Tosiaikajärjestelmät – Luento 4: Mallinnus ja mittaaminen

Tiina Niklander

22.3.2006

Lähteinä mm: Burns & Wellings luku 12

B.P. Douglass: Designing real-time systems with UML. Embedded System Programming,1998.

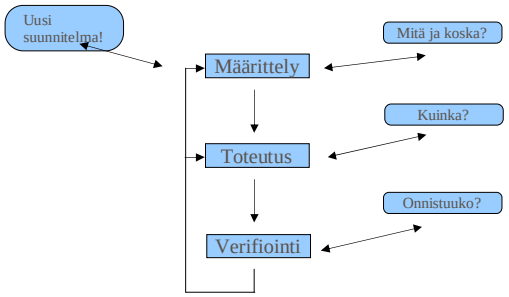
Sisältö

- Ohjelmistotuotanto ja tosiaikajärjestelmä
- Mittaus
- Mallinnusmenetelmiä
- WCET (Worst-case estimation time)
- Aikavaatimuksia ja ajan verifiointi

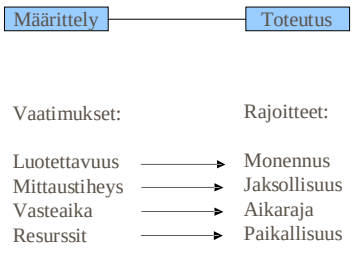
Ohjelmistotuotanto ja tosiaikajärjestelmät

- Tuotantoprosessi
- Ohjelmiston rajoitteet
- Suunnittelupäätöksiä
- Verifiointi

Tosiaikajärjestelmän suunnittelu



Määrittely ja toteutus



Rajoitteita

- Ajalliset rajoitteet: esim. jaksollisuus, aikarajat
- Poissulkeminen
- Suoritusjärjestys
- Paikallisuus
  - Jokin tehtävä suoritettava tietyllä prosessorilla
- Ryvästys
  - Samassa paikassa: toiminnallisuus, suorituskyky
  - Eri paikassa: luotettavuus, suorituskyky

Mistä aikarajoitteet tulevat?

- Luonnonlait
  - Liikkeen kesto: robotin käsivarren liikkeet
  - Silmän erottelukyky: kuvia sekunnissa
- Matemaattiset teoriat
  - Näytteenottotaajuus
  - Shannonin teoreema
    - sitoo siirtokanavan kaistaleveystarpeen ja signalointinopeuden toisiinsa
- Komponenttien rajoitukset
  - Sensorit: minimiaikaväli peräkkäisille operaatioille

Rajoitteiden kriittisyys?

- Kovat aikarajat: laskennan tulos hyödytön, jos ei aikarajojen puitteissa
  - Ei-kriittinen: järjestelmä voi vielä toimia jotenkuten, vaikka laskenta viivästyi
    - Navigointi, diagnostiikka
  - Kriittinen: järjestelmän toiminta ei voi jatkua
    - Lennonvalvonta, jokin muu valvontajärjestelmä
  - Turvallisuuskriittinen: toimintahäiriö voi olla hengenvaarallinen
    - ABS-jarrut, suojausjärjestelmä (robotit)

Rajoitteiden kriittisyys?

- Pehmeät aikarajat: yksittäinen laskennan viivästyminen on yleensä sallittua, mutta sen käyttökelpoisuus laskee (usein hyvin lähelle hyödytöntä)
  - Varausjärjestelmät
    - Paikanvaraus, raha-automaatti
  - Multimedia
    - Video-on-demand, verkkopelit, virtuaalitodellisuus

Mitä pitäisi tehdä

- Kovat rajoitteet:
  - Oikeellisuus on verifioitava ennen kuin järjestelmä otetaan tuotantokäyttöön
- Pehmeät rajoitteet:
  - Tilastollinen analyysi usein riittää (esim. testaustulosten perusteella)

Suunnittelupäätöksiä 1/3

- Ohjelmointiparadigma
  - Peräkkäisohjelmointi
    - Vain yksi silmukka, jossa kaikki toiminnallisuus
  - Rinnakkaisohjelmointi
    - Ohjelmassa useita tehtäviä
    - Mallinnetaan rinnakkainen tehtävien suoritus
      - Yksiprosessorikone: vain pseudorinnakkaisuutta
      - Moniprosessorikone: todellinen rinnakkaisuus mahdollista

