technische universität
dortmund

fakultät für informatik
informatik 12

(More) Computational Graphs

Peter Marwedel
TU Dortmund,
Informatik 12

2008/10/30

Models of computation considered in this course

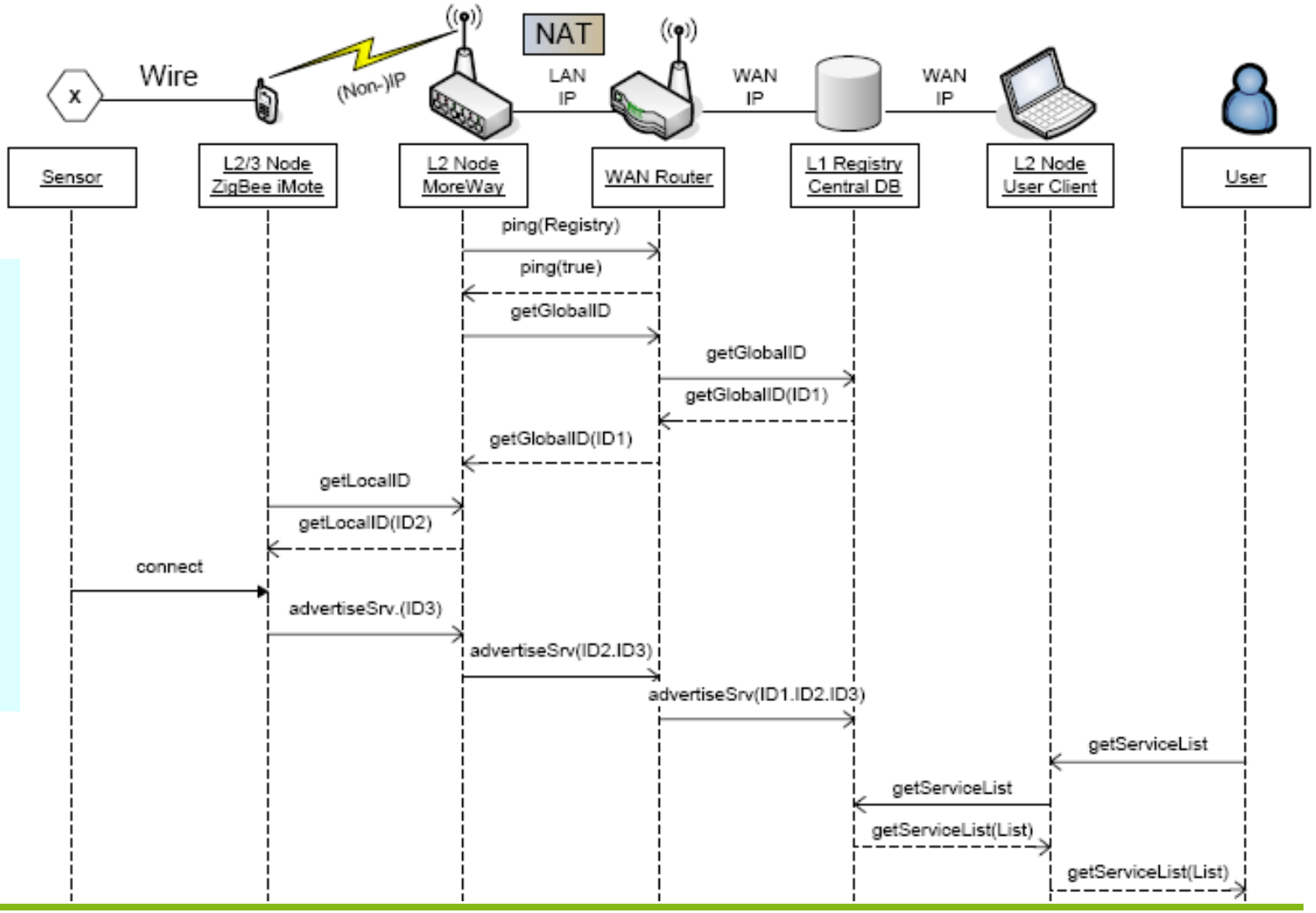
Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Communicating finite state machines	StateCharts		SDL
Data flow model ⊂ **Computational graphs**	Not useful	Simulink Sequence dia-gram, Petri nets	Kahn process networks, SDF
Von Neumann model	C, C++, Java	C, C++,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

