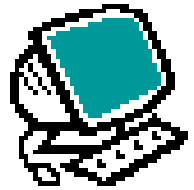
SDL

Peter Marwedel
TU Dortmund,
Informatik 12

2009/10/08

SDL

Used here as a (prominent) example of a model of computation based on **asynchronous message passing communication**.



☞ appropriate also for distributed systems

Communication/ Computation	**Shared memory**	**Message passing**	
		blocking	**Non-blocking**
FSM	StateCharts		SDL

SDL

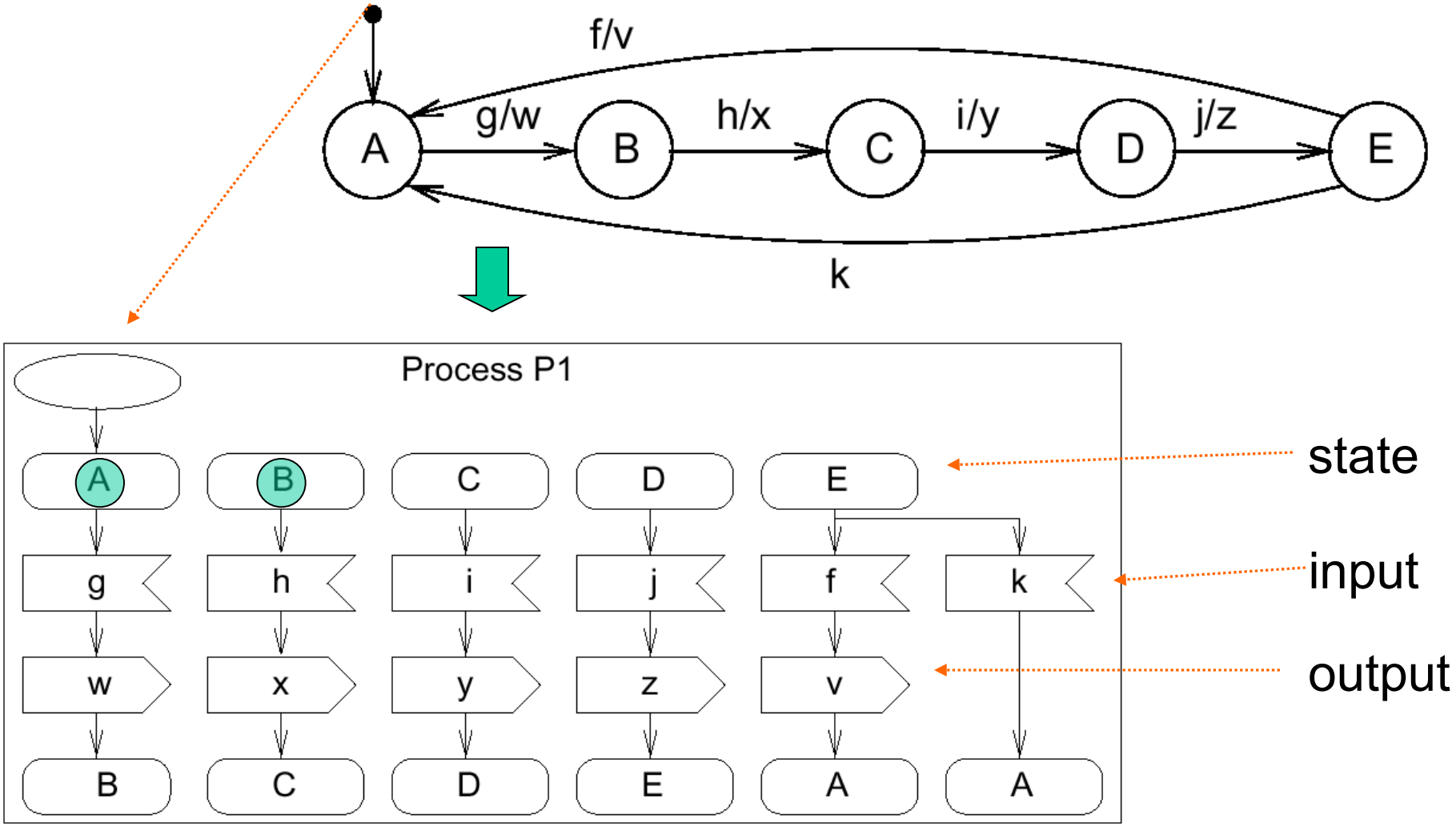
Language designed for specification of distributed systems.