Suunnittelupäätöksiä 2/3

- Laitteistoarkkitehtuuri
  - Yksi- vai moniprosessorijärjestelmä:
    - Määrää todellisen rinnakkaisuuden asteen
  - Prosessoriarkkitehtuuri
    - Mikrokontrolleri, RISC-prosessori
    - Määrää WCET-analyysin kustannuksia
  - Verkkoarkkitehtuuri
    - Jaettu verkko vai yksittäiset yhteydet

Suunnittelupäätöksiä 3/3

- Suoritusympäristö
  - Järjestelmäpalvelut
    - Käyttöjärjestelmä vai itseriittoinen ratkaisu
  - Suoritusmalli
    - Ajoitus: aikaviipaleita vai prioriteetteja
    - Keskeytys: sallitaan vai ei sallita
  - Kommunikointimalli (sanoman välitys)
    - Aika- vai prioriteettipohjainen
    - Synkroninen vai asynkroninen
    - Yksi- vai kaksisuuntainen

Verifiointi

- Testaus:
  - Kokeilutestaus
    - Koekäyttöön, jos ei virheitä, niin toimii
    - Nopea vaihtoehto, valitettavan usein ainoa
  - Kattava testaus
    - Kaikki syötteet, ajoitukset ja vikatilanteet
    - Vaatii aivan liian paljon aikaa

Verifiointi

- Formaali analyysi:
  - Looginen oikeellisuus ”todistuskoneella”
    - Vaatii formaalin kuvauskielen
    - Hyvin korkea abstraktiotaso
  - Ajallinen oikeellisuus ajoitettavuusanalyysillä
    - Tärkeä koville tosiaikajärjestelmille
    - WCET jokaiselle tehtävälle
    - Ohjelmointikielen ja suoritusympäristön tuettava

Verifioinnin tulos pätee vain, jos kaikki oletukset voimassa suoritusaikana

Formaalin verifioinnin epävarmuustekijöitä

- Epädeterminismiä WCET analyysissä (tehtävän sisäinen)
  - Syötteet ja suoritustilat vaikuttavat polkuun
  - Muistinkäyttötapa vaikuttaa prosessorin viipeisiin (liukuhihna, cache)
- Konflikteja jaettujen resurssien käytössä (tehtävien välinen)
  - (Pseudo)rinnakkaisten tehtävien suoritusjärjestys voi aiheuttaa odotusta

Sisältö

- Ohjelmistotuotanto ja tosiaikajärjestelmä
- **Mittaus**
  - **Suorituskykymittaus**
  - **Erilaisia mittareita**
- Mallinnusmenetelmiä
- WCET (Worst-case estimation time)
- Ohjelmointikieliä
- Aikavaatimuksia ja ajan verifiointi

Tosiaikajärjestelmän suorituskyky?

- Valitse käyttökelpoinen mittari (metriikka)
- Tee suorituskykyanalyysi
  - Valitse sopiva evaluointimenetelmä (teoreettinen vs. kokeellinen analyysi)
- Vertaa erilaisten suunnitelmien suorituskyky
- Tunnista perustavanlaatuiset rajoitukset
  - Etsi suorituskyvyn 'pullonkaulat'

Mittareita

- Millainen on hyvä mittari?
  - Kiteyttää järjestelmän keskeiset ominaisuudet yhteen tai muutamaan numeeriseen arvoon
  - Tarjoaa 'objektiivisen' menetelmän vaihtoehtojen arviointiin tiettyyn tarpeeseen
  - Mahdollistaa suunnitelman arvioinnin tai optimoinnin
  - Perustuu varmennettaviin tietoihin (verifiable facts)

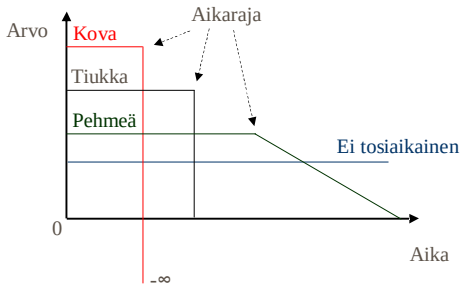
Perinteisiä mittareita

- Luotettavuus (reliability)
  - Todennäköisyys että mikään osa järjestelmää ei vikaannu annetun aikayksikön kuluessa
- Saatavuus (availability)
  - Järjestelmän toiminta-ajan osuus koko ajasta
- Suoritusteho (throughput)
  - Keskimääräinen suoritettujen operaatioiden lukumäärä aikayksikössä

