

Fra sekvensielt til parallelt

«Sanntidprogrammering etter 33 år»

Øyvind Teig
senior utviklingsingeniør

Autronica Fire and Security, «a UTC company»

Gjesteforelesning på Høgskolen i Sør-Trøndelag (HiST) 13.nov. 2009

Fra sekvensielt til parallelt

«Sanntidprogrammering etter 33 år»



frem-
deles
gøy!

Øyvind Teig
senior utviklingsingeniør

Autronica Fire and Security, «a UTC company»

Gjesteforelesning på Høgskolen i Sør-Trøndelag (HiST) 13.nov. 2009



AFS: brannvarsling

Fra brannmeldere

Fra brannmeldere

til kontrollere

Fra brannmeldere

til kontrollere

til sentraler

Fra brannmeldere

til kontrollere

til sentraler

(neste time med Ommund Øgård)

Sekvensielt

C

Sekvensielt

C

C++

Sekvensielt

C

C++

Java

Sekvensielt

C

C++

Java

main()

Sekvensielt

C

C++

Java

```
main()  
new ThisObject(); new ThatObject();
```

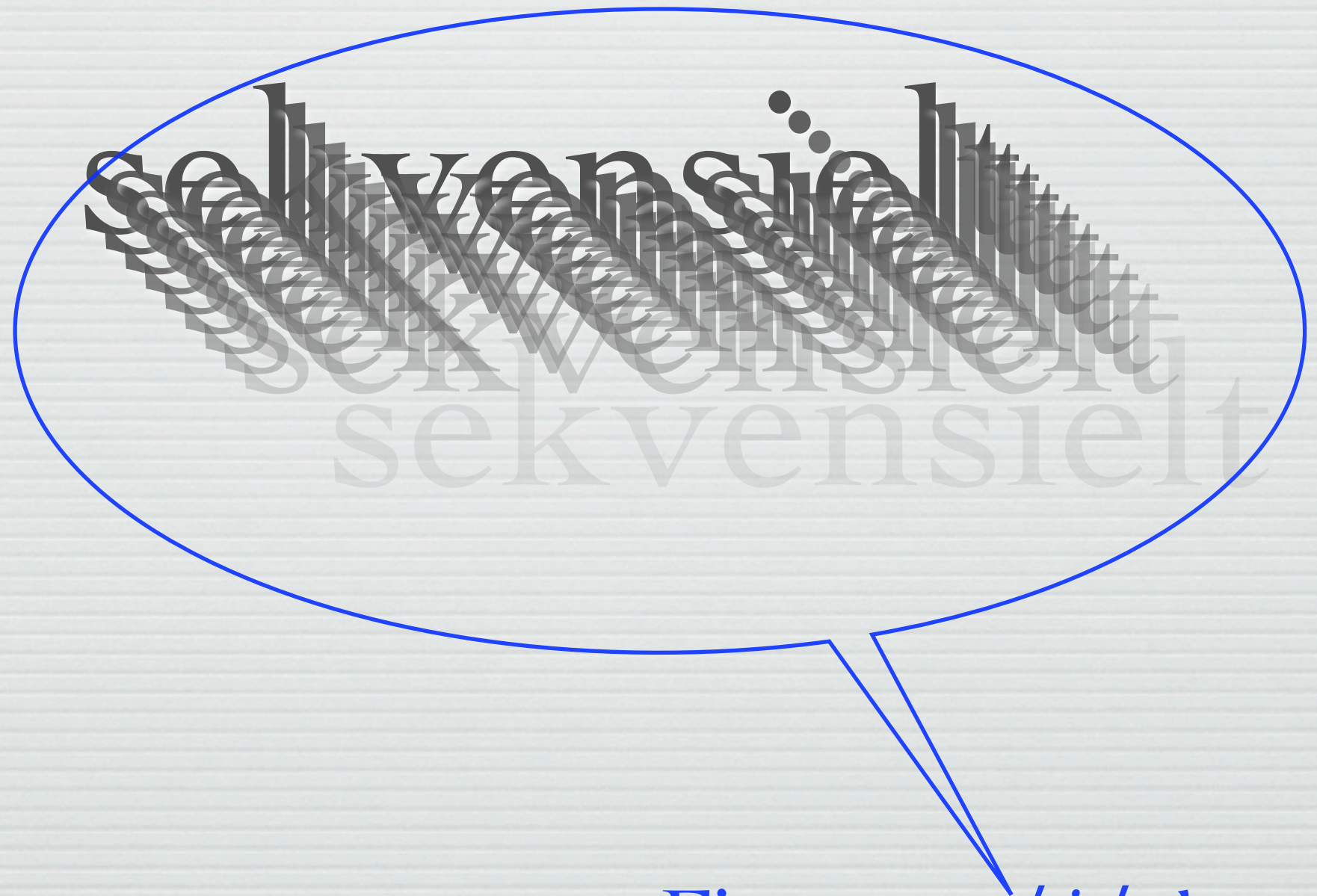

Fremdeles like

C

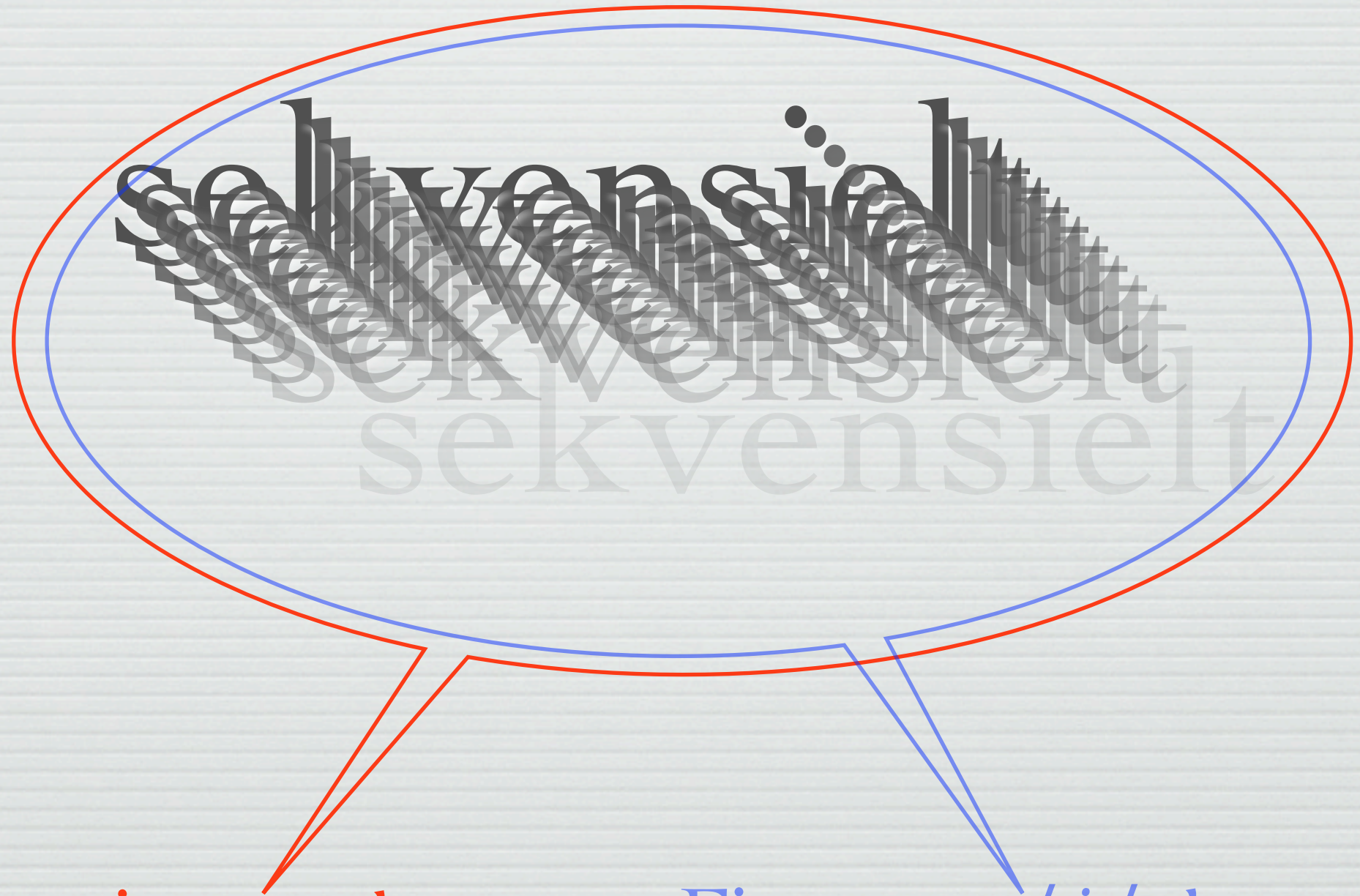
C++

Java

```
main()  
new ThisObject(); new ThatObject();
```

Fint som *objekt*!



Fin innmat i *prosess*!

Fint som *objekt*!

Objekt

-orientert
programmering

Prosess -orientert programmering

Objekt
Prosess
-orientert
programmering
ja, takk!

Objekt

Prosess

-orientert

programmering

ja, takk!

Hvilke, når, hvor?

