# SRSV – 1. Uvod

Osnovni pojmovi

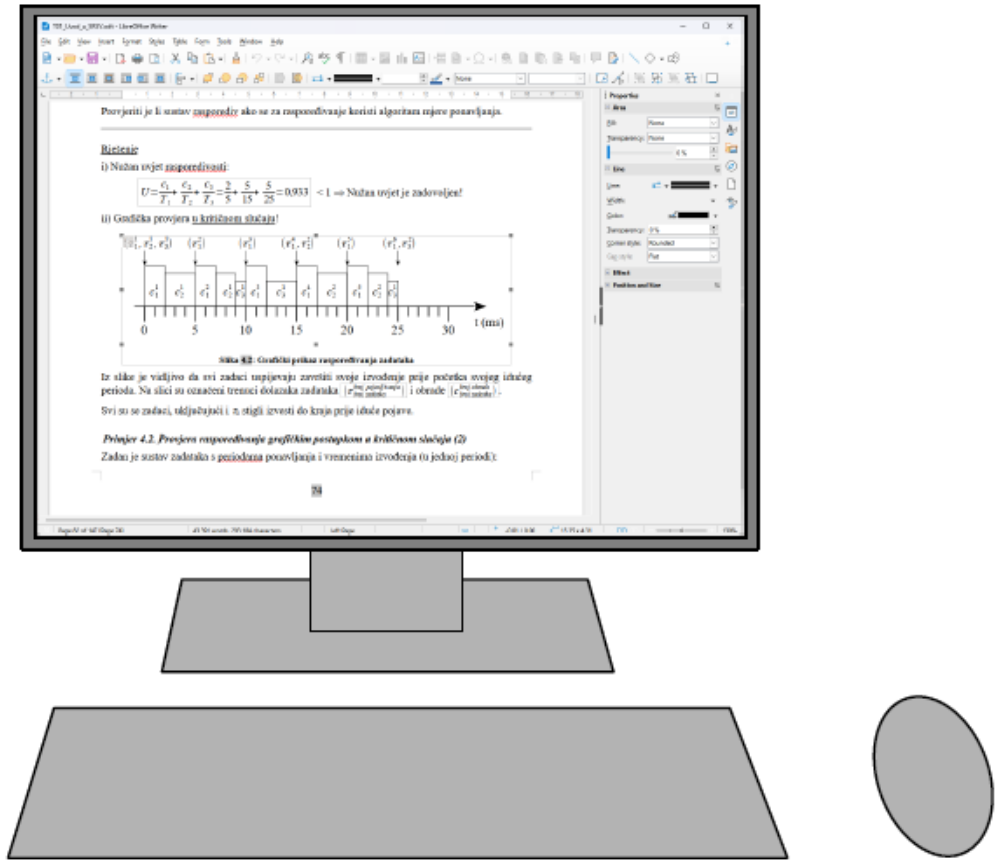
(malo teorije, nekoliko novih pojmova)

# 1. Uvod

- Što su to *sustavi za rad u stvarnom vremenu*?
- Svi sustavi rade u stvarnom vremenu, vrijeme uvijek teče!
- U kontekstu računarstva SRSV ima posebno značenje, predstavlja samo neke sustave.
  - glavna je razlika u posljedicama neispravnog rada
    - kod SRSV-a posljedice mogu biti značajne, ozljede ljudi, velika materijalna šteta i sl.
- Kratki osvrt na različite kategorije sustava (u ovom kontekstu) slijedi

# 1.1.1. Uredsko korištenje računala

- uredske aplikacije (uređivanje teksta, e-pošta, web, ...)
- očekujemo „ugodan rad”
  - čovjek je „spor” pa je ovo relativno jednostavno ostvariti
  - reakcija do ~100 ms na radnje korisnika je ok
- što ako nešto ne radi, čekamo duže ili aplikacija se sruši?
  - korisnik će biti „manje zadovoljan”
  - ali nitko neće stradati
  - materijalna šteta je „samo” u izgubljenom vremenu korisnika