Tosiaikajärjestelmille sopivia mittareita

- Löyhyys (Laxity) ennen ajoitusta
  - Aika, jonka tapahtumaa voi viivästää ilman myöhästymistä
- Myöhäisyys (Lateness) ajoituksen jälkeen
  - Tapahtuman myöhästymisen kesto
- Myöhästyneet (Late tasks) ajoituksen jälkeen
  - Myöhästyneiden tapahtumien lukumäärä

Yleinen kustannusfunktio



Suorituskyvyn suunnittelu

- Tavoitteena:
  - Rakennetaan suorituskyky järjestelmään jo suunnitteluvaiheessa
  - Analysoidaan jo suunnitelmaa ja etsitään sen perusteella parannettavia kohtia
- Oma kurssi:
  - Ohjelmistojen suorituskyvyn suunnittelu

Sisältö

- Ohjelmistotuotanto ja tosiaikajärjestelmä
- Mittaus
- **Mallinnusmenetelmiä**
  - **Yleistä**
  - **UML esimerkki**
- WCET (Worst-case estimation time)
- Aikavaatimuksia ja ajan verifiointi

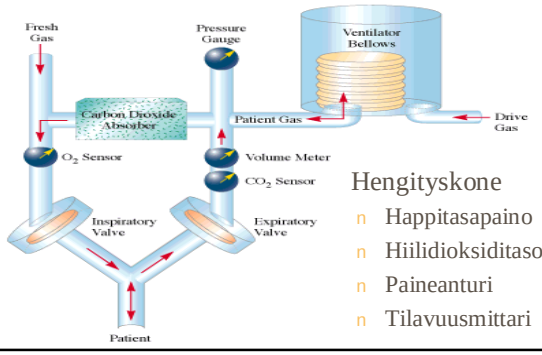
Mallinnusmenetelmiä

- Useita erilaisia mallinnusmenetelmiä
- UML:ään on olemassa joitakin aikalaajennoksia
- HRT-HOOD (Burns & Wellings käyttää)
- SDL & MSC
- ... (paljon muita)

Mallinnusmenetelmiä

- Tutustutaan UML:ään
- OCL:llä (Object Constraint Language) voi kuvata myös aikarajoitteita
- Tässä käytetään B.P. Douglassin esimerkkiä Gasp-o-matic artikkelista, joka on ilmestynyt Embedded Systems Programming -lehdessä vuonna 1998. Linkki v. 2004 englanninkielisiltä kurssisivuilta www.embedded.com:n

Esimerkki: Gasp-o-matic

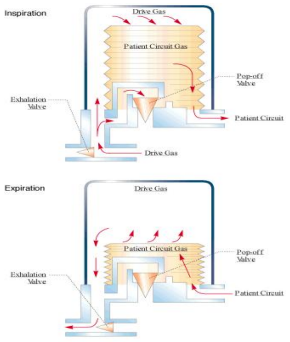


Hengityskone

- Happitasapaino
- Hiilidioksiditaso
- Paineanturi
- Tilavuusmittari

Hengityskoneen toiminta

- Sisäänhengitys
  - Ulkoinen paine
  - Ylipainesuoja
- Uloshengitys
  - Kehon paine
  - CO2 -anturi
  - tilavuusmittari



Hälytysesimerkkejä

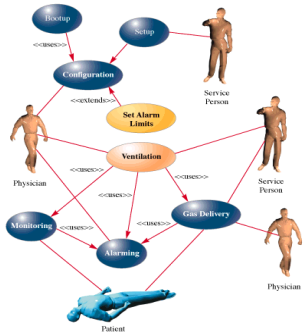
Hälytys	Kriittisyys	Kuvaus
Apnea	Kriittinen	Potilas ei hengitä
Matala O2	Kriittinen	Happipitoisuus ilmaa alhaisempi
Korkea paine	Kriittinen	Puristuspaine liian korkea, -> hengitystietukos
Matala paine	Kriittinen	Puristuspaine liian matala, -> vuoto, ei kytketty
Matala Co2	Kriittinen	Uloshengityksen Co2 liian matala -> ruokatorvessa
Sisäinen virhe	Kriittinen	Järjestelmän sisäinen virhe
Asetettua alh. minuuttitilavuus	Huomautus	Säädettyä minuuttitilavuutta ei saavutettu
Asetettua alh. Kertatilavuus	Huomautus	Säädettyä kertatilavuutta ei saavutettu
Kertatilavuus virheellinen	Huomautus	Säädettyä kertatilavuutta ei voida toteuttaa
Huolto <pvm>	Tiedote	Käynnistettäessä tuleva huoltotiedote 30 pv ennen
Versionumero	Tiedotus	Järjestelmän käytössä oleva versionumero