Prosess

Prosess

Startes

lik objekter eller *function* med *static* data

Prosess

Startes

lik objekter eller *function* med *static* data

Kjører

av seg sjøl

Prosess

Startes

lik objekter eller *function* med *static* data

Kjører

av seg sjøl

Er tette

ingen lekkasje

Prosess

Startes

lik objekter eller *function* med *static* data

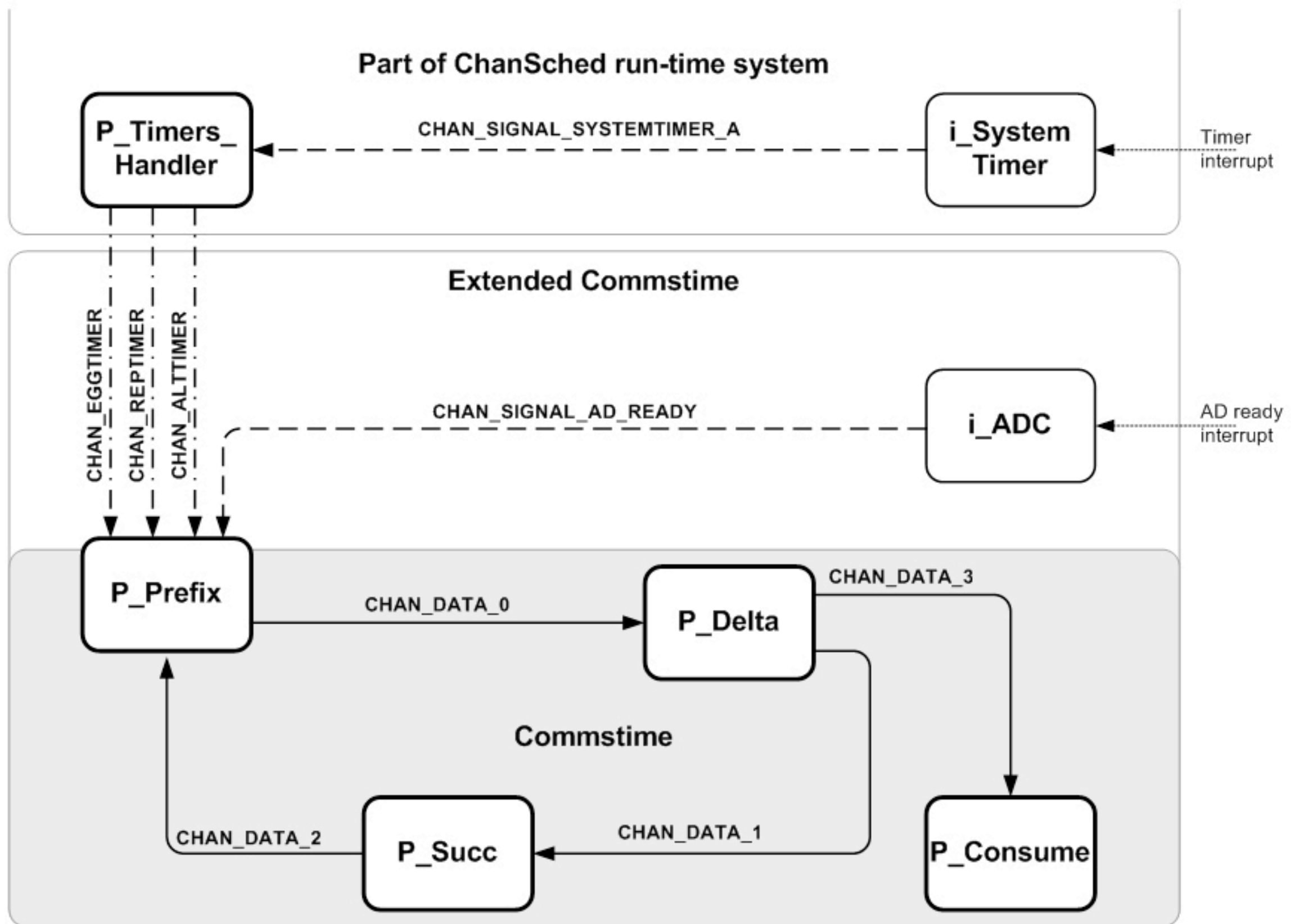
Kjører

av seg sjøl

Er tette

ingen lekkasje

Kan ha egne tidskrav



Meldinger, *ikke* vha

Meldinger, *ikke* vha

funksjonskall

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

pipes