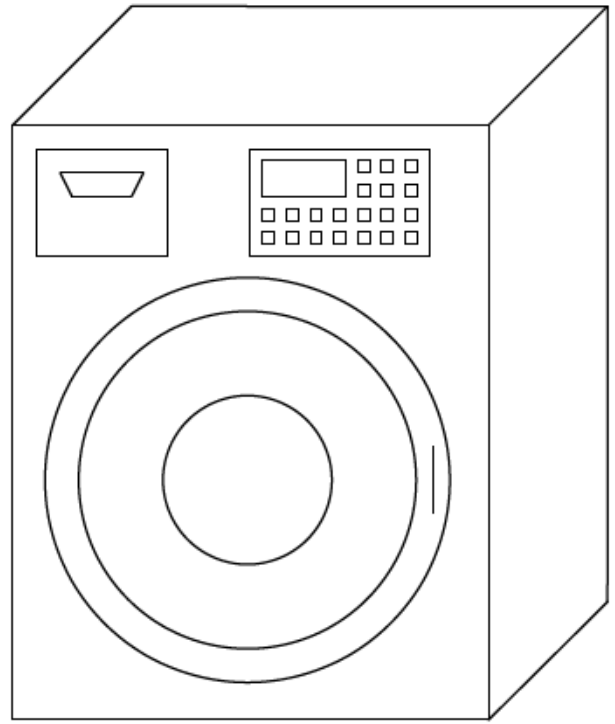
Slika 1.1. Uređivanje dokumenta

# 1.1.2. Reprodukcija audiosadržaja/videosadržaja

- reprodukcija mora biti „glatka”
- svaka nepravilnost će se primijetiti
  - slika „trza”
  - zvuk prekida, krči
- ipak „nitko neće stradati”
  - osim ako se to koristi za upravljanje (onda to je SRSV)

# 1.1.3. Računalom upravljani kućanski uređaji

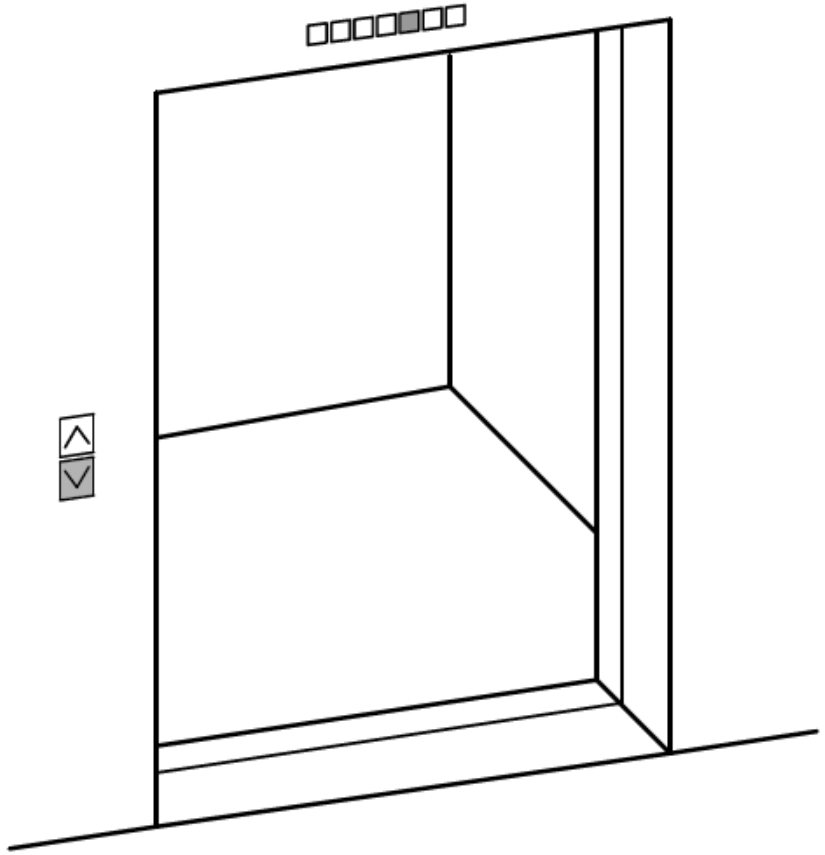
- npr. perilice, pećnice, hladnjaci, klimatizacijski uređaji
- njima upravljaju jednostavni mikrokontroleri (male snage)
- ali krivo upravljanje može izazvati štetu
  - požar, poplava
  - moguća veća materijalna šteta i/ili ozljede ljudi
- stoga njih svrstavamo u SRSV