UML:n kaaviot

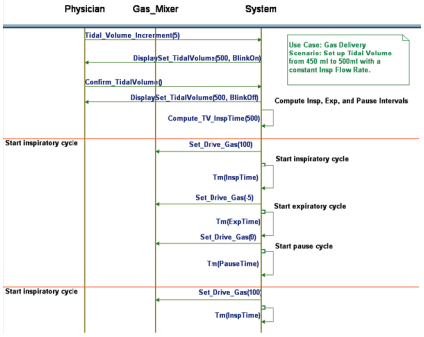
- Vaatimusanalyysiin sopii
  - Käyttötapauskaavio (use case diagram)
    - Ulkopuoliset toimijat, järjestelmän palvelut
    - Skenaario: käyttötapauksen ilmentymä
  - Sekvenssikaavio (sequence diagram)
    - Toimenpiteen kuvaaminen viesteinä
  - Yhteistyökaavio (collaboration diagram)
    - Olioiden välinen yhteistoiminta
  - Aktiviteettikaavio (Activity diagram)
    - Kontrollin kulun kuvaaminen
    - Käyttötapauksille, mutta usein hiukan myöhemmin

Gasp-o-matic: Käyttötapauskaavio

- Käyttäjät
- Toiminnot tai tapaukset
- Toiminnot kuvataan kaavion lisäksi muilla kaavioilla tai sanallisesti



Sekvenssikaavio

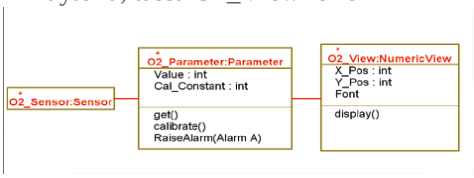


Vaatimuksista suunnitelmaksi

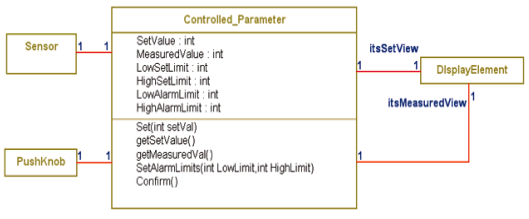
- Tunnistetaan oliot ja niiden luokat
- Olioiden perustoiminnallisuus
- Uusia kaavioita, joilla kuvataan uusia asioita
  - Luokkakaavio (class diagram)
  - Oliokaavio (object diagram)
  - Tilakaavio (statechart diagram)
    - Yhden olion tila ja sen muutokset

Oliokaavio

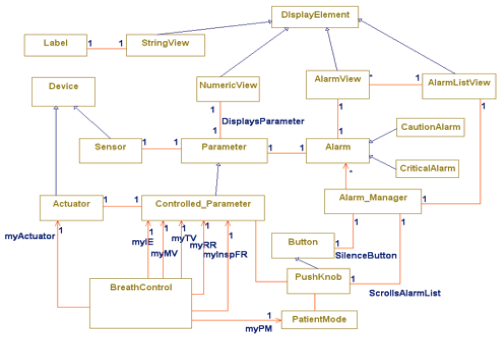
- Olioiden nimet,luokat, parametrit
- Suhteet muihin olioihin
- Gasp-o-Matic: jokaisella oliolla on edustaja näytöllä, tässä O2_View -olio



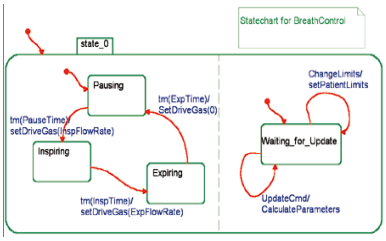
Luokkakaavio



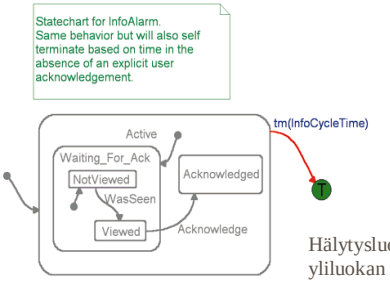
Luokkakaavio



Tilakaavio



Tilakaavio: HälytysTiedote



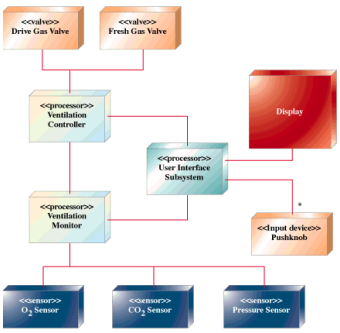
Hälytysluokan aliluokka, yliluokan toiminnot harmaalla