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

pipes

meldinger

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

pipes

asynkrone
meldinger

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

pipes

asynkrone

meldinger

«skjeve» meldingsdiagram

Meldinger, *ikke* vha

Som (alle) kan være ok til sitt bruk

funksjonskall

pekere til fellesdata

direkte bruk av semaforer

pipes

asynkrone

meldinger

«skjeve» meldingsdiagram

Meldinger, vha

Meldinger, vha

synkroner meldinger

Meldinger, vha

synkroner meldinger

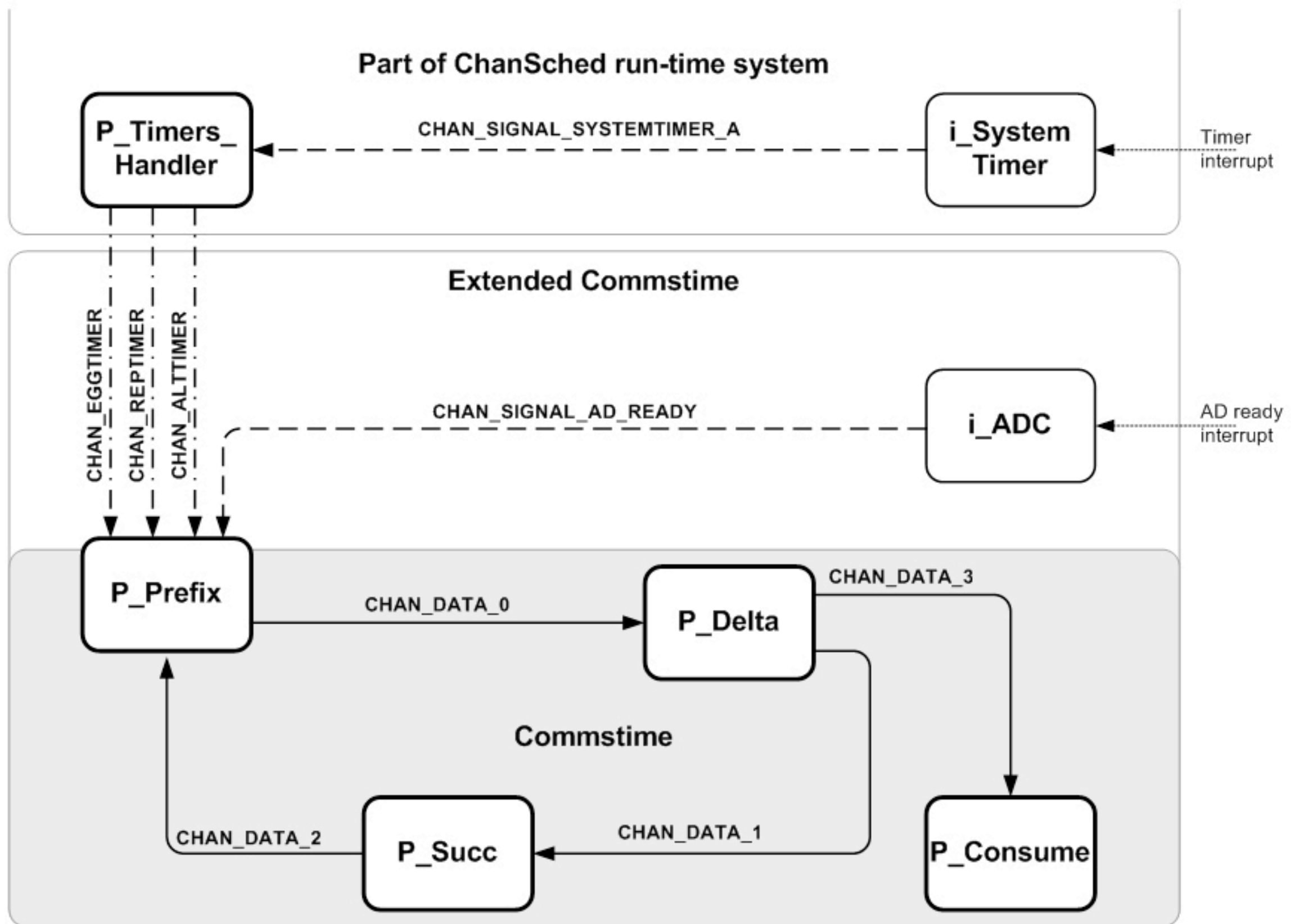
asynkroner «signaler»