Slika 1.3. Perilica rublja

# 1.1.4. Dizalo u zgradi

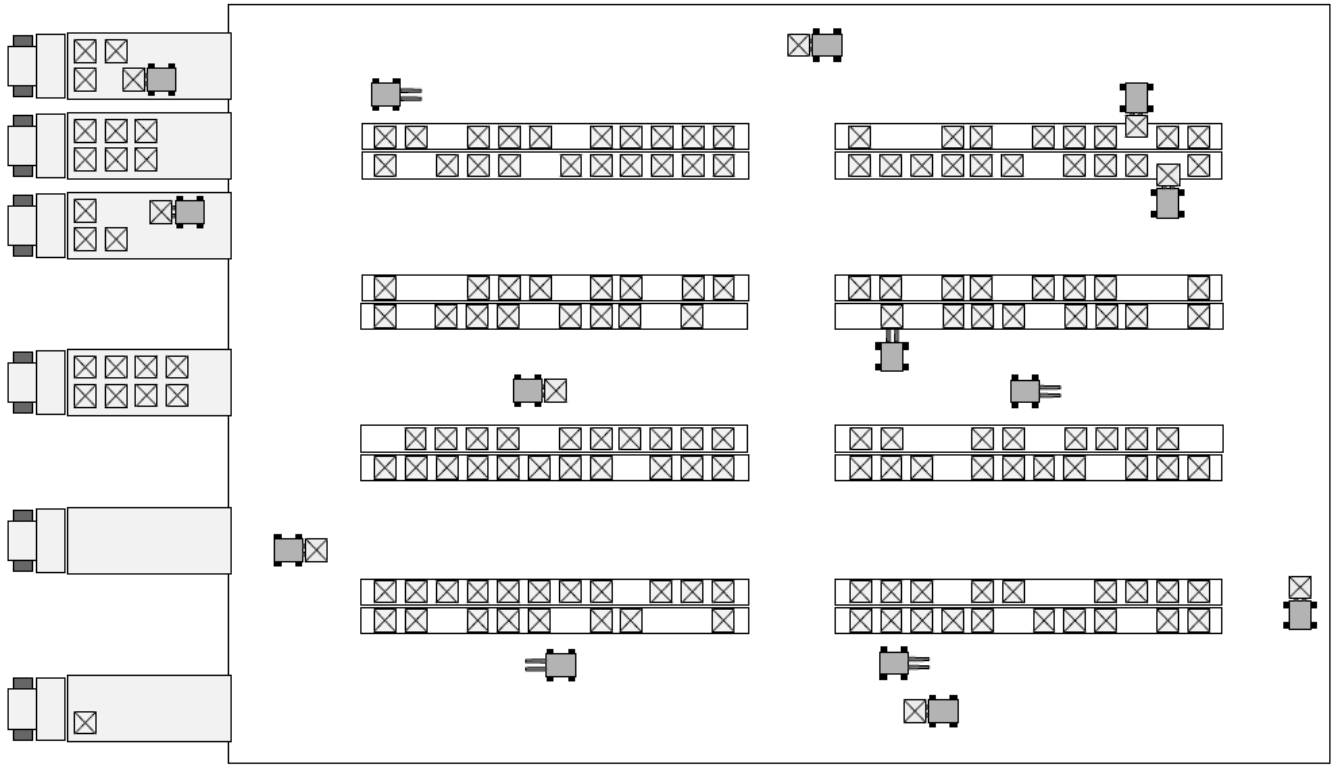
- očito da mora raditi ispravno jer bi posljedice neispravnog rada mogle biti kobne
  - ne zatvarati vrata dok se ulazi/izlazi
  - ne kretati prije zatvaranja vrata
  - prilagoditi brzinu kretanja
  - ...
- samo upravljanje se može obaviti jednostavnim mikrokontrolerima jer je logika jednostavna



Slika 1.4. Dizalo

# 1.1.5. Automatizirano skladište

- složeni sustav
- složeno i raspodijeljeno upravljanje
- loše upravljanje može imati materijalne posljedice
  - oštećeni proizvodi
  - kašnjenje u transportu
  - ...
- i ovo spada u SRSV