- Dates back to early 70s,
- Formal semantics defined in the late 80s,
- Defined by ITU (International Telecommunication Union): Z.100 recommendation in 1980
  Updates in 1984, 1988, 1992, 1996 and 1999

SDL

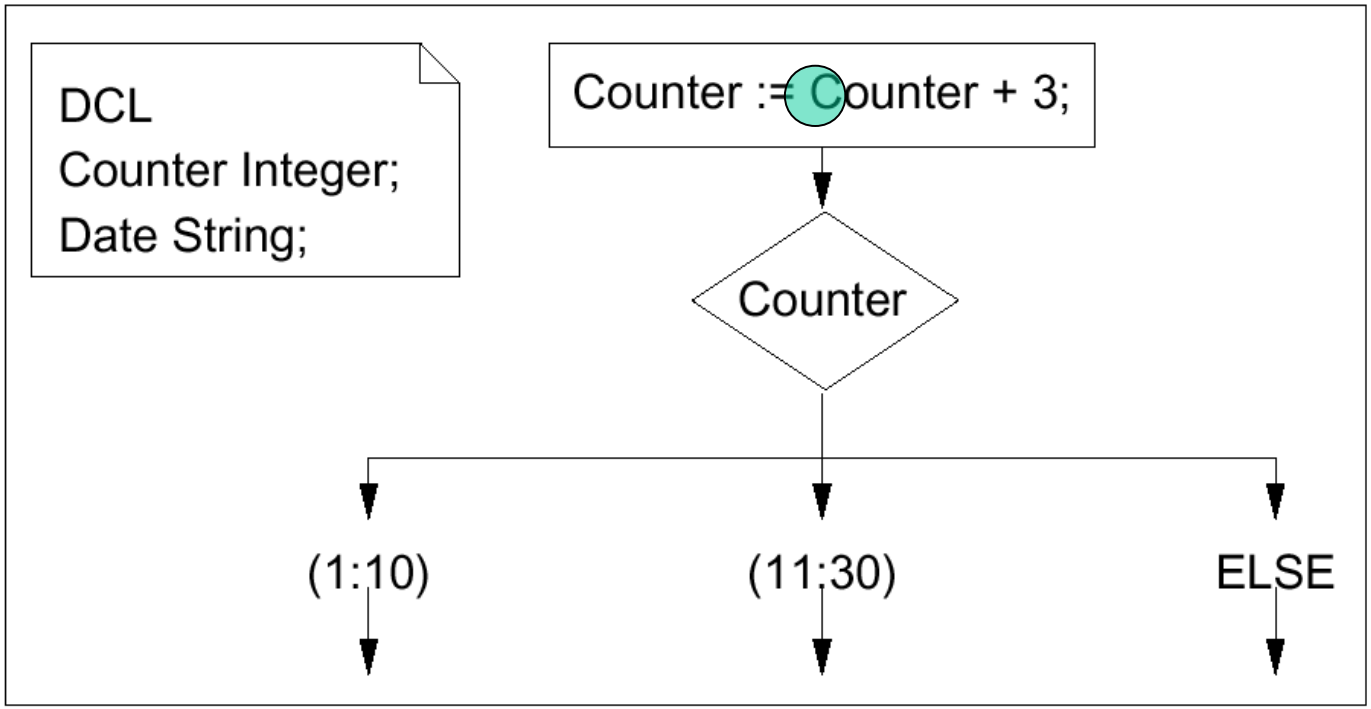
- Provides textual and graphical formats to please all users,
- Just like StateCharts, it is based on the CFSM model of computation; each FSM is called a **process**,
- However, it uses message passing instead of shared memory for communications,
- SDL supports operations on data.