Lähemmäs toteutusta

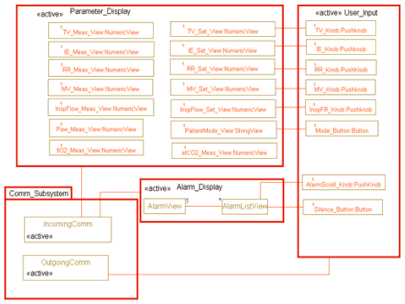
- Olioiden sijoittelu säikeisiin ja prosessoreille
- Arkkitehtuuriratkaisut
- Käytettävät kaaviot:
  - Sijoittelukaavio (deployment diagram)
    - Komponenttien sijoittelu fyysisiin solmuihin (laitteet)
  - Komponenttikaavio (component diagram)
  - Yhteistyökaavio (collaboration diagram)

Sijoittelukaavio

- Fyysinen arkkitehtuuri
- Laitteisto-komponentit:
  - Venttiilit
  - Anturit
  - Näyttö
  - Prosessorit



Tehtävät ja oliot



Oliokaavio, jossa vain aktiivit tehtäväoliot

Rajoitteet (constraint)

- Rajoitteiden avulla muokataan kaavioiden semantiikkaa
- Ilmaistaan luonnollisella tai formaalilla kielellä esim. OCL
- Väline **ajan** ilmaisemiseksi
  - {j.time -i.time <5s}
  - {suoritus alkaa klo 9:30}
  - Ajoitusmerkki: msg.sendTime() tm(tapahtumakesto)

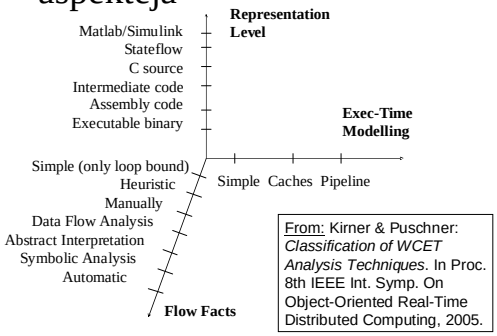
Sisältö

- Ohjelmistotuotanto ja tosiaikajärjestelmä
- Mittaus
- Mallinnusmenetelmiä
- **WCET (Worst-case estimation time)**
- Ohjelmointikieliä
- Aikavaatimuksia ja ajan verifiointi

WCET (Worst-case execution time)

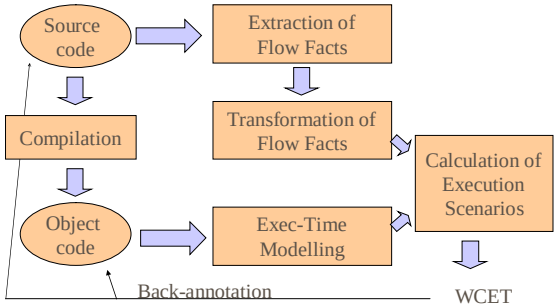
- Joko mittaamalla tai analysoimalla
- Mittaus: Oliko tämä nyt pahin mahdollinen tilanne?
- Analyysi: Prosessorin tarkka malli?
- Yleensä kaksivaiheinen:
  - Ensin: pura ohjelma peräkkäisiin suorituslohkoihin
  - Sitten: analysoi kunkin lohkon suoritusaika prosessorimallin avulla

WCET - ortogonaalisia aspekteja



From: Kirner & Puschner: *Classification of WCET Analysis Techniques*. In Proc. 8th IEEE Int. Symp. On Object-Oriented Real-Time Distributed Computing, 2005.

Generic WCET Analysis Framework



Semanttinen tieto voi auttaa

```
for I in 1.. 10 loop
  if Cond then
    -- basic block of cost 100
  else
    -- basic block of cost 10
  end if;
end loop;
```

- Yksinkertainen laskelma 10*100 (+yleiskuorma) => 1005.
- Jos tiedetään että `Cond` voi olla tosi korkeintaan 3 kertaa yhden suorituksen aikana on pahinkin kustannus enää 375.