Meldinger, vha

synkroner meldinger

asynkroner «signaler»

«rette» meldingsdiagram



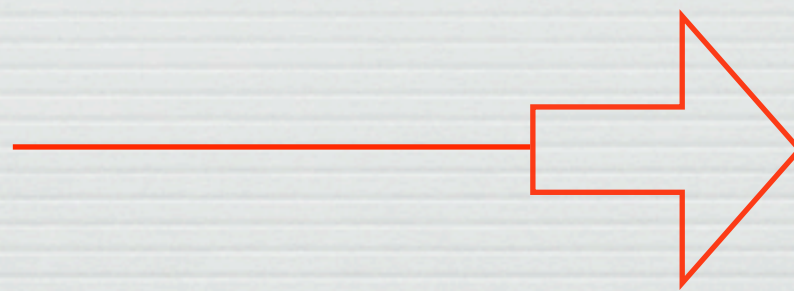
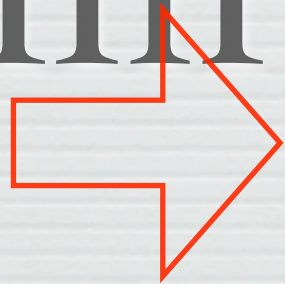
kommunikasjon=

kommunikasjon=



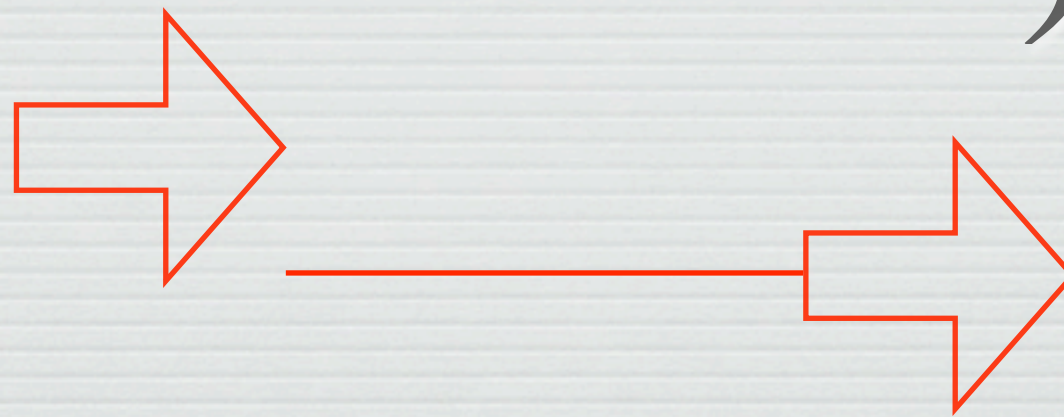
synkronisering

halvferdig
kommunikasjon =



synkronisering

halvferdig
kommunikasjon =



synkronisering
blokkering

CSP

Ada

CSP

Ada

CSP

KroC

Ada
CSP

KRoC
JCSP

Ada
CSP
Transfer,
preter
KRoC
JCSP

Ada
CSP
Transfer,
preter
KRoC
JCS
C+kjøresystem

parallelitet

parallelitet

alt blir enklere

parallelitet

alt blir enklere

WYSIWYG-semantikk

parallelitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

parallelitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

parallelitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeblokker / bibliotek

Klient / tjener

parallelitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklusser / bibliotek

Klient / tjener

parallelitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeblokker / bibliotek

Klient / tjener

Vranglås!

parallelitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeblokker / bibliotek

Klient / tjener

Vranglås!

Sikre design mønstre

parallelitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

Sikre design mønstre

Formell modellering = bevis

parallelitet

rett modell for

(nesten)

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

Sikre design mønstre

Formell modellering = bevis

En formell modell i Promela kan formelt verifiseres i Spin

```
mtype = {M_UP, M_DW};
chan Chan_data_down = [0] of {mtype};
chan Chan_data_up    = [0] of {mtype};
proctype P1 (chan Chan_data_in, Chan_data_out)
{
    do
        :: Chan_data_in  ? M_UP -> skip;
        :: Chan_data_out ! M_DW -> skip;
    od;
}
proctype P2 (chan Chan_data_in, Chan_data_out)
{
    do
        :: Chan_data_in  ? M_DW -> skip;
        :: Chan_data_out ! M_UP -> skip;
    od;
};
init
{
    atomic
    {
        run P1 (Chan_data_down, Chan_data_up);
        run P2 (Chan_data_up,    Chan_data_down);
    }
}
```