SDL-representation of FSMs/processes



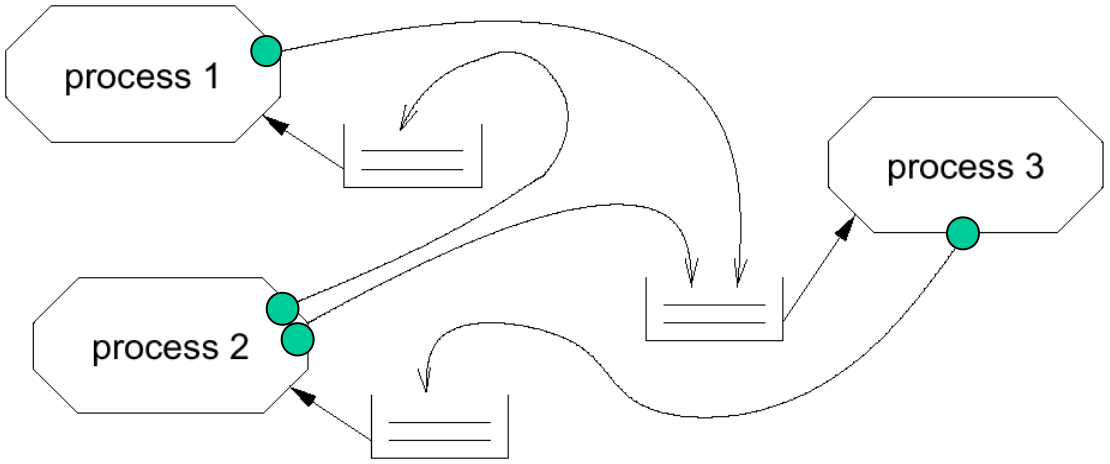
Operations on data

Variables can be declared locally for processes.
Their type can be predefined or defined in SDL itself.
SDL supports abstract data types (ADTs). Examples:



Communication among SDL-FSMs

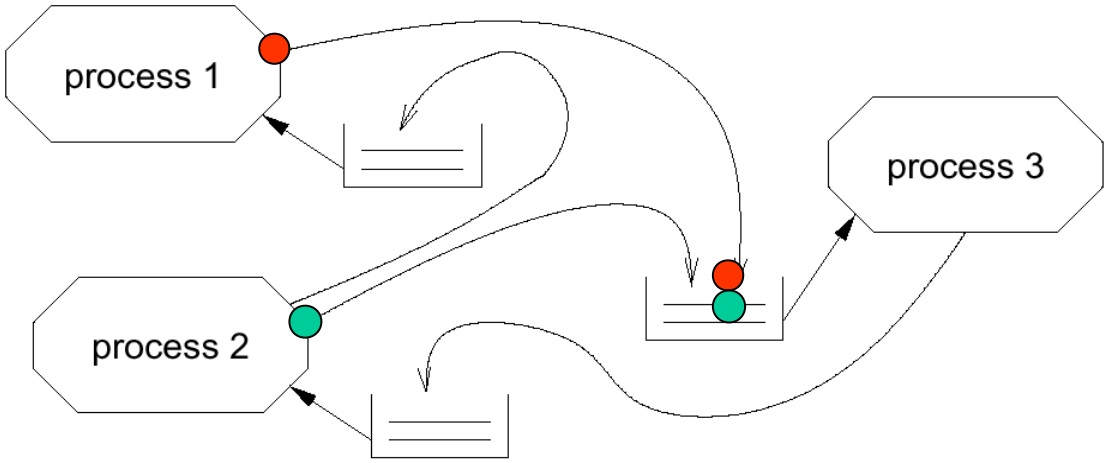
Communication between FSMs (or „processes“) is based on **message-passing**, assuming a **potentially indefinitely large FIFO-queue**.



- Each process fetches next entry from FIFO,
- checks if input enables transition,
- if yes: transition takes place,
- if no: input is ignored (exception: SAVE-mechanism).

Determinate?

