Data flow models

Peter Marwedel
TU Dortmund,
Informatik 12

2008/10/10

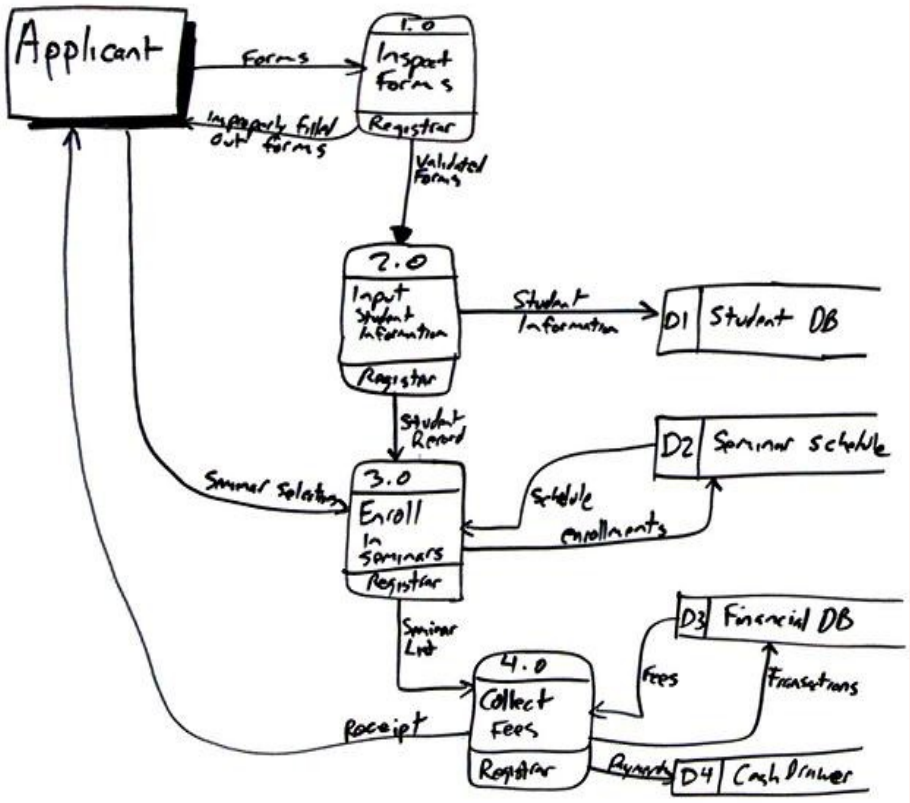
Models of computation considered in this course

Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Communicating finite state machines	StateCharts		SDL
Data flow model ⊂ **Computational graphs**	Not useful	Simulink Sequence dia-gram, Petri nets	Kahn process networks, SDF
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA |	
Discrete event (DE) model	VHDL, …	Only experimental systems, e.g. distributed DE in Ptolemy	

Data flow as a “natural” model of applications

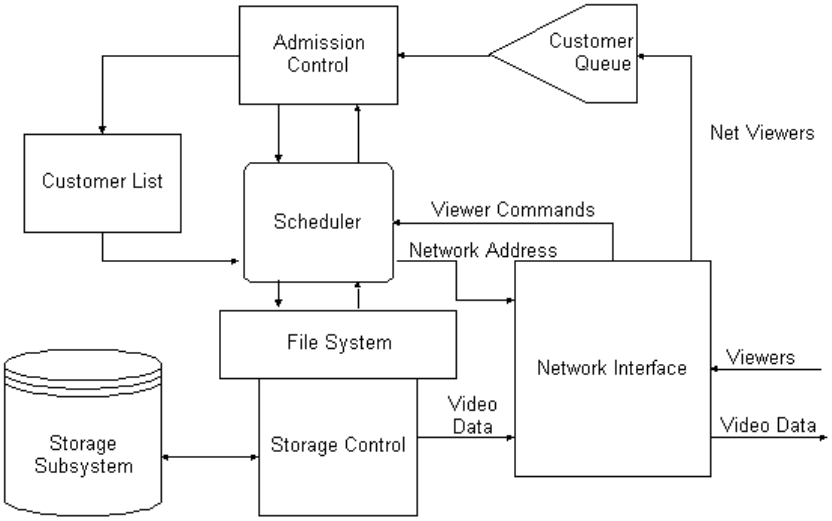
Examples

Registering for courses



http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm

Video on demand system



www.ece.ubc.ca/~irenek/techpaps/vod/vod.html

Data flow modeling

Def.: The process of identifying, modeling and documenting how data moves around an information system.