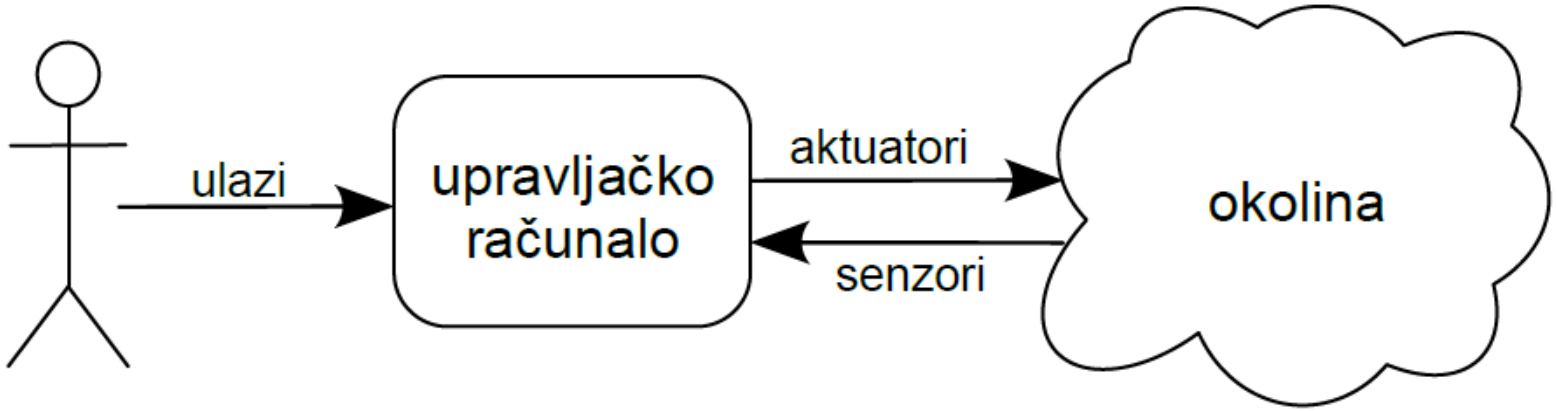
Slika 1.5. Automatizirano skladište

# 1.1.6. Usporedba

- svi rade u „stvarnom vremenu”, ali za neke je ono bitnije obzirom na posljedice neispravnog rada
- „rad u stvarnom vremenu” se u ovom kontekstu odnosi na to da treba
  - u pravom trenutku nešto napraviti
  - propuštanje „pravog trenutka” je problem jer nosi **ozbiljne posljedice**
- kako ostvariti takvo upravljanje?
  - sustavi su složeni, ima puno stvari koje mogu biti uzrok greške
    - sklopovlje (mikroračunalo)
    - programska potpora (operacijski sustav, programi)
    - ostale komponente u sustavu (senzori, aktuatori, „vanjski utjecaji”)
  - sve komponente moraju biti ispravne!

# 1.2. Osnovni pojmovi

- razmatramo upravljačko računalo, i (uglavnom) samo programsku potporu, tj. kako ju ostvariti
- svi elementi sustava prikazani su na slici 1.6.




**Slika 1.6. Sustavi upravljani računalom**

# 1.2.1. Ispravnost, greške, zatajenje

- ispravnost – radi ono što se od njega traži
  - kako to bolje definirati? što ne smije raditi?
- „neočekivana stanja” – zbog grešaka
  - kratkotrajne greške (iz kojih se može izvući)
  - zatajenje (ne može se izvući)

# 1.2.2. Logička ispravnost

- Za mnoge sustave je logička ispravnost dovoljna
  - matematički proračuni
  - uredski posao
- Od svih sustava očekujemo logičku ispravnost
- Moguće je da ima više „ispravnih rješenja”
  - neka mogu biti bolja do drugih
  - izračun „boljih” rješenja može potrajati
  - ponekad si više vremena možemo uzeti, ako je rješenje bolje

# 1.2.3. Vremenska ispravnost

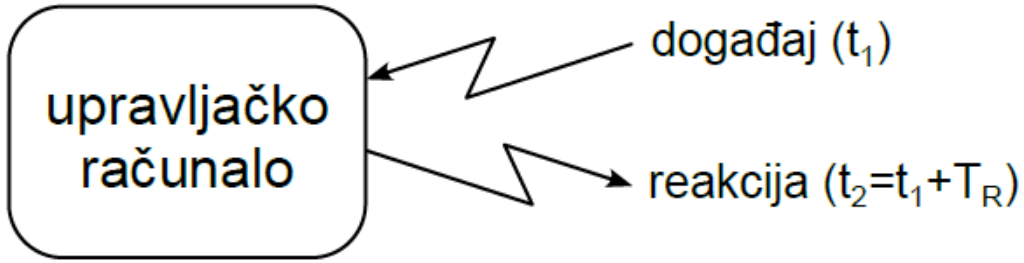
- rezultat mora biti generiran u pravom trenutku, ne kasnije
- „rezultat” može biti i naredba koja se nekamo šalje (na aktuator)
- ako rezultat/naredba dolazi prekasno (ili prerano!) upravljanje može biti krivo (i snosimo posljedice)
- ponekad je bitnije dati podatak (naredbu) u pravom trenutku nego čekati da se proračuna „bolje” rješenje koja onda dolazi prekasno
  - lošije rješenje („manje optimalno”) u pravom trenutku je bolje od „optimalnog rješenja” koje dolazi prekasno
- obično se upravljanje svodi na upravljačku petlju
  - „optimalno rješenje” možda i nije potrebno
  - u svakoj iteraciji petlje se daje nova naredba
  - možda i „lošija rješenja” dovoljno „brzo” pomiču sustav u željenom smjeru