From data flow to computational graphs

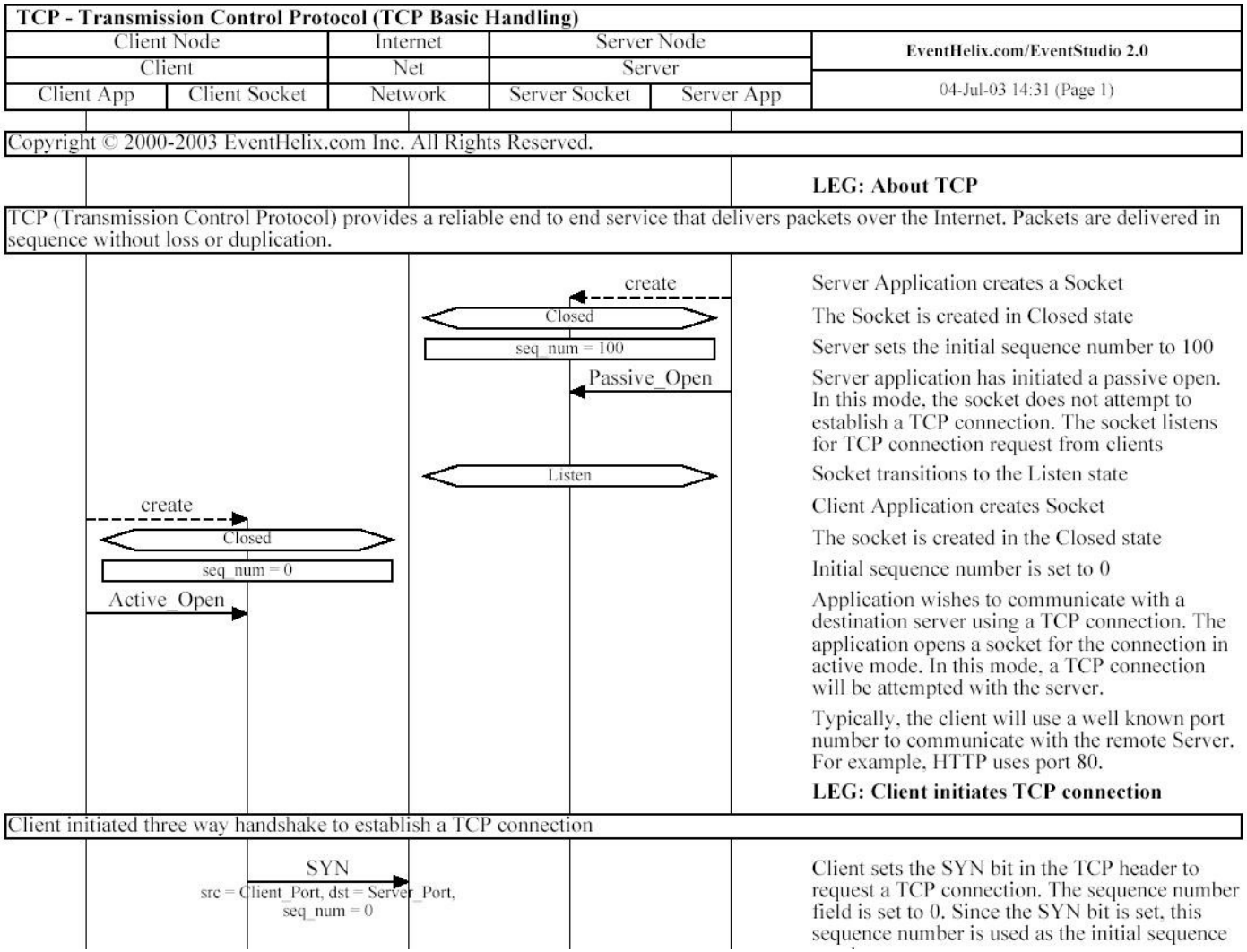
- Pure data flow frequently too restrictive: lack of modeling control, resources etc.
- Modern computers contain subsystems build on the data flow paradigm (scoreboard, Tomasulo's algorithm)
- However, "Lunatics" (S. Edwards) tried to build general purpose data flow computers, but failed
- Data flow graphs are a special case of the somewhat more general computational graphs.
- ☞ Petri nets + MSCs + …