Data flow modeling examines

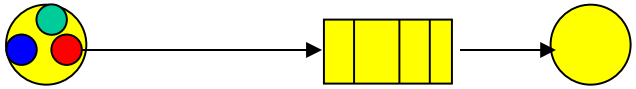
- *processes* (activities that transform data from one form to another),
- *data stores* (the holding areas for data),
- *external entities* (what sends data into a system or receives data from a system, and
- *data flows* (routes by which data can flow).

http://www.webopedia.com/TERM/D/data_flow_modeling.html

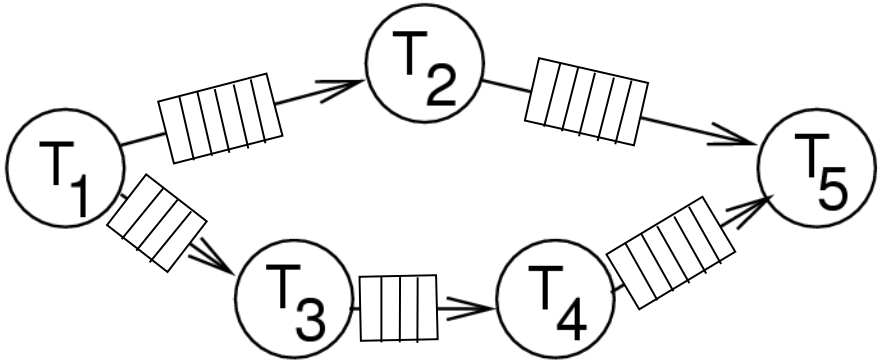
See also S. Edwards: http://www.cs.columbia.edu/~sedwards/classes/2001/w4995-02/presentations/dataflow.ppt

Reference model for data flow:
Kahn process networks

For asynchronous message passing:
communication between tasks is buffered



Special case: Kahn process networks:
executable task graphs;
Communication via infinitely large FIFOs



Properties of Kahn process networks (1)

- Each node corresponds to one program/task;
- Communication is only via channels;
- Channels include FIFOs as large as needed;
- Channels transmit information within an unpredictable but finite amount of time;
- Mapping from ≥1 input seq. to ≥1 output sequence;
- In general, execution times are unknown;
- Send operations are non-blocking, reads are blocking.
- One producer and one consumer;
  i.e. there is only one sender per channel;

Properties of Kahn process networks (2)

- There is only one sender per channel.
- A process cannot check whether data is available before attempting a read.
- A process cannot wait for data for more than one port at a time.
- Therefore, the order of reads depends only on data, not on the arrival time.
- Therefore, Kahn process networks are deterministic (!); for a given input, the result will always the same, regardless of the speed of the nodes. SDL-like conflicts at FIFOs do not exist.

This is the key beauty of KPNs!

Properties of Kahn process networks (3)

- Model of parallel computations used in practice (@NXP)
- Example:

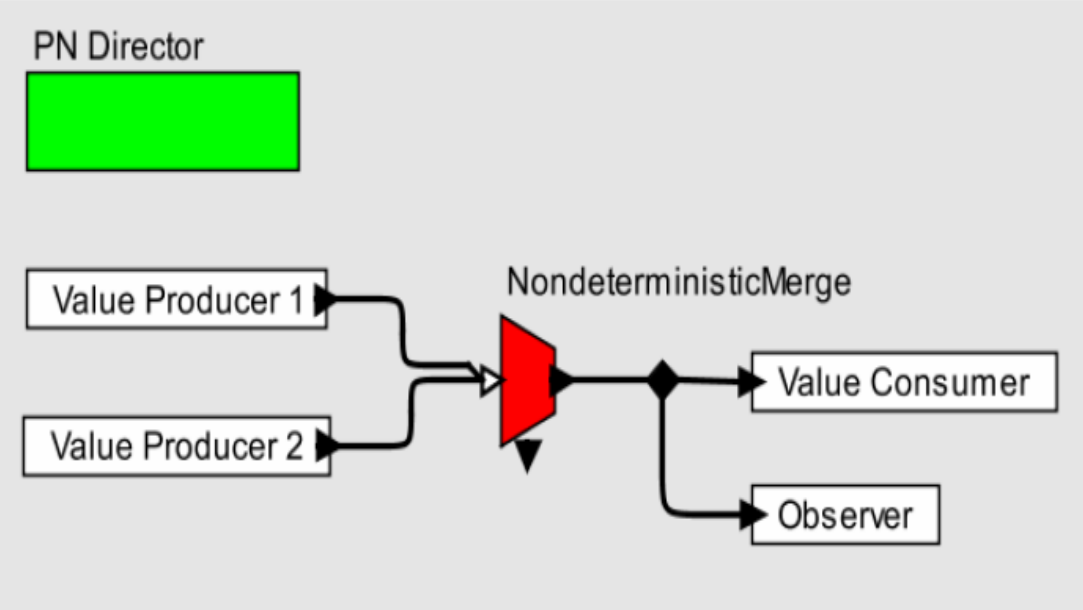
```
Process f(in int u, in int v, out int w){
  int i; bool b = true;
  for (;;) {
   i= b ? wait(u) : wait(v); //wait return next token in FIFO, blocks if empty
   printf ("%i\n",i);
   send (i,w); //writes a token into a FIFO w/o blocking
   b = !b;
   }
```