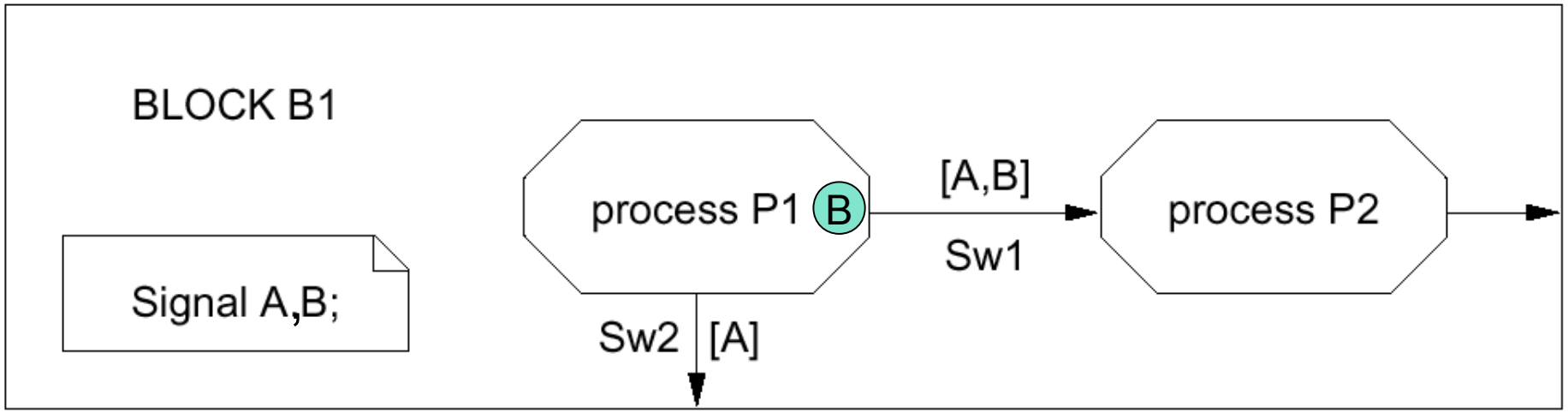
Let tokens be arriving at FIFO at the same time:
☞Order in which they are stored, is unknown:



All orders are legal: ☞simulators can show different behaviors for the same input, all of which are correct.