# 1.2.4. SRSV

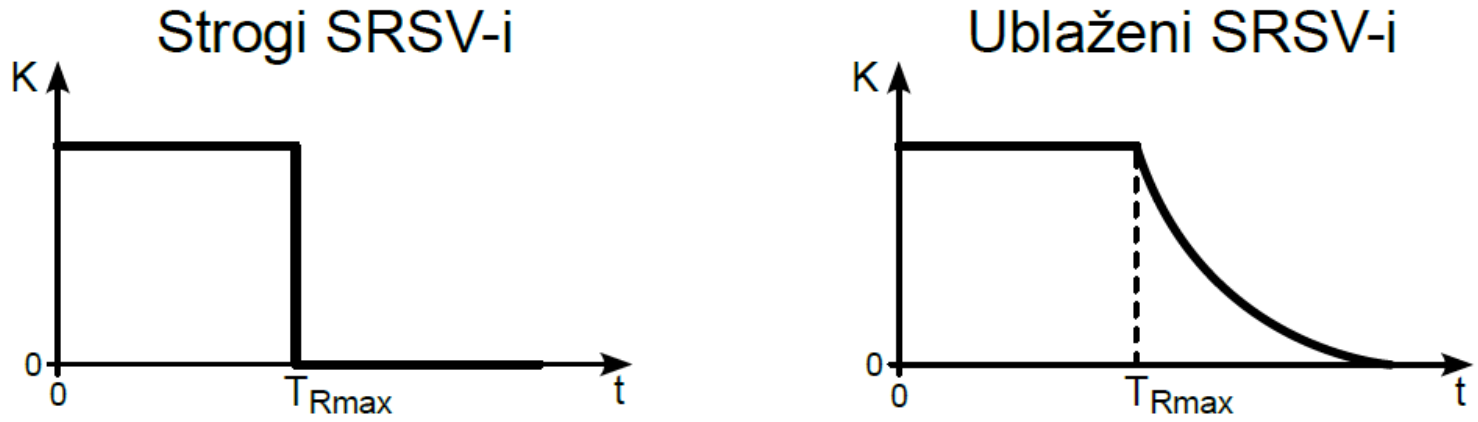
- kod SRSV-a je potrebna i logička i vremenska ispravnost

# 1.2.5. Podjela SRSV-a

- ima raznih podjela
- najčešća je prema vremenu reakcije, tj. posljedicama loše reakcije:
  - strogi – *hard RTS*
  - ublaženi – *soft RTS*
  - čvrsti – *firm RTS*




**Slika 1.7. Vrijeme reakcije na vanjski događaj**



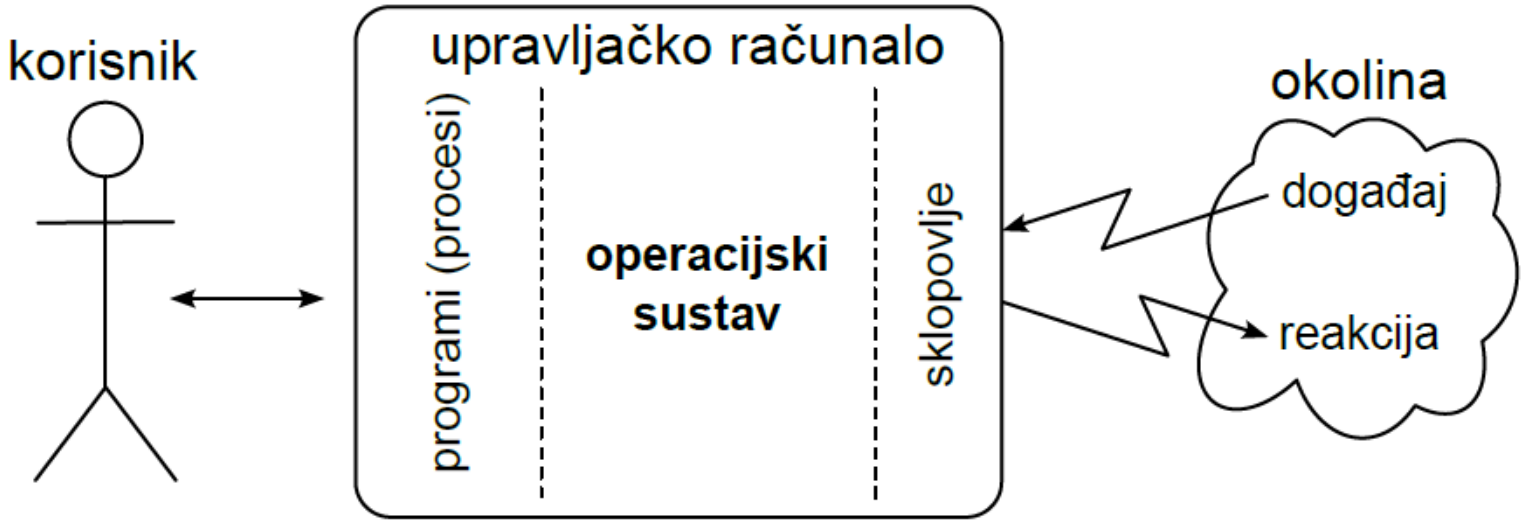

**Slika 1.8. Usporedba strogih i ublaženih SRSV-a prema vremenu reakcije**

# 1.2.6. Vrste događaja

- periodički
  - s vremenom, npr. svakih 50 ms treba napraviti „to i to”
  - najčešći u SRSV-u
- aperiodički
  - javljaju se rijetko, nepravilno pojavljivanje (često su to neke greške)
  - često se mogu dojavljivati mehanizmom prekida – nešto se dogodilo izvana
- sporadični
  - slični aperiodičkim (sitne razlike)
  - nepravilno se javljaju, ali obično je poznato minimalno vrijeme između dva javljanja
- sinkroni/asinkroni (s vremenom, drugim događajima)