- It is a challenge to schedule KPNs without accumulating tokens

☞ http://en.wikipedia.org/wiki/Kahn_process_networks

☞ http://ls12-www.cs.tu-dortmund.de/edu/ES/leviKPN.zip: Animation

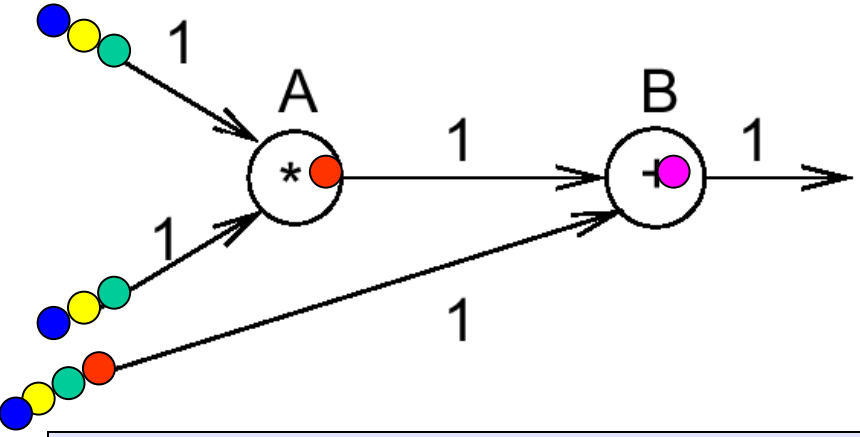
Observer Pattern using Process Networks [Kahn 1974] Extended with Nondeterministic Merge

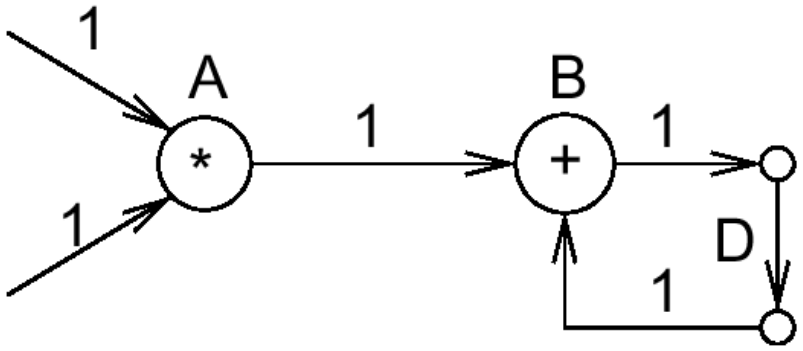


Each actor is a process, communication is via streams, and the NondeterministicMerge explicitly merges streams nondeterministically.

Asynchronous message passing: Synchronous data flow (SDF)

Asynchronous message passing=
tasks do not have to wait until output is accepted.
Synchronous data flow =
all tokens are consumed at the same time.





SDF model allows static scheduling of token production and consumption.
In the general case, buffers may be needed at edges.

Synchronous Dataflow (SDF)

Fixed Production/Consumption Rates

Balance equations (one for each channel):

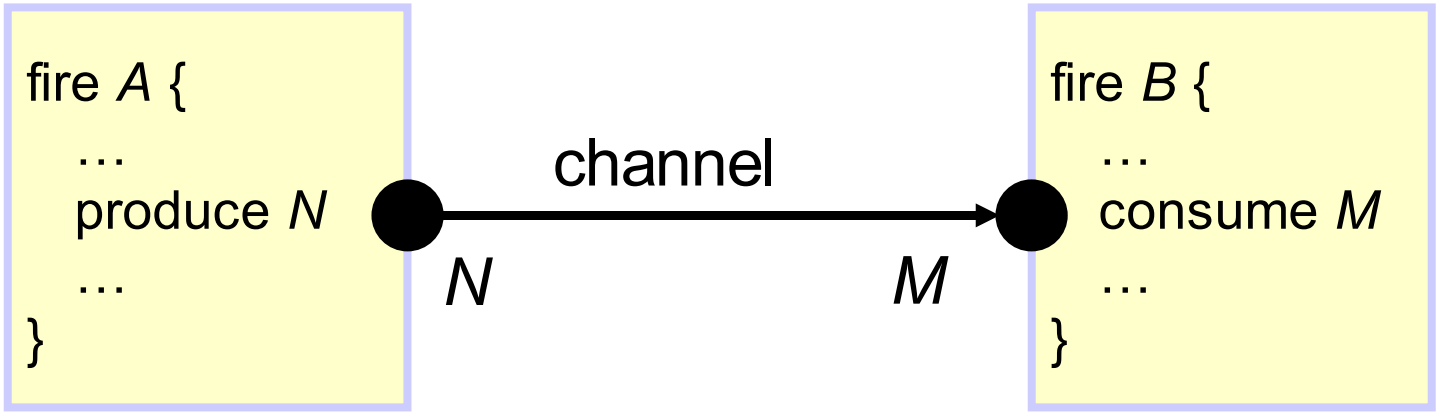
$$f_A N = f_B M$$

- f_A: number of tokens consumed
- N: number of firings per "iteration"
- f_B: number of tokens produced