<http://en.wikipedia.org/wiki/Promela#Executability>

Promela / Spin


```

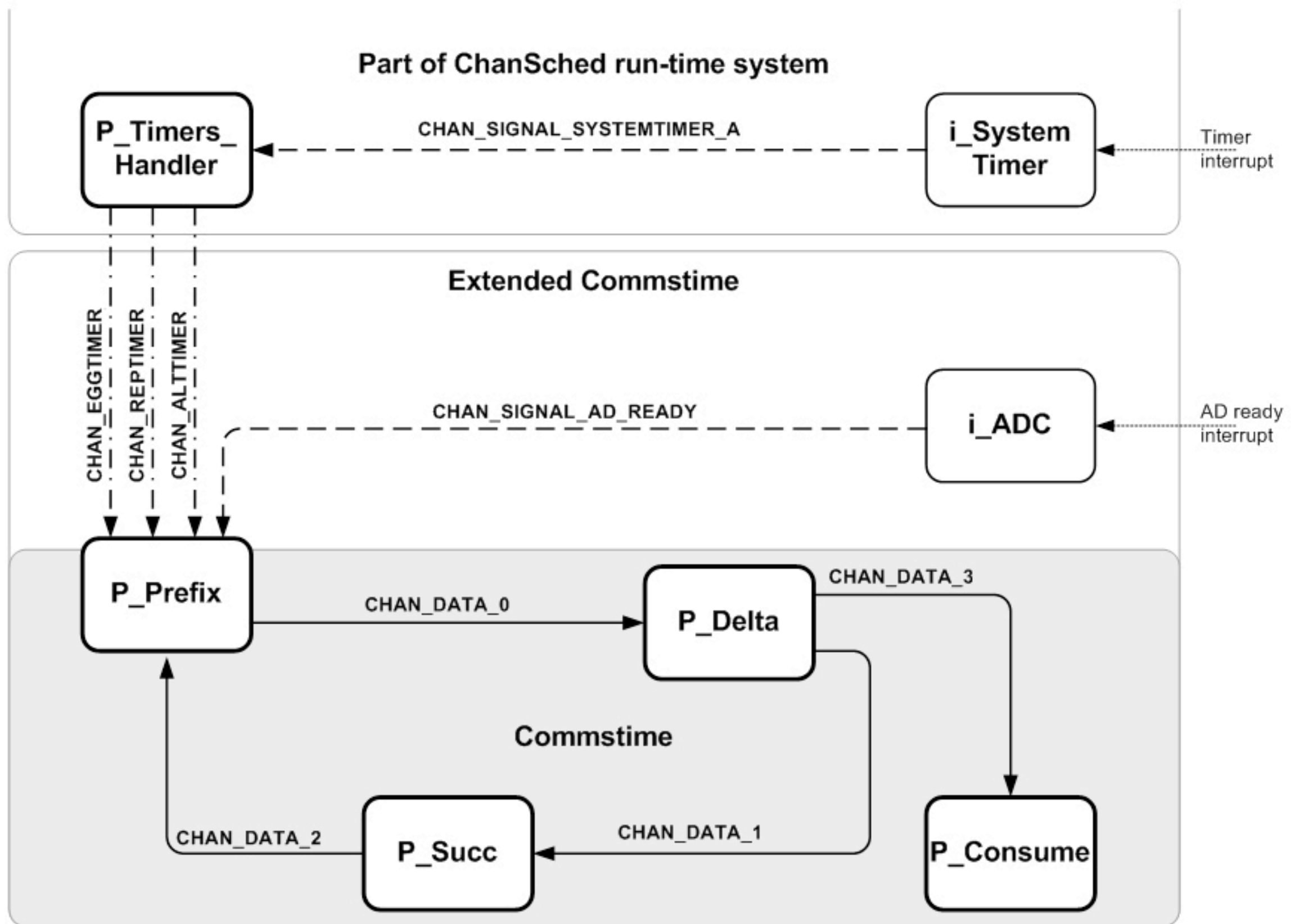
mtype = {M_UP, M_DW};
chan Chan_data_down = [0] of {mtype};
chan Chan_data_up    = [0] of {mtype};
proctype P1 (chan Chan_data_in, Chan_data_out)
{
    do
        :: Chan_data_in  ? M_UP -> skip;
        :: Chan_data_out ! M_DW -> skip;
    od;
};
proctype P2 (chan Chan_data_in, Chan_data_out)
{
    do
        :: Chan_data_in  ? M_DW -> skip;
        :: Chan_data_out ! M_UP -> skip;
    od;
};
init
{
    atomic
    {
        run P1 (Chan_data_down, Chan_data_up);
        run P2 (Chan_data_up,    Chan_data_down);
    }
}

```

<http://en.wikipedia.org/wiki/Promela#Executability>

Promela / Spin

C+kjøresystem
«ChanSched»




```

01 Void P_Prefix (void)                // extended "Prefix"
02 {
03     Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get proc Context from Scheduler
04     PROCTOR_PREFIX()                 // jump table (see Section 2)
05     ... some initialisation
06     SET_EGGTIMER (CHAN_EGGTIMER, CP->LED_Timeout_Tick);
07     SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
08     CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof(CP->Data_0)); //first output
09     while (TRUE)
10     {
11         ALT();                        // this is the needed "PRI_ALT"
12         ALT_EGGREPTIMER_IN (CHAN_EGGTIMER);
13         ALT_EGGREPTIMER_IN (CHAN_REPTIMER);
14         gALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
15         ALT_CHAN_IN (CHAN_DATA_2, &CP->Data_2, sizeof(CP->Data_2));
16         ALT_ALTTIMER_IN (CHAN_ALTTIMER, TIME_TICKS_100_MSECS);
17         gALT_END();
18         switch (g_ThisChannelId)
19         {
20             ... process the guard that has been taken, e.g. CHAN_DATA_2
21             CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof (CP->Data_0));
22         };
23     }
24 }

```




Kanal i Amsterdam, nov. 09



//

Ada
CSP
Transfer-
preter
KRoC
JCS
C+kjøresystem



//

Go

Ada

KR

C

JCSP

Transfer-
preter

C+kjøresystem



Google

Go (programming language)

From Wikipedia, the free encyclopedia

Go is a compiled, concurrent programming language. It is being developed by Google^[2], with initial design by Robert Griesemer, Rob Pike and Ken Thompson.