Computational graphs in UML: Sequence Diagram

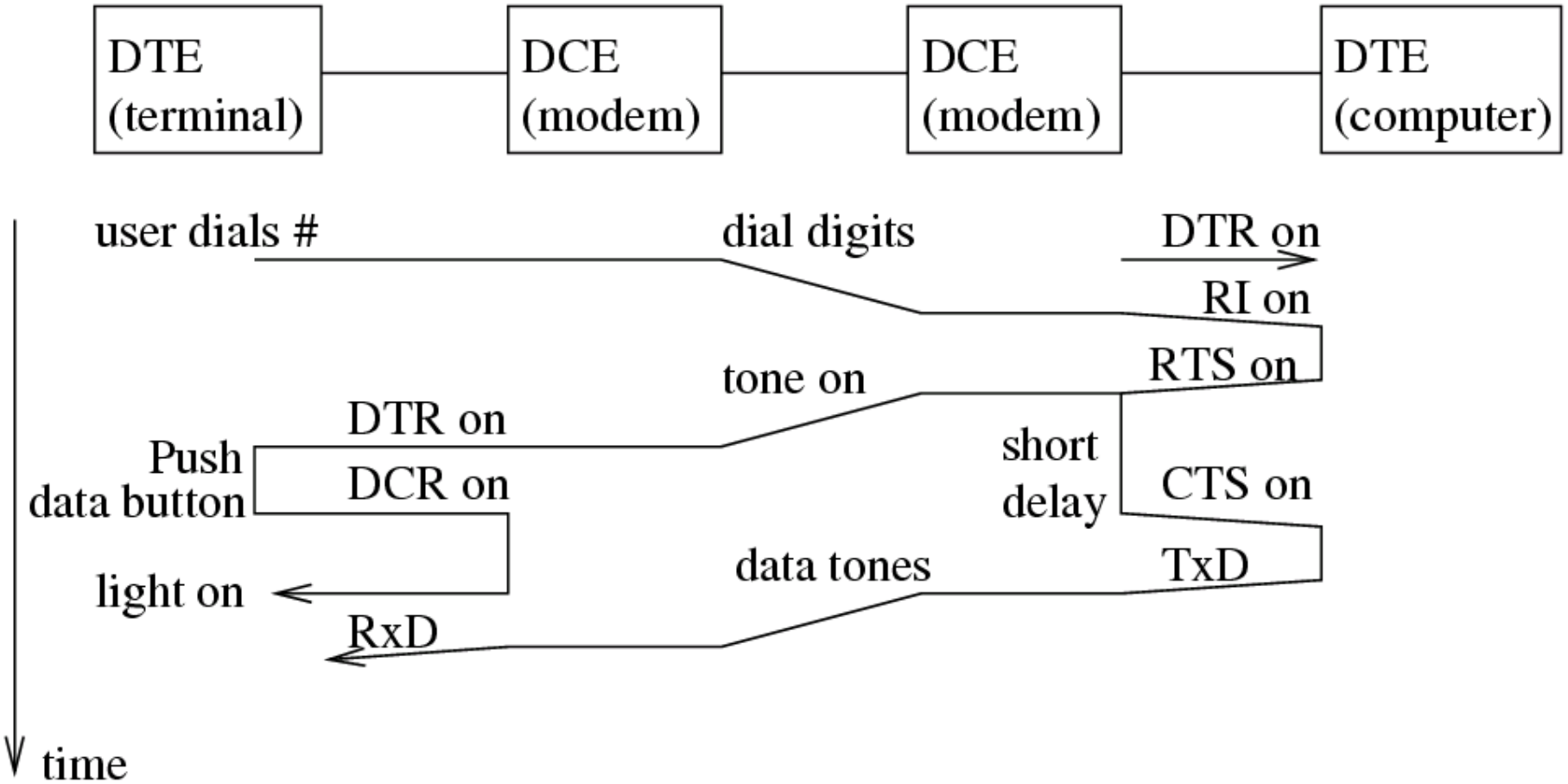
Graphical means for representing sequences; “time” used vertically, geographical distribution horizontally.