Schedulable statically

Decidable:

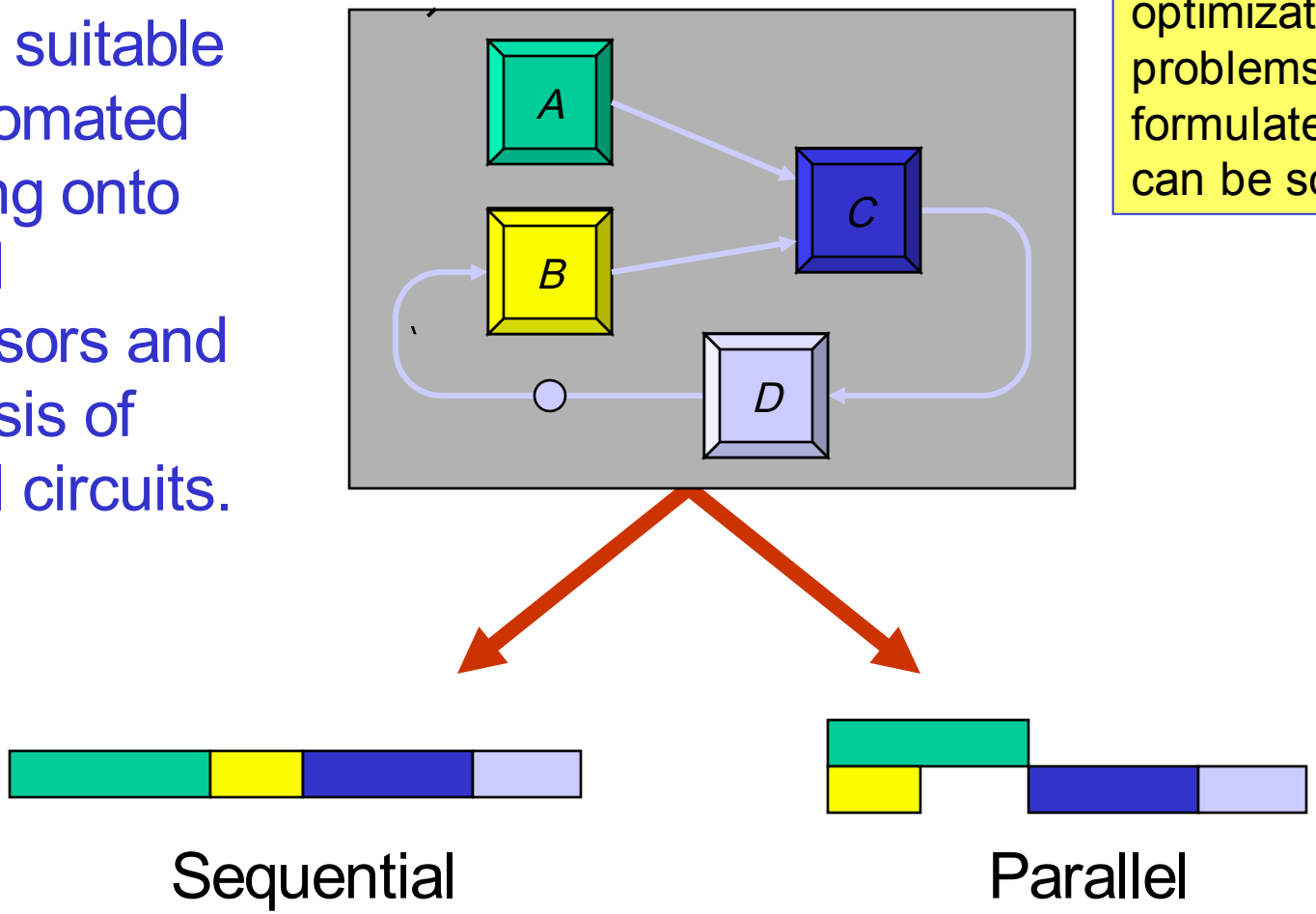
- buffer memory requirements
- deadlock



Source: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt

Parallel Scheduling of SDF Models

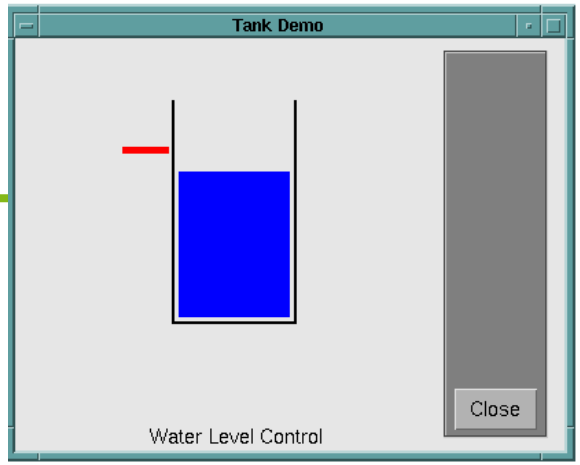
