "Open Source" licenco)
2. *Spyware* programi – vključujejo naslednje komponente:
   (a) Keylogger – program, ki snema tipkanje po računalniški tipkovnici
   (b) Password capture – program, ki beleži vaša gesla
3. *Spamware* – program, ki uporablja računalnik kot odskočno desko za "spam" (neželena reklamna sporočila):
   (a) pogosto služijo tudi za ugotavljanje naših interesov na spletu
   (b) instalirajo se s pomočjo trikov – pod navedbo brezplačnih programov, ki vam bodo koristili (npr. prav "osvobodili vaš računalnik spyware-a")

*Adware* programi so manj nevarni od *Spyware* programov, oboji pa nekoristno trošijo računalniške vire. *Spam* predstavljajo predvsem neželena e-poštna sporočila, ki oglašujejo določene izdelke ali usluge. Pri tem ne gre samo za nadležna sporočila, ampak za resno grožnjo podjetjem zaradi nekoristne izrabe sredstev, izgube časa in s tem tudi denarja. Če vračunamo še stroške njihovega preprečevanja, dodatnih virov za hrambo e-poštnih sporočil, dodatne uporabniške podpore, ipd. ti stroški niso zanemarljivi (npr. po aktualni raziskavi

leta 2020 s tem povezane direktne stroške v ZDA ocenjujejo na cca. 20 mrd. USD letno). Sodobne AntiSpam rešitve nudijo t.i. ISP (Intelligent antiSPam) požarni zidovi.

*Phishing* predstavlja zbiranje občutljivih osebnih informacij (uporabniških imen, gesel in podatkov o kreditnih karticah) z lažnim predstavljanjem kot zaupanja vredna entiteta na spletu ali v elektronski komunikaciji (npr. spletišča in e-pošta renomiranih firm, bank, zavarovalnic, ipd.). Gre tudi za vrsto Spyware programov, ki z lažnim predstavljanjem skušajo pridobiti zaupne podatke od uporabnikov z namenom izsiljevanja ali finančne koristi.

**Primer**: Phishing in Spoofing

Znan je "phishing napad" na eBay – uporabniki so dobili e-pošto, kot uradno sporočilo eBay uporabniškega servisa, v katerem je bilo zapisano, da avkcija za predmet, ki jih zanima, zaradi tehničnih razlogov (odpravljanja težav z regularnimi strežniki) poteka preko "nadomestnih spletnih strani" in pri tem navedli ustrezen *link* na lastno spletno stran. Veliko uporabnikov je temu nasedlo in opravilo nakupe preko teh strani ter pri tem ostalo brez nakupa, pa tudi brez nakazanega denarja.

Razen omenjenega primera se je pojavilo tudi razpošiljanje lažnih DNS naslovov, kjer so bili napadeni določeni DNS strežniki, ki služijo kot imenik med uporabniki in spletišči. Napadalci so pri tem pravi IP naslov (npr. za eBay) zamenjali z drugim, kar bi uporabnike tega DNS strežnika avtomatsko preusmerilo na lažno spletno stran. Takšnemu napadu pravimo "DNS Spoofing" in so nevarnejši od "Phishinga", saj uporabnik ne more zaznati napada, dokler ni prepozno. Vsi sodobni spletni brskalniki imajo sedaj že t.i. "Phishing filtre", ki odkrivajo takšne anomalije z navzkrižnim primerjanjem rezultatov več DNS strežnikov, vendar tudi ti popolne zaščite, ker gre za osnovno funkcionalnost DNS strežnikov, ne morejo nuditi.

Če opazimo sumljiv izvedljivi progam, se za informacijo o njegovem izvoru lahko obrnemo na Internet. Za vsak znan izvedljivi program tu lahko dobimo informacijo o njegovem proizvajalcu in funkcionalnosti, pa tudi, ali obstaja možnost, da gre za škodljiv program. Na osnovi pridobljene informacije se nato lažje odločimo kako postopati naprej.

## Literatura

Flood, R. L. (1987). Complexity: A definition by construction of a conceptual framework. *Systems Research*, 4(3), 177–185.

Grant, D., Lambert, D., Stock, J., & Ellram, L. (2006). *Fundementals of Logistics Management*. McGraw-Hill, Berkshire, UK, european edition.

Institute, P. M. (2003). *A Guide To The Project Management Body Of Knowledge*. Project Management Institute, 3rd edition.

Keene, S. (1994). Comparing hardware and software reliability. *Reliability Review*, 14(4), 5–7, 21.

Kerzner, H. (2003). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. Wiley, 8th edition.

Kričej, D. (2002). *e-uprava na dlani, poslovanje z državo po internetu danes in jutri*. Pasadena.

Maslow, A. (1943). A theory of human motivation. *Psychological Review*, 50(4), 370–96.

Nandrajog, I. (2001). *CIS 679 Management of IS: Simplified Object Access Protocol*. Technical report, NIJT.

Rainer, R., Turban, E., & Potter, R. (2006). *Introduction to information systems*. Wiley.

Shannon, C. & Weaver, W. (1963). *A Mathematical Theory of Communication*. University of Illinois Press, Champaign, IL, USA.

Šuhel, P. & Murovec, B. (2003). *Računalniška integracija proizvodnje*. Gorenje d.d., Izobraževalni center Ljubljana, Velenje, Maribor.