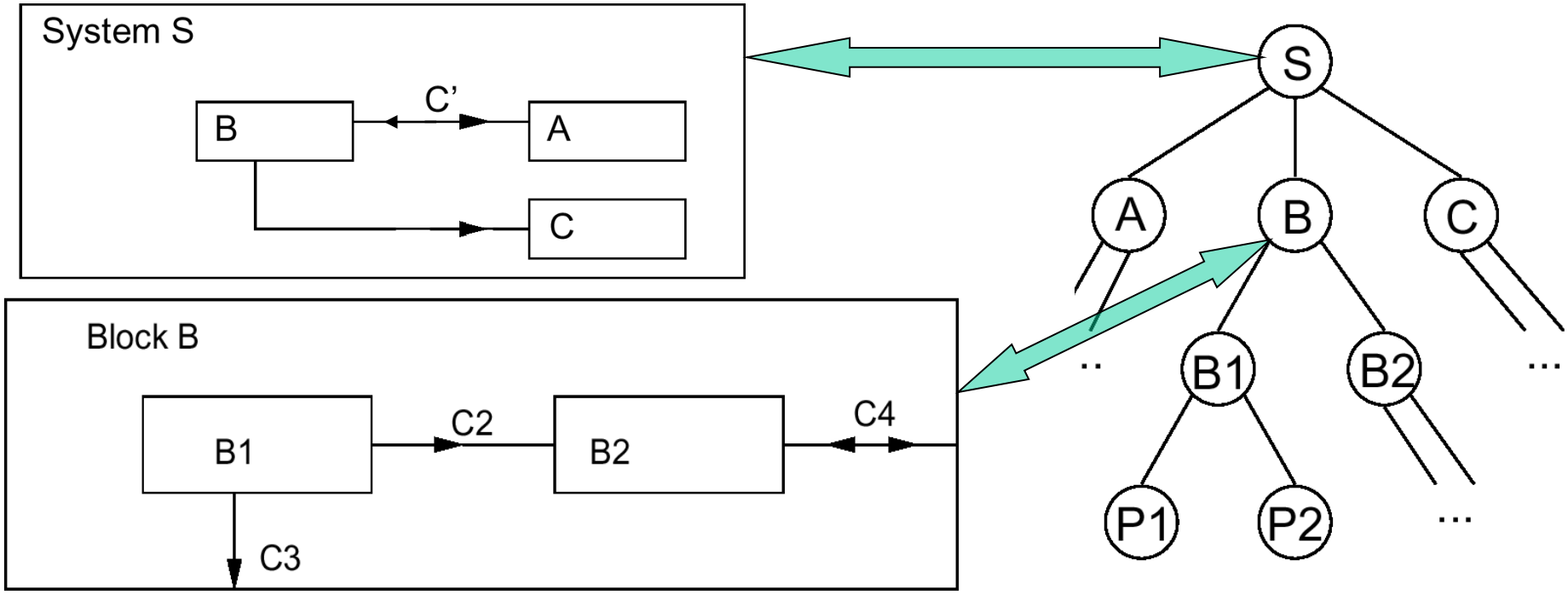
Process interaction diagrams

Interaction between processes can be described in process interaction diagrams (special case of block diagrams).
In addition to processes, these diagrams