SDF is suitable for automated mapping onto parallel processors and synthesis of parallel circuits.



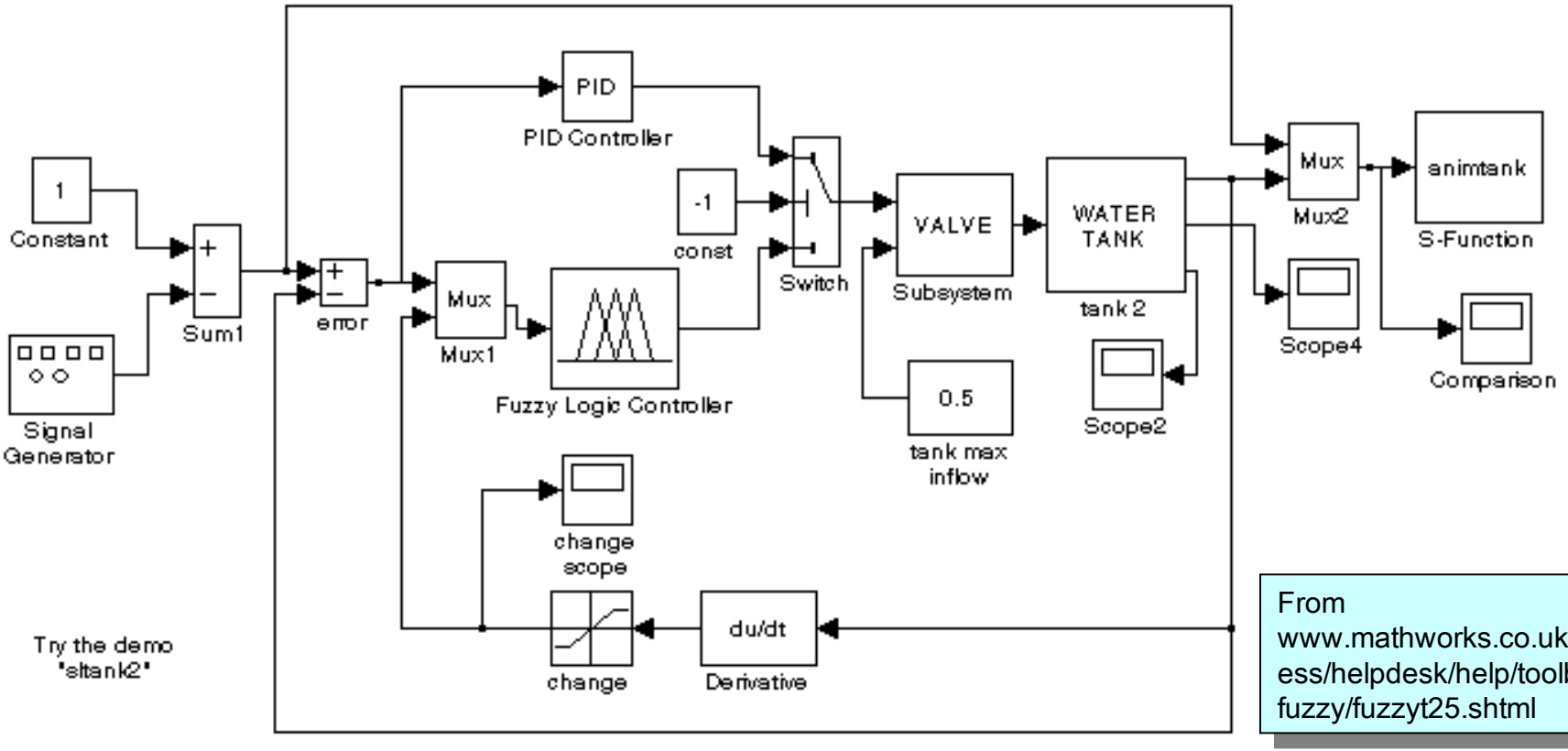
Many scheduling optimization problems can be formulated. Some can be solved, too!