A second example: setting up a TCP connection

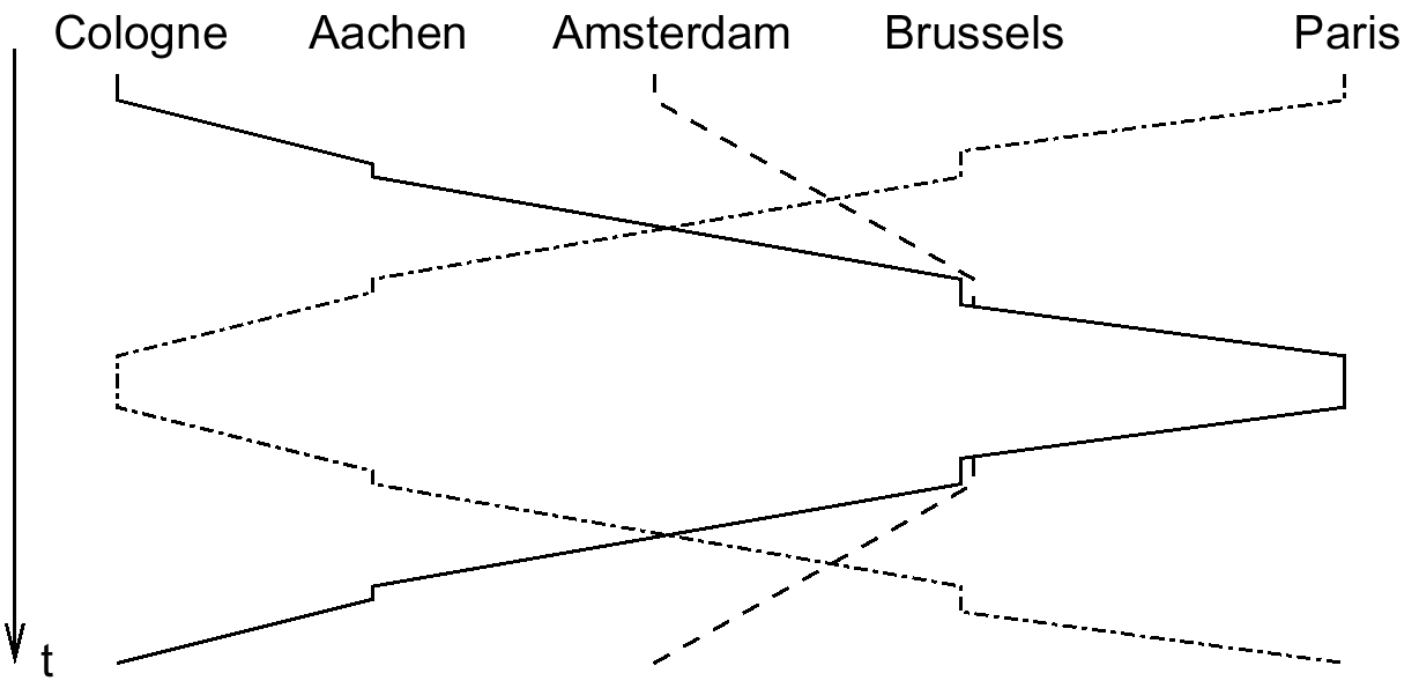


Earlier Example: establishing an (old-fashioned) modem connection



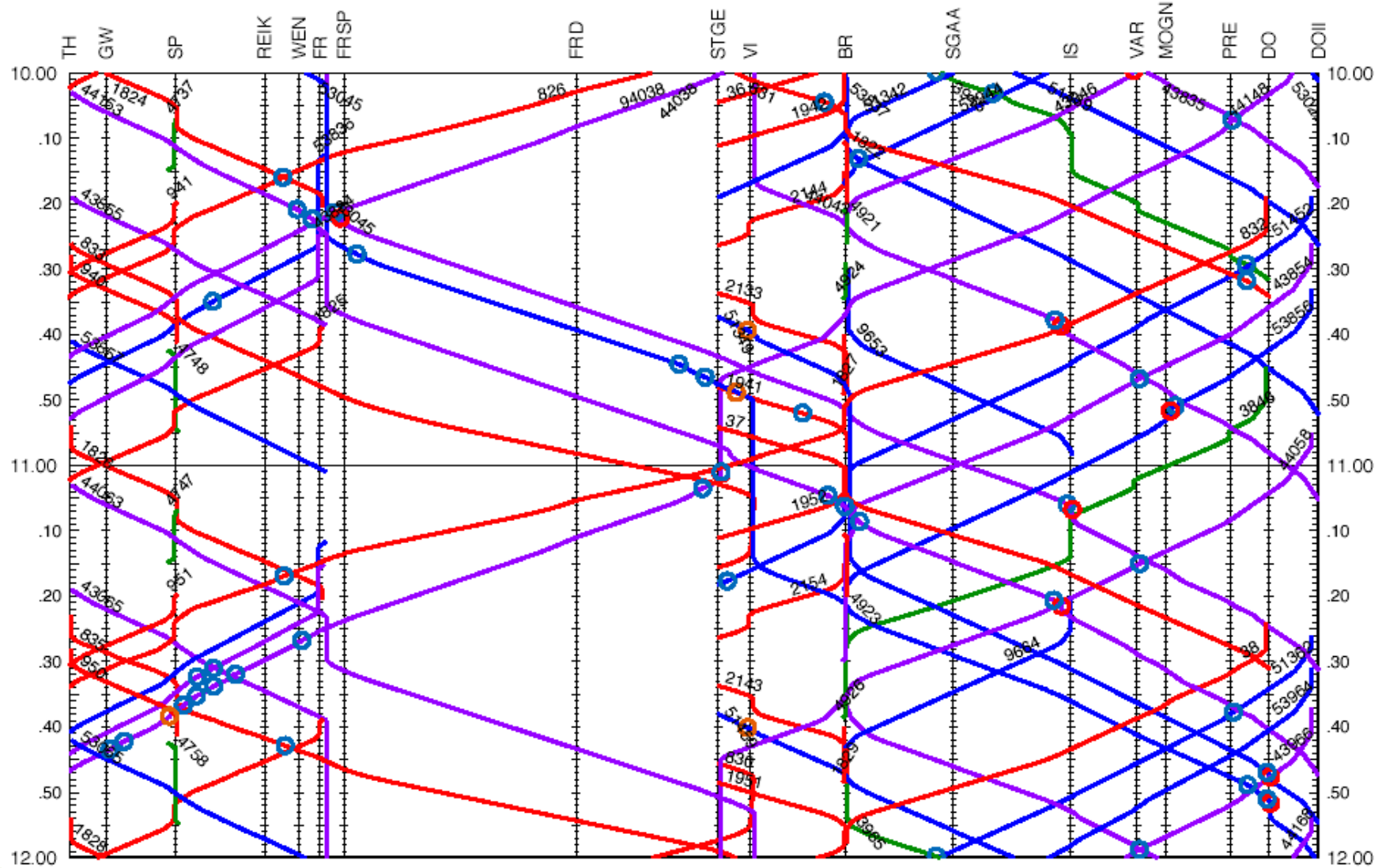
According to Stallings

(Message) Sequence Charts (MSC)



No distinction between accidental overlap and synchronization

Time/distance diagrams as a special case



(Message) Sequence Charts

PROs:

- Appropriate for visualizing schedules,
- Proven method for representing schedules in transportation.
- Standard defined: *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC*), ITU-TS, Geneva, 1996.
- Semantics also defined: *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)—Annex* B: *Algebraic Semantics of Message Sequence Charts*, ITU-TS, Geneva.