contain channels and declarations of local signals.
Example:



Hierarchy in SDL

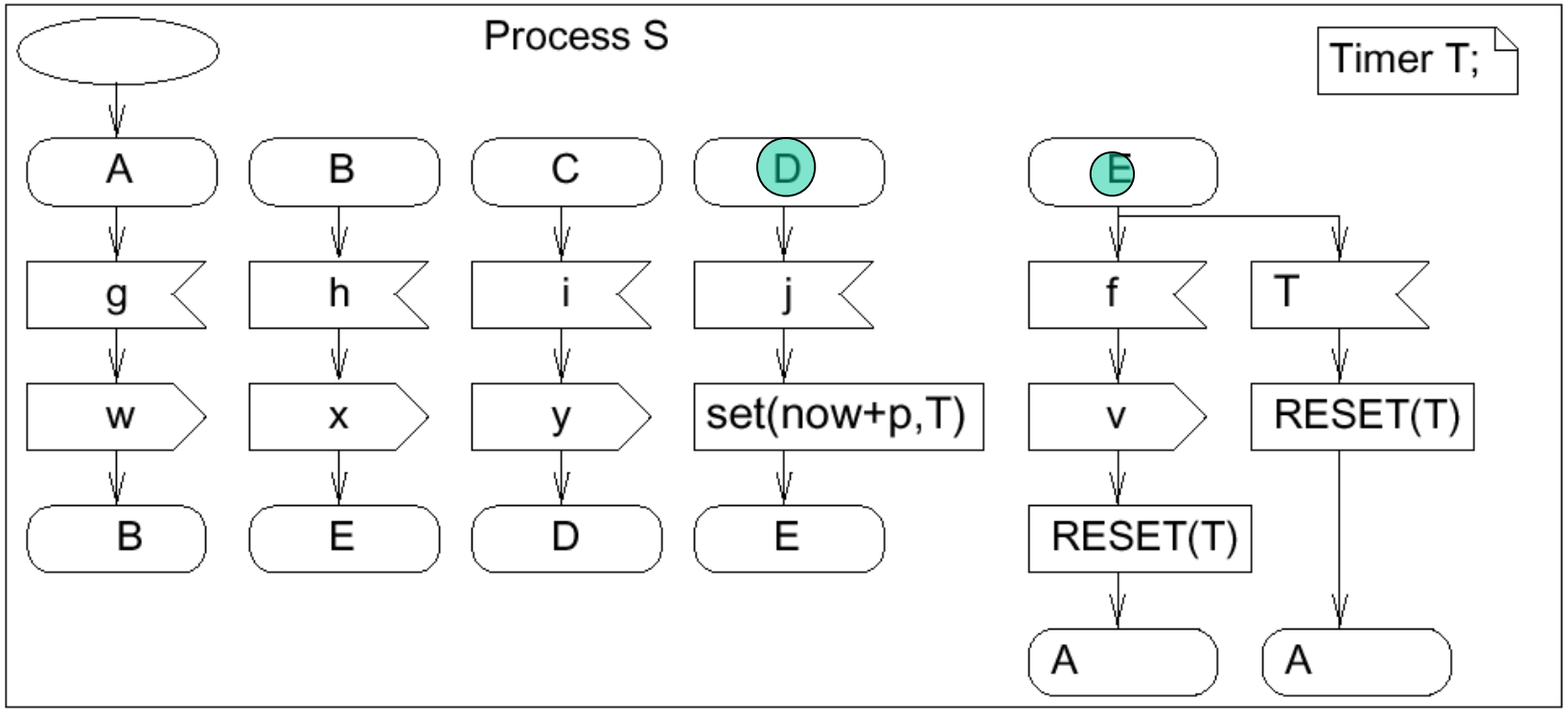
Process interaction diagrams can be included in **blocks.**
The root block is called **system.**