Source: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt

Similar MoC: Simulink
- example -

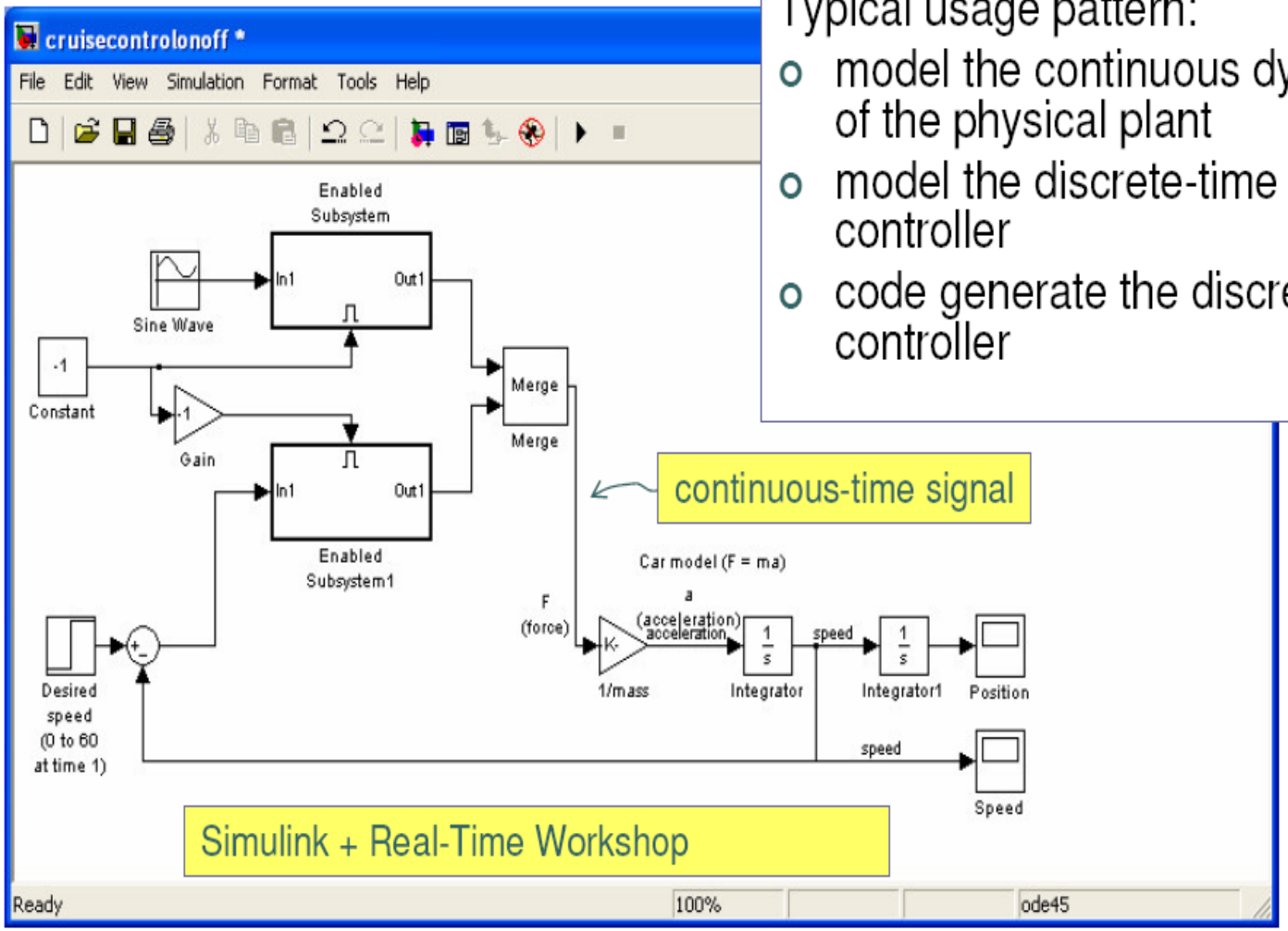


Semantics? *"Simulink uses an idealized timing model for block execution and communication. Both happen infinitely fast at exact points in simulated time. Thereafter, simulated time is advanced by exact time steps. All values on edges are constant in between time steps."* [Nicolae Marian, Yue Ma]



