Tilstandsmaskinparken

= f(Scheduler)

Øyvind Teig

ansatt i
Autronica Fire
and Security
Trondheim
Norway

http://home.no.net/oyvteig/

men foredraget
står for egen regning

Ekstranotat ved gjesteforelesningen på NTNU, fag TTK4145 Real-time programming, nov. 2008

Dette er en historie..

* .. om samlivet mellom en prosess og en scheduler
* For om scheduleren ikke er "preemptive"..
* men kooperativ "run to complete"

  - dvs. schedule start av prosess
    som switch/case'er til tilstand
    Nok for asynkron (FSM/SDL)

  - og ikke også
    schedule til neste linje uten tilstand
    Godt for synkron (CHAN/CSP)

- [x] da kan man oppdage denne historien

For da fikk vi nyttige men strengt tatt unødvendige sub-maskiner

* Buffer-prosesser har
  * bare **kommunikasjons(sub)**tilstander
* Applikasjons-prosesser fikk
  * **applikasjons**tilstander og **kommunikasjons**tilstander
  * unødvendige
  * **kommunikasjonssub**tilstander

strengt tatt

Jeg, som trodde det skulle bli enkelt!

* Men det ble ikke så greit
* å gjenbruke tilstandmaskins

monster

```
switch (state)
    case ALT_1
    case IN_1
    case OUT_1
    case APPL_1
    case APPL_2
```

* ref:

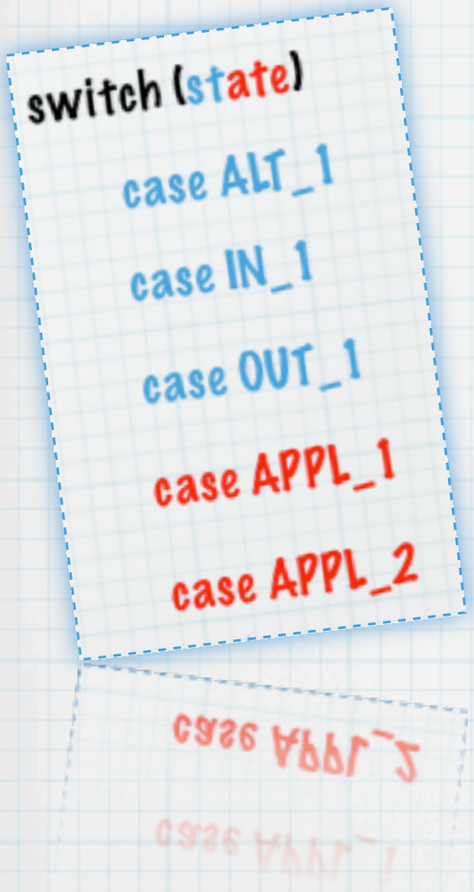
http://home.no.net/oyvteig/pub/NTNU_2008/presentation_w3ref.html
http://home.no.net/oyvteig/pub/ercim05_at_euromicro-2005/paper.pdf
http://home.no.net/oyvteig/pub/CPA2006/paper.pdf

eneste "koden" i denne presentasjonen
= en "maskinpark" på to tilstandsmaskiner: appl og komm-sub

CHAN_CSP

* Jeg bestemte meg for bygge på SPoC
  Southampton Portable occam Compiler
  som generer tilstandsmaskinkode i C fra occam kildekode
* og bygge oppå et AFS-utviklet FSM kjøresystem
  FSM = Finite State Machine med asynkrone meldinger

* Sjøl om jeg hadde års erfaring med occam
* og hadde sett en del CSP bibliotek
* så ble komm- og appl -koden som i SPoC
* og kodemessig lik appl-koden i FSM-systemet



* jeg var vant med å se maskingenererte SPoC-type tilstandsmaskiner i C
* men de som kunne FSM oppfattet kanal-tilstandene som forstyrrende
* for de var vant med at den strukturen ble brukt mest til applikasjon
* selv om de nye representerte allment akspetert, lagdelt, tilstandsmaskins-metodikk

Hva brukes tilstander til?

* Kanal-maskinene var "blokkerende", noe helt nytt, med en tilstand per "kanalbruk"
* for, mens asynkrone meldinger kunne sendes og mottas, hvor mange man ville, per applikasjons-tilstand
* så det ut til at kanaltilstandene blandet seg inn i applikasjons-tilstandene

Jeg hadde ikke tenkt

* at det ville vært best å gå fra asynkron metodikk til synkron metodikk
* ved at kanalbruk også kunne settes rett under hverandre, uten eksplisitte tilstander (som jeg var vant med)
* Forståelsesmessig ble det nye uvant
* Men dette kan forandres!

Modifisert kjøresystem

* Jeg måtte ha skrevet om scheduleren litt
* i stedet for bare ett fast call-entry-punkt
* måtte den nye scheduleren kunne kjøre fra neste linje etter det punktet vi returnerte fra i prosessen
* Scheduleren ville fremdeles ikke være preemptiv, men måtte dynamisk lagre neste adresse

I tillegg

* måtte den nye scheduleren ha lagt på plass peker til de interne prosessdataene
* og vi måtte ha forholdt oss til stack: det enkleste ville vært å hatt også lokale data i det interne prosessdatafeltet ("local context")
* for det ville bare vært kanalprimitivene som trengte denne måten å bli penset inn
* Kalte funksjoner kunne bruke stack uhemmet

Likevel, for at "maskingenerert" kode

* skrevet av programmereren her
* skal forstås av programmereren der

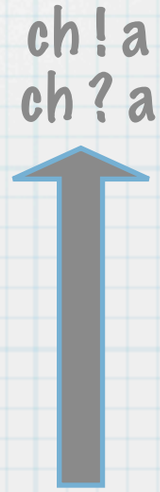
* er egentlig ikke verre enn å forstå f.eks. OO arv eller annen hierarkisk tenking
* Man må "opp et trinn" for å forstå helt

Kanskje

* modifiserer vi scheduleren litt
* kanskje ikke..
* for, som regel er det applikasjons-tilstander mellom kommunikasjons-tilstandene likevel..
* og det vi har er både rett og lærbart

Noe er mer elegant

* for uansett hvilket språk vi koder i
* lærer vi oss alltids metodikkene
* Likevel, vi trenger disse tette prosessene
* noe å putte OO inn i
* aller helst som førsteordens primitiver i språkene
* Analyse av beslektet tematikk:
  http://home.no.net/oyvteig/pub/pub_details.html#CSP:arriving_at_the_CHANnel_island



utdrag fra en drøm jeg må ha hatt

Tilstandsmaskinparken = funksjon av(Scheduler)

En scheduler er ikke så usynlig som ~~du tror~~ jeg trodde

dvs. Hvor skal pensen stå?

Spørsmål?