# 1.2.7. Operacijski sustavi

- OS je dio svih netrivijalnih sustava
- jednostavni mikronoktroleri ne trebaju (pravi) OS, ali bi ipak dio programske potpore (prihvat prekida, upravljački programi naprava, i sl.) mogli označiti kao jednostavan OS
- nisu svi OS-i za SRSV
- „desktop OS”
  - brz, ali nije 100%
- RTOS
  - ne mora biti brz
  - ali reakcija u 100% slučajeva je ok




Slika 1.9. Uloga operacijskog sustava u računalnom sustavu

# 1.2.8. Sklopovlje za SRSV

- za SRSV treba voditi pažnju i o skopovlju
- osobno računalo (PC) je građeno za mnoge stvari (upotrebe)
  - također je građeno da bude modularno, da se razne fizičke komponente mogu u njega staviti, prema potrebi korisnika
  - ali zbog toga nema deterministično ponašanje koje je UVIJEK potrebno

# 1.2.9. Deterministično ponašanje

- u stanju **X** s ulazom **a** uvijek treba otići u stanje **Y**
- ako to nije tako onda se sustavom ne može upravljati (on je stohastički)

- problemi ostvarenja deterministčkog ponašanja
  - složenost sustava onemogućava predviđanje (izračun)
    - ne znamo što će se dogoditi s **a**
  - složenost algoritama (malo/veliko O()) unosi nedeterminizam jer iste operacije različito traju ovisno o stanju sustava

# 1.2.10. Pouzdanost

- kažemo kad sustav ispravno radi da je i pouzdan
- ali taj pojam nosi i „socijalnu” komponentu, treba korisniku dati povjerenje u ispravan rad sustava
  - kako to napraviti?
  - uvid u postupak izgradnje, prikaz testiranja, gdje se to već koristi (tko)

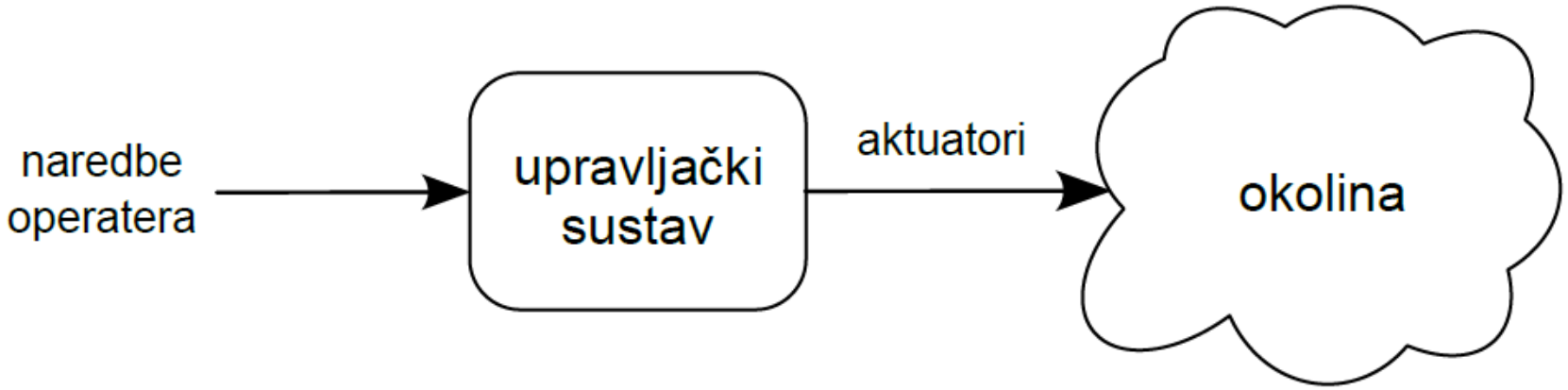
# 1.2.11. Robusnost

- predvidjeti i situacije van normalnih i izgraditi ponašanje i za njih
  - za to treba osim specifikacije koju daje sam naručitelj, poći u „pogon” i tamo još dodatno ispitati uobičajeni i manje uobičajeni postupak (što se sve može dogoditi
    - ponekad naručitelj nešto pretpostavlja da „svi znaju” ali to nije uvijek tako
- dodatne mogućnosti povećanja robusnosti:
  - redundantno upravljanje (više senzora, upravljačkih računala, aplikacija)
  - nadzorni alarm (o njemu detaljnije kasnije)