From www.mathworks.co.uk/access/helpdesk/help/toolbox/fuzzy/fuzzyt25.shtml

Threads are Not the Only Possibility: 6th example: Continuous-Time Languages

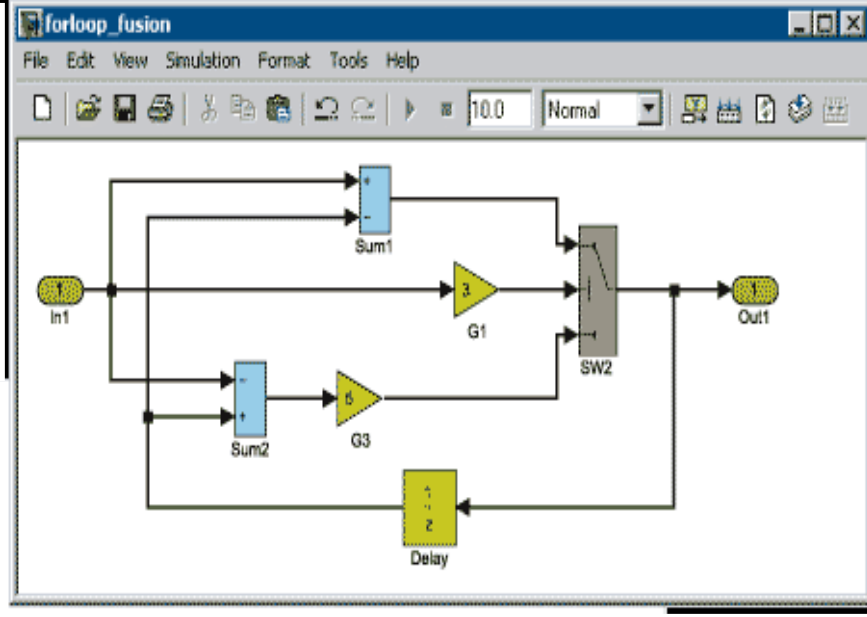


Typical usage pattern:

- model the continuous dynamics of the physical plant
- model the discrete-time controller
- code generate the discrete-time controller

Starting point for “model-based design”

Code automatically generated

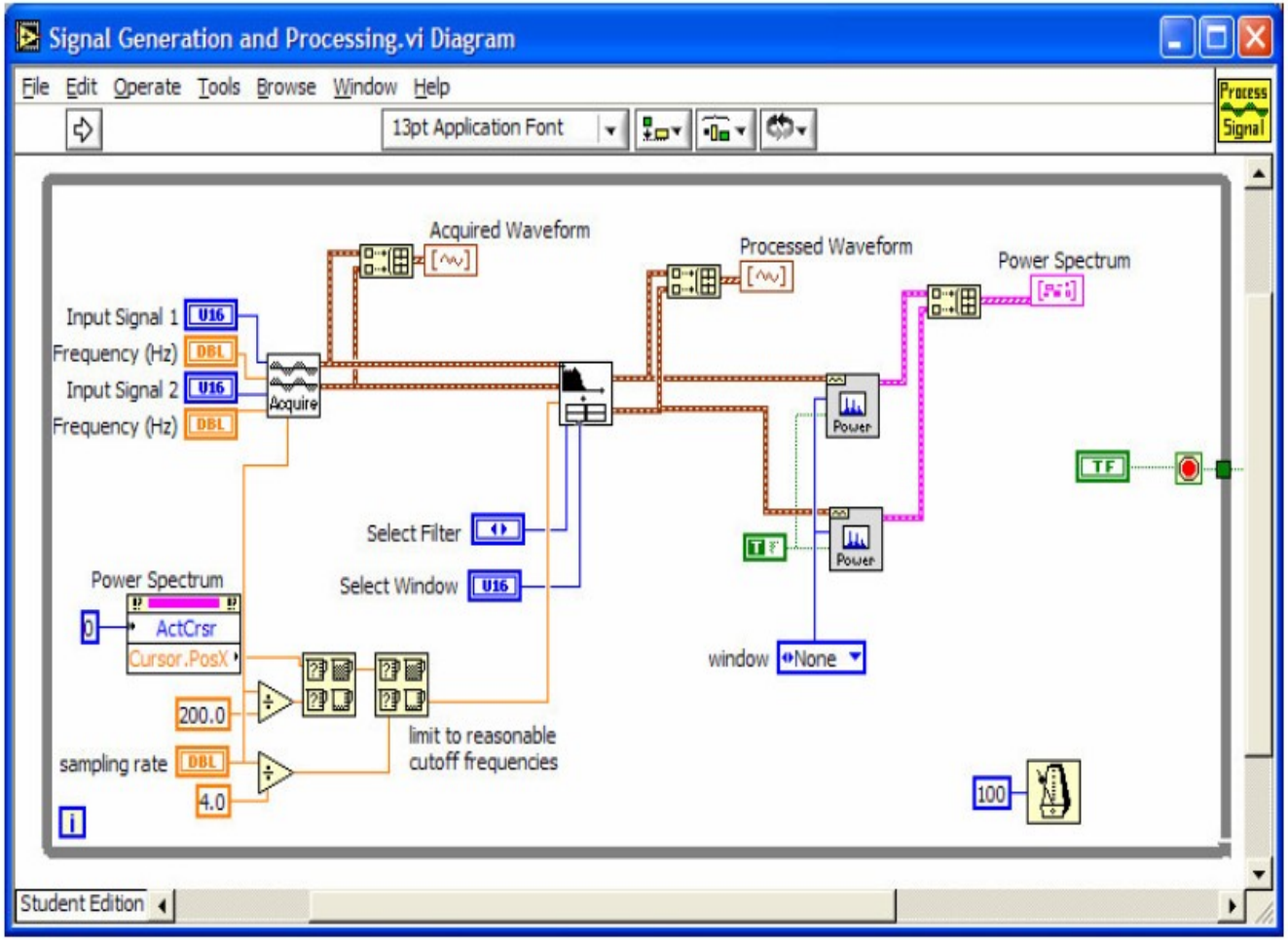



```
 *   Gain: '<Root>/G1'
 *   Sum: '<Root>/Sum2'
 *   Gain: '<Root>/G3'
 */
for(i1=0; i1<10; i1++) {
  if(rtU.In1[i1] * 3.0 >= 0.0) {
    rtb_SW2_c[i1] = rtU.In1[i1] - rtDWork.Delay_DSTATE[i1];
  } else {
    rtb_SW2_c[i1] = (rtDWork.Delay_DSTATE[i1] - rtU.In1[i1]) * 5.0;
  }

  /* Outport: '<Root>/Out1' */
  rtY.Out1[i1] = rtb_SW2_c[i1];

  /* Update for UnitDelay: '<Root>/Delay' */
  rtDWork.Delay_DSTATE[i1] = rtb_SW2_c[i1];
}
```