Sisältö

- Ohjelmistotuotanto ja tosiaikajärjestelmä
- Mittaus
- Mallinnusmenetelmiä
- WCET (Worst-case estimation time)
- **Aikavaatimuksia ja ajan verifiointi**
  - **Ajalliset rajoitteet**
  - **Ajoitushäiriöt**

Aikavaatimuksia ja ajan verifiointi

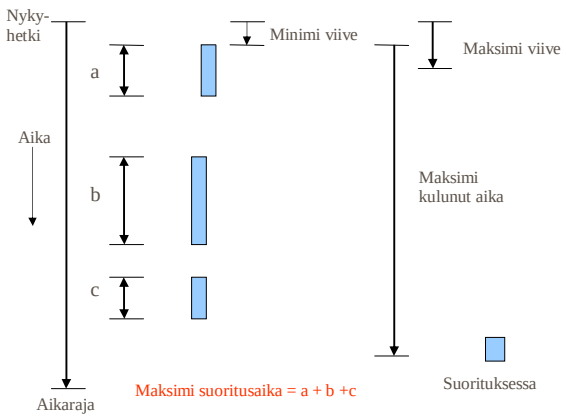
- Kaksi teoreettista lähestymistapaa aikaan
- Formaalisti määritellyt kielten semantiikat ja aikavaatimukset, logiikka vaatimusten kuvaamiselle
- Tosiaikajärjestelmän suorituskyky ja ajoitettavuusanalyysi: tehtävien suoritettavuus annetuilla resursseilla

Ajan verifiointi

- Kaksi vaihetta
- Vaatimusten verifiointi
  - Oletetaan äärettömän nopea tosiaikainen kone, ovatko vaatimukset johdonmukaisia (coherent) ja yhtenäisiä (consistent): voidaanko toteuttaa?
- Toteutuksen verifiointi
  - Voidaanko aikavaatimukset saavuttaa annetuilla äärellisillä (ja epäluotettavilla) (laitteisto)resursseilla?

Erilaiset ajalliset rajoitteet

- Aikaraja (deadline)
  - Suorituksen on päätyttävä tähän mennessä
- Minimi viive (minimum delay)
  - Minimiaika, joka pitää odottaa, ennen suoritusta
- Maksimi viive (maximum delay)
  - Maksimi odotusaika ennen suoritusta
- Maksimi suoritusaika (maximum exec. time)
- Maksimi kulunut aika (maximum elapse time)



Ajoitushäiriöt ja vikasietoisuus

- Havaittava:
  - Aikarajojen ylitykset
  - Maksimisuoritusajan ylitys (WCET)
  - Jaksollisen tapahtuman tapahtuminen ennustettua (tai suunniteltua) useammin
  - Kommunikoinnin aikakatkaisut
- Listan kolme viimeistä eivät aina ennakoi koko tehtävän viivästymistä
  - Päätettävä toimenpiteet tapauskohtaisesti
  - Voidaan toipua eteen- tai taaksepäin

Aikarajan ylitys

- Aikaraja voidaan ylittää myös 'todennetussa' järjestelmässä, jos
  - WCET analyysi oli epätarkka
  - Ajoitettavuusanalyysin oletukset eivät päde
  - Ajoitettavuusanalyysissä itsessään oli virhe
  - Skedulointialgoritmi ei selviäkään kuormasta, vaikka sen teoreettisesti pitäisi
  - Järjestelmä toimii suunniteltujen rajojen ulkopuolella

Suoritusajan ylitys

- Rajoitettava virheen seuraukset hyvin määritellylle ohjelman osalle.
- Prosessi, joka käyttää odotettua enemmän aikaa, saa ylittää aikarajansa.
- Jos korkean prioriteetin prosessi käyttää enemmän aikaa, saattaa alemman prioriteetin prosessi ylittää oman aikarajansa.
- Aikapoikkeamat pitäisi käsitellä ne aiheuttaneessa prosessissa.

Jaksollisen tapahtuman toisteisuus

- Liian usein tapahtuminen voi sekoittaa ajoituksen
- Estettävä tai havaittava ja korjattava
- Estäminen
  - laitteistokeskeytysten estäminen (ajurin param.)
  - Jaksottava palvelin (sporadic server)
    - Palvelee satunnaisia tapahtumia jaksollisessa ympäristössä; voi samalla valvoa jaksollisten saapumista.
- Havaitseminen: EI riittävää tukea ohj.kielissä

Yhteenvetona:

- Tosiaikaisten järjestelmien suunnitteluun on kehitetty ja kehitetään edelleen runsaasti erilaisia menetelmiä
- Huomioitava erityisesti
  - Ajan kuvaaminen
  - Rajoitteiden käsittely
  - Mallin luotettavuus