Processes cannot contain other processes, unlike in StateCharts.

Timers

Timers can be declared locally. Elapsed timers put signal into queue (not necessarily processed immediately). RESET removes timer (also from FIFO-queue).

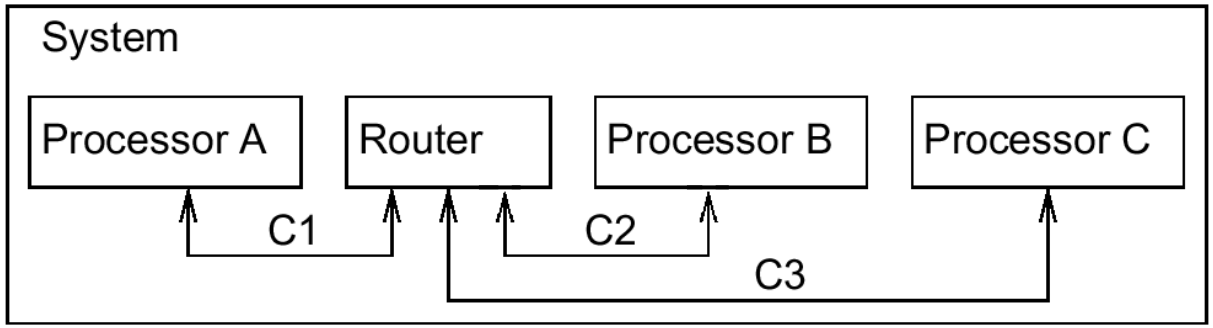


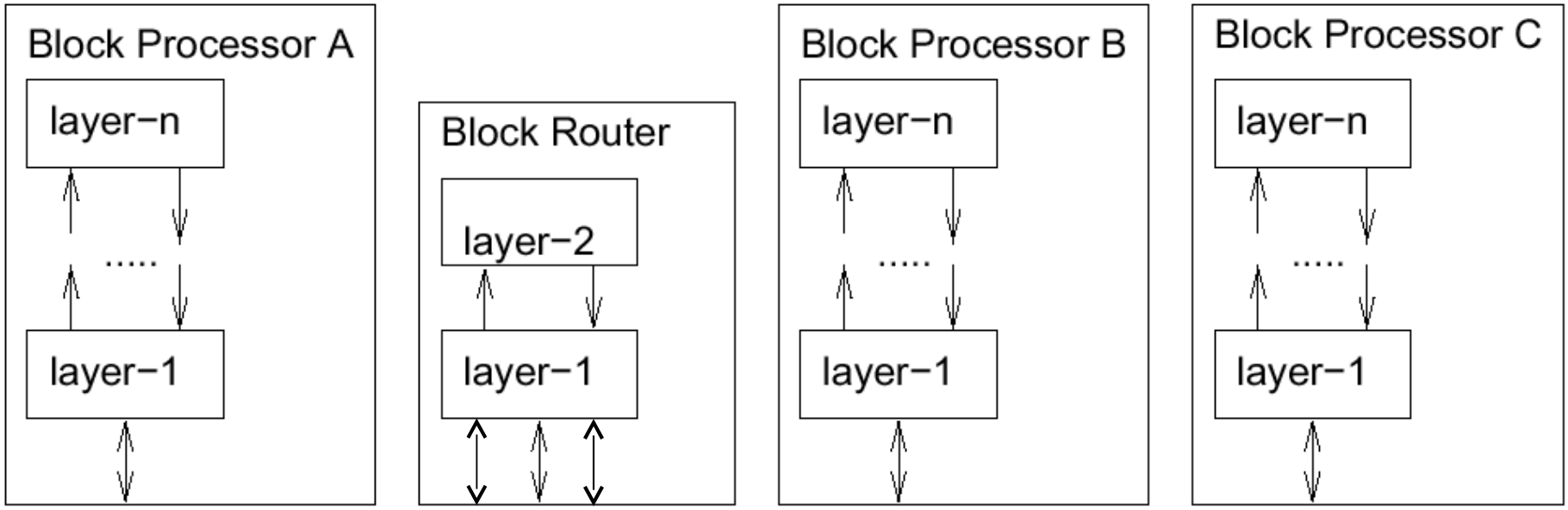
Additional language elements

SDL includes a number of additional language elements, like

- procedures
- creation and termination of processes
- advanced description of data
- More features added for SDL-2000 (not well accepted)

Application: description of network protocols



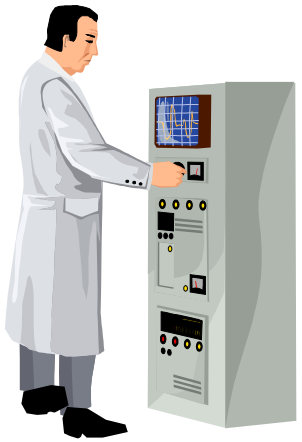


Larger example: vending machine

Machine° selling pretzels, (potato) chips, cookies, and doughnuts:

accepts nickels, dime, quarters, and half-dollar coins.

Not a distributed application.



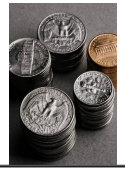
° [J.M. Bergé, O. Levia, J. Roullard: High-Level System Modeling, Kluwer Academic Publishers, 1995]

System VendingMachine

Creject
[reject_coin]

Ccoins
[nickel,dime,
quarter,half]

CoinInterface



Cadd
[add]

Ccointctrl
[rej_further_coins,
accept_coins]

DecodeRequests

CamontDisplay
[amount_entered]

Cemptydisplay
[pretzel_empty, chip_empty,
cookie_empty, doughnut_empty]

CspitPurchased
[spit_pretzel, spit_chip,
spit_cookie, spit_doughnut]

Crequest
[pur_pretzel, pur_chip, pur_cookie, pur_doughnut,
reload_pretzel, reload_chip, reload_cookie,
reload_doughnut]

[spit_change],Cchange

ChangeInterface

CexaktDisplay
[exact_only]