References

- [↑] http://golang.org/doc/go_faq.html#Implementation
- [↑] Google-go-language (<http://www.techcrunch.com/2009/11/10/google-go-language/>)

External links

- Go Programming Language Homepage (<http://golang.org/>)
- Go language FAQ (http://golang.org/doc/go_lang_faq.html)
- The Go Programming Language (<http://www.youtube.com/watch?v=rKnDgT73v8s>) at YouTube (requires Adobe Flash)
- IRC : #go-nuts (<irc://irc.freenode.net/go-nuts>) on irc.freenode.net
- Mailing list : <http://groups.google.com/group/golang-nuts>

Retrieved from "[http://en.wikipedia.org/wiki/Go_\(programming_language\)](http://en.wikipedia.org/wiki/Go_(programming_language))"

Categories: Computer language stubs | C++ programming language family | Curly bracket programming languages | Concurrent programming languages | Google

- This page was last modified on 11 November 2009 at 08:59.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details.
- Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Go	
Paradigm	compiled, concurrent
Appeared in	2009
Designed by	Robert Griesemer, Rob Pike and Ken Thompson
Major implementations	Gc is in C using yacc/bison for the parser and Gccgo has C++ front-end with a recursive descent parser coupled to the standard GCC back end ^[1] .
OS	Linux, Mac OS X
License	BSD
Website	http://golang.org

[http://en.wikipedia.org/wiki/Go_\(programming_language\)](http://en.wikipedia.org/wiki/Go_(programming_language))

Go (programming language)

From Wikipedia, the free encyclopedia

Go is a compiled, concurrent programming language. It is being developed by Google^[2], with initial design by Robert Griesemer, Rob Pike and Ken Thompson.

References

- [^] http://golang.org/doc/go_faq.html#Implementation
- [^] Google-go-language (<http://www.techcrunch.com/2009/11/10/google-go-language/>)

External links

- Go Programming Language Homepage (<http://golang.org/>)
- Go language FAQ (http://golang.org/doc/go_faq.html)
- The Go Programming Language (<http://www.youtube.com/watch?v=rKn0g173v8s>) at YouTube (requires Adobe Flash)
- IRC : #go-nuts (<irc://irc.freenode.net/go-nuts>) on irc.freenode.net
- Mailing list : <http://groups.google.com/group/golang-nuts>

Retrieved from "[http://en.wikipedia.org/wiki/Go_\(programming_language\)](http://en.wikipedia.org/wiki/Go_(programming_language))"

Categories: Computer language stubs | C++ programming language family | Curly bracket programming languages | Concurrent programming languages | Google

- This page was last modified on 11 November 2009 at 08:59.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details.

Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Paradigm	compiled, concurrent
Appeared in	2009
Designed by	Robert Griesemer, Rob Pike and Ken Thompson
Major implementations	Gc is in C using yacc/bison for the parser and Gccgo has C++ front-end with a recursive descent parser coupled to the standard GCC back end ^[1] .
OS	Linux, Mac OS X
License	BSD
Website	http://golang.org

[http://en.wikipedia.org/wiki/Go_\(programming_language\)](http://en.wikipedia.org/wiki/Go_(programming_language))

Why build concurrency on the ideas of CSP?

Concurrency and multi-threaded programming have a reputation for difficulty. We believe the problem is due partly to **complex designs such as pthreads** and partly to **overemphasis on low-level details such as mutexes**, condition variables, and even memory barriers. Higher-level interfaces enable much simpler code, even if there are still mutexes and such under the covers.

One of the most successful models for providing high-level linguistic support for concurrency comes from **Hoare's Communicating Sequential Processes**, or **CSP**. **Occam and Erlang are two well known languages that stem from CSP**. Go's concurrency primitives derive from a different part of the family tree whose main contribution is the powerful notion of channels as first class objects.

http://golang.org/doc/go_lang_faq.html#csp

Skal vi gå og prøve Go?

Fra sekvensielt til parallelt

Fra sekvensielt til parallelt og multikjerne

Fra sekvensielt til parallelt og multikjerne

Øyvind Teig

<http://www.teigfam.net/oyvind/>

A red speech bubble with a tail pointing towards the name Øyvind Teig.

takk
for meg!