# 1.3. Načini upravljanja

- Kontinuirani i diskretni sustavi
  - računalo je diskretno
  - prilagodba frekvencije uzorkovanja senzora i izdavanja naredbi
  - reakcija na događaje
- Načini upravljanja
  - samo nadzor – bez utjecaja na sustav, ali bilježenje događaja i alarmiranje...
  - bez povratne veze
  - s povratnom vezom

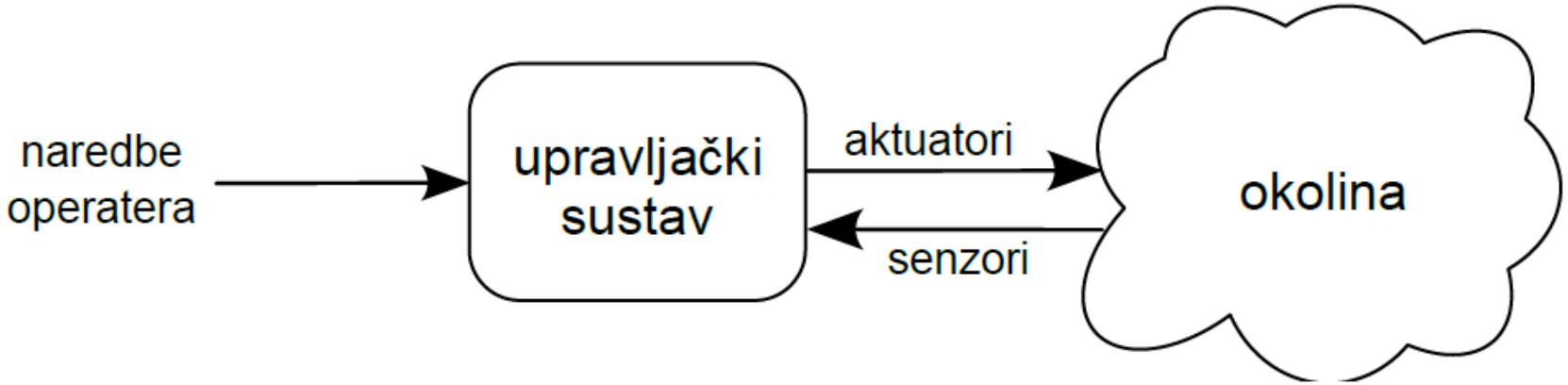
# 1.3.2. Upravljanje bez povratne veze




Slika 1.11. Upravljanje bez povratne veze

- predvidljivi sustavi, zna se (precizno) što je se dogoditi sa sustavom (okolinom) nakon naredbe **x**
- ili ako naredbe dinamički izdaje čovjek (koji je tada „povratna veza”)

# 1.3.3. Upravljanje primjenom povratne veze




Slika 1.12. Upravljanje uz povratnu vezu

- „naredbe” kažu u koje stanje sustav treba donijeti, ali se odluke donose na osnovu trenutna stvarna stanja (očitavajući senzore)
- složenije za ostvariti, ali neophodno za sustave bez izravnog nadzora čovjeka

# 1.4. Operacijski sustavi za SRSV-e (RTOS)

- koji OS odabrati (više o tome u zadnjem poglavlju)
  1. kupiti RTOS? „super”, ali to košta, tko će programirati programe?
  2. običan OS? ali ima li dovoljno dobra vremenska svojstva?
  3. napraviti novi OS? to traje, greške? na kraju možda skuplje od kupnje
- možda RTOS nije potreban?
  - može je dovoljan prilagođeni „obični” OS?
- najčešće je potreban kompromis
  - netko mora odlučiti što
  - bilo koja od gornjih opcija može biti ispravna u nekim slučajevima

# 1.5. Raspoređivanje zadataka u SRSV-ima

- netrivijalni sustavi koriste OS i zadatke (dretve) koje treba „rasporediti”
  - rasporediti = odabrati koji će se obavljati u pojedinom trenutku
- uobičajeni načini raspoređivanja:
  - prema prioritetu (najčešći za SRSV, najjednostavniji za izvesti, ali ...)
  - prema redu prispijeća (najjednostavniji za izvesti, ali ...)
  - podjelom vremena – pravedan, ali je li i najboljih?
- o toj temi u zasebnim poglavljima
- ukratko
  - najčešće prema prioritetu
  - ako nije dovoljno onda predimenzionirati (jači procesor)
  - vrlo rijetko nešto drugo (ali ipak da)