# Slike

# Tabele

# RAČUNALNIŠTVO IN INFORMATIKA V LOGISTIKI

ROMAN GUMZEJ

Univerza v Mariboru, Fakulteta za logistiko, Celje, Slovenija
roman.gumzej@um.si

Učbenik je namenjen predvsem študentom prvih letnikov visokošolskega strokovnega in univerzitetnega študijskega programa Fakultete za logistiko. Je osnovna literatura učnih predmetov Osnove računalništva v logistiki na visokošolskem strokovnem in Računalništvo v logistiki na univerzitetnem študijskem programu. V okviru teh predmetov zasnujemo logistične informacijske sisteme. Predstavimo njihov pomen, vlogo in naloge ter komponente: računalniška podatkovna, strojna, mrežna, programska in organizacijska oprema. Pri praktičnem delu se posvetimo predvsem računalniško podprtemu inženiringu logističnih informacijskih sistemov na univerzitetnem ter poslovni analitiki in podpori odločanju na visokošolskem strokovnem programu. Vsebino sklenemo z metodami in mehanizmi za zagotavljanje varnosti računalniško podprtih logističnih informacijskih sistemov. Učbenik predstavlja osnovo za naknadno obravnavo specializiranih vsebin pri predmetih Poslovni informacijski sistemi v logistiki, Informacijska podpora logističnim sistemom in procesom ter Integracije logističnih informacijskih sistemov.

**DOI**
https://doi.org/10.18690/um.fl.1.2024

**ISBN**
978-961-286-816-1

**Ključne besede:**
logistični informacijski sistemi,
računalniške platforme,
komunikacijske mreže,
logistični podatki,
poslovna inteligenca,
informacijska varnost

Univerzitetna založba Univerze v Mariboru

**DOI**
https://doi.org/
10.18690/um.fl.1.2024

**ISBN**
978-961-286-816-1

**Keywords:**
logistics information systems,
computing platforms,
communication networks,
logistics data,
business intelligence,
information security

# COMPUTING AND INFORMATICS IN LOGISTICS

ROMAN GUMZEJ
University of Maribor, Faculty of Logistics, Celje, Slovenia
roman.gumzej@um.si

This textbook is predominantly meant for first year students at the Faculty of Logistics. It represents the primary source of information for the subjects Fundamentals of Computer Science at the bachelors and Computer Science and Informatics in Logistics at the university study programme. By these courses the fundamentals of computer-based logistics information systems are set. Here, their purpose, role and tasks as well as their components are presented, namely their dataware, hardware, netware, software and orgware. Within their tutorials the focus is on computer-aided engineering of logistics information systems at the university and business analytics and decision support systems at the bachelors study programme, respectively. Their contents are rounded up with the presentation of the methods and mechanisms of safety and security in computer based logistics information systems. This textbook lays the foundations for subsequent specialised courses on Business information systems, Information Support in Logistics Information Systems and Processes as well as Logistics Information Systems Integrations.

University of Maribor Press

Učbenik Računalništvo in informatika v logistiki avtorja dr. Romana Gumzeja je prvenstveno namenjen študentom prvega letnika visokošolskega strokovnega in univerzitetnega študijskega programa Fakultete za Logistiko Univerze v Mariboru. Učbenik, ki je osnovna literatura študijskih predmetov Osnove računalništva v logistiki in Računalništvo v logistiki, v osmih poglavjih podaja osnovno znanje s področja informacijske podpore logističnim sistemom in procesom, ki jih bodo študentje s pridom uporabili in nadgradili pri ostalih predmetih izbranega programa. Učbenik zajema vsebino naslednjih področij: utemeljitev računalništva in informatike, informacijski sistemi v logistiki, računalniška strojna oprema, računalniška komunikacijska oprema, računalniška programska oprema, računalniška podatkovna oprema, računalniška organizacijska oprema in varnost informacijskih sistemov.

Prof. dr. **Ajda FOŠNER**
Univerza na Primorskem

Učbenik bo z aktualnimi vsebinami obogatil zakladnico znanj na področju računalništva ter informatike in je zato širše študijsko uporaben. Hkrati pa ga priporočam tudi vsem, ki se ukvarjajo z logistično dejavnostjo in bi želeli nadgraditi svoja znanja na področju računalništva in informatike v logistiki.

Prof. dr. **Bojan ROSI**
Univerza v Mariboru

Skladno s to usmeritvijo je nastalo tudi to delo, ki ga ob tej priliki priporočam zlasti študentom logistike, pa tudi raziskovalcem in inženirjem ter vsem drugim, ki bi se želeli bolje seznaniti z zanimivim področjem informatike v logistiki.

Prof. dr. **Maja FOŠNER**
Univerza v Mariboru

Menim, da bo učbenik Računalništvo in informatika v logistiki študentom Fakultete za logistiko v veliko pomoč pri študiju predmetov Osnove računalništva v logistiki na visokošolskem strokovnem programu in Računalništvo v logistiki na univerzitetnem študijskem programu ter tudi vsem ostalim bralcem, ki jih ta tematika zanima.

Izr. prof. dr. **Simona STERNAD ZABUKOVŠEK**
Univerza v Mariboru

Univerza v Mariboru
Fakulteta za logistiko