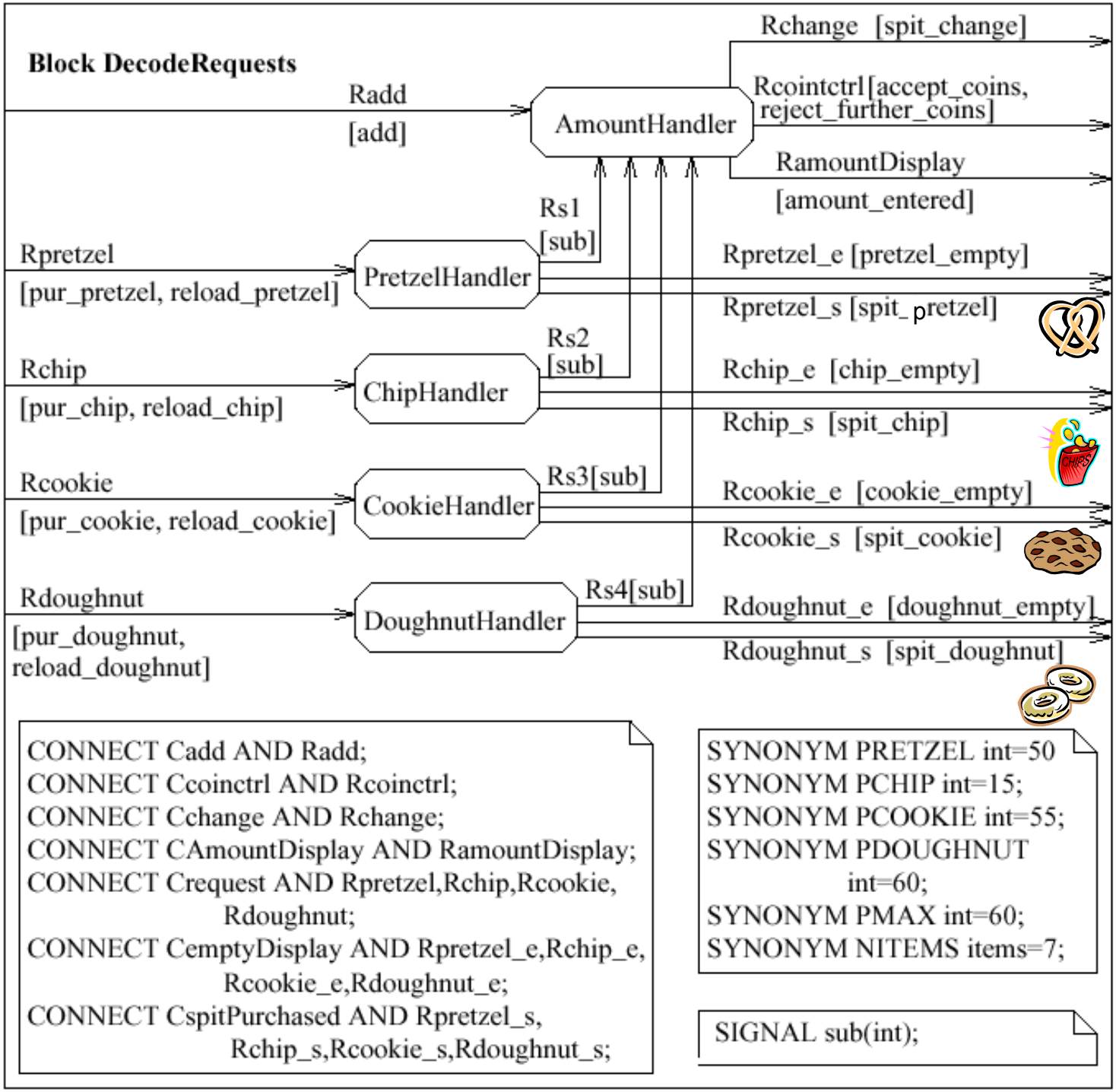
CspitChange
[spit_nickel,
spit_dime]

SIGNAL
[dime, nickel, quarter, half, pur_pretzel,
pur_cookie, pur_doughnut, pur_chip,
add(int), spit_change(int),
amount_entered(int), reject_further_coins,
exact_only, accept_coins, reject_coins,
spit_dime, spit_nickel, pretzel_empty,
spit_pretzel,chip_empty, spit_chip,
cookie_empty, spit_cookie, doughnut_empty,
spit_doughnut, reload_pretzel, reload_chip,
reload_cookie, reload_doughnut]

```
SYNTYPE items=INTEGER
  CONSTANTS 0:7
ENDSYNTYPE items;
```

```
SYNTYPE int=INTEGER
  CONSTANTS 0:127
ENDSYNTYPE int;
```

Decode Requests



ChipHandler

Process ChipHandler

DCL nchip items:=NITEMS;

VIEWED current int;

pur_wait

pur_chip

VIEW(current) >= PCHIP

no

pur_wait

yes

sub(PCHIP)

nchip:= nchip-1;

spit_chip

nchip=0

yes

chip_empty

empty

reload_chip

nchip:=NITEMS

pur_wait

no

pur_wait

Evaluation

- Excellent for distributed applications (used for ISDN),
- Commercial tools available from SINTEF, Telelogic, Cinderella (//www.cinderella.dk).
- Not necessarily determinate
  (order, in which FSMs are reading input is unknown)
  ☞ no synchronous language,
- Implementation requires bound for the maximum length of FIFOs; may be very difficult to compute,
- Timer concept adequate just for soft deadlines,
- Limited way of using hierarchies,
- Limited programming language support,
- No description of non-functional properties,
- Becoming less popular

Summary

SDL

- MoC: finite state machine components
  + non-blocking message passing communication
- Representation of processes
- Communication & block diagrams
- Timers and other language elements
- Example: Vending machine
- Evaluation