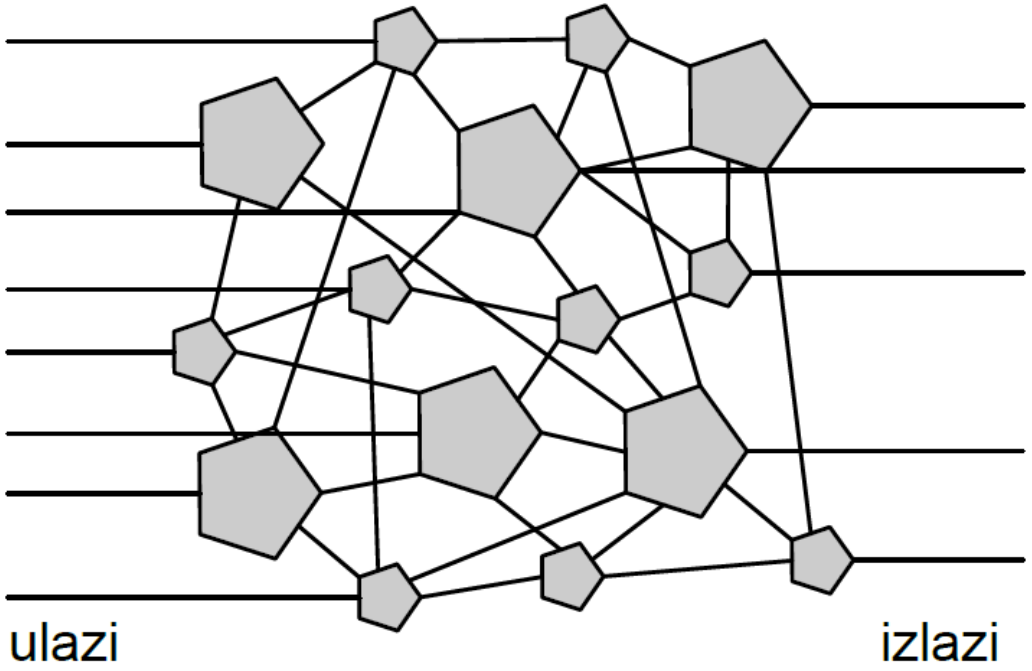
# 1.6. Primjeri SRSV-a upravljanih računalom

- u vozilima: kočenje, rad motora, održavanje brzine/razmaka, prepreke, ...
- roboti: u medici, u industriji, ...
- složeni sustavi: zrakoplovi, elektrane, energetski sustavi

# 1.7. Problem složenosti

- ❑ većina sustava je složena
- ❑ složenost onemogućava da se „sve jasno vidi”
- ❑ lako se u rješenje unesu „greške”

- ❑ kako onda ostvariti takve sustave?




Slika 1.13. Složeni sustav

# 1.7. Problem složenosti (2)

- težiti jednostavnim rješenjima (bar za dijelove)
- ***najjednostavnije rješenje je najbolje rješenje***
- smanjiti složenost (tj. upravljati složenošću) sa:
  - podjelom složena sustava na dijelove (slojeve, podsustave, razine, ...)
  - koristiti modele (prikaze samo iz nekih perspektiva)
  - koristi grafičke alate (skice, UML, Petrijeve mreže)
- sam proces implementacije mora biti prilagođen
  - „postupci dobre inženjerske prakse”
  - programsko inženjerstvo (specifikacija zahtjeva, oblikovanje arhitekture...)

# 1.7.2. Metoda “podijeli i vladaj”

- najčešće korištena metoda
- složenost ne nestaje, ali je po pojedinom dijelu manja
  - dijelove je tada lakše (ispravno) napraviti
- primjer primjene na operacijski sustav:
  - podjela izvorna koda na slojeve: sklopovlje, jezgra, sučelje za programe
  - podjela na podsustave: UI, pohrana, procesi i dretve, mreža, ...
- nedostatci podjela
  - dodatna sučelja, višestruko kopiranje podataka, ...
  - problem za sustave gdje je potrebna visoka učinkovitost (*high performance*)
  - moguće rješenje: tunelirati kroz neke slojeve, spojite neke komponente i sl.

# 1.8. Uobičajeni pristupi ostvarenja upravljanja

- Uglavnom se upravljanje svodi na:
  - reakcija na vanjske događaje
  - periodički poslovi

- Uobičajena struktura koda u rješenjima:
  - upravljačka petlja
  - prekidi
  - alarmi
  - dretve

# 1.9. Zašto proučavati SRSV-e?

- izgradnja SRSV-a zbog mogućih posljedica grešaka je mnogo detaljnija (stroža, temeljitija) od izgradnje programske komponente za obične sustave, ali ta znanja su primjenjiva i inače
- primjeri „stečenih znanja”:
  1. odabir sklopovskih i programskih komponenti
  2. odabir (komercijalno) dostupnog operacijskog sustava
  3. odabir razvojnih alata
  4. maksimiziranje otpornosti na greške i povećanje pouzdanosti
  5. oblikovanje, planiranje i izvođenje ispitivanja tijekom cijelog razvojnog procesa
- naučiti kako predviđati ponašanje sustava, kako očitovati njegovo ponašanje, kako mjeriti vrijeme reakcije te kako ga smanjiti