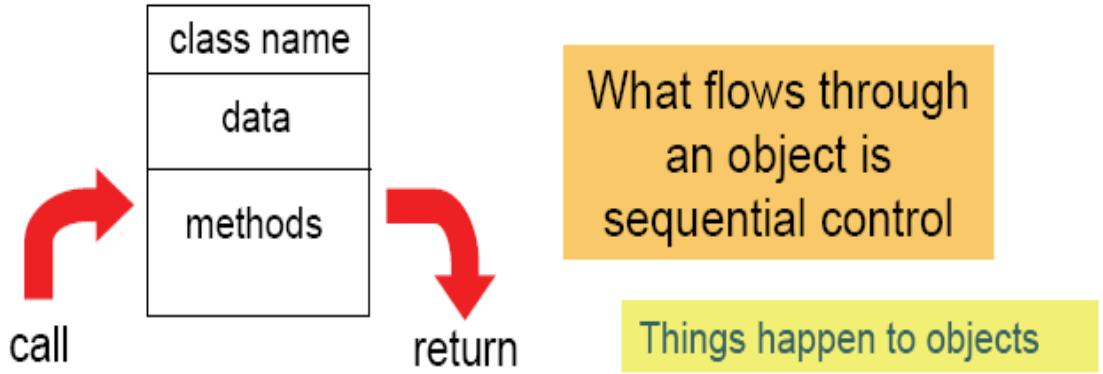
Threads are Not the Only Possibility: 5th example: Instrumentation Languages



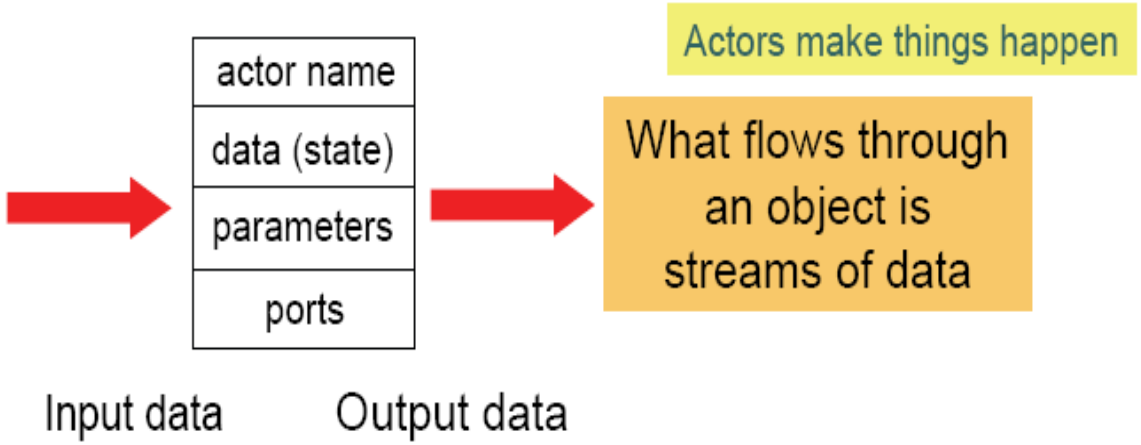
e.g. LabVIEW, Structured dataflow model of computation

Actor languages

The established: Object-oriented:



The alternative: Actor oriented:



© E. Lee, Berkeley

Summary

Data flow model of computation

- Motivation
- Kahn process networks
- SDF
- Visual programming languages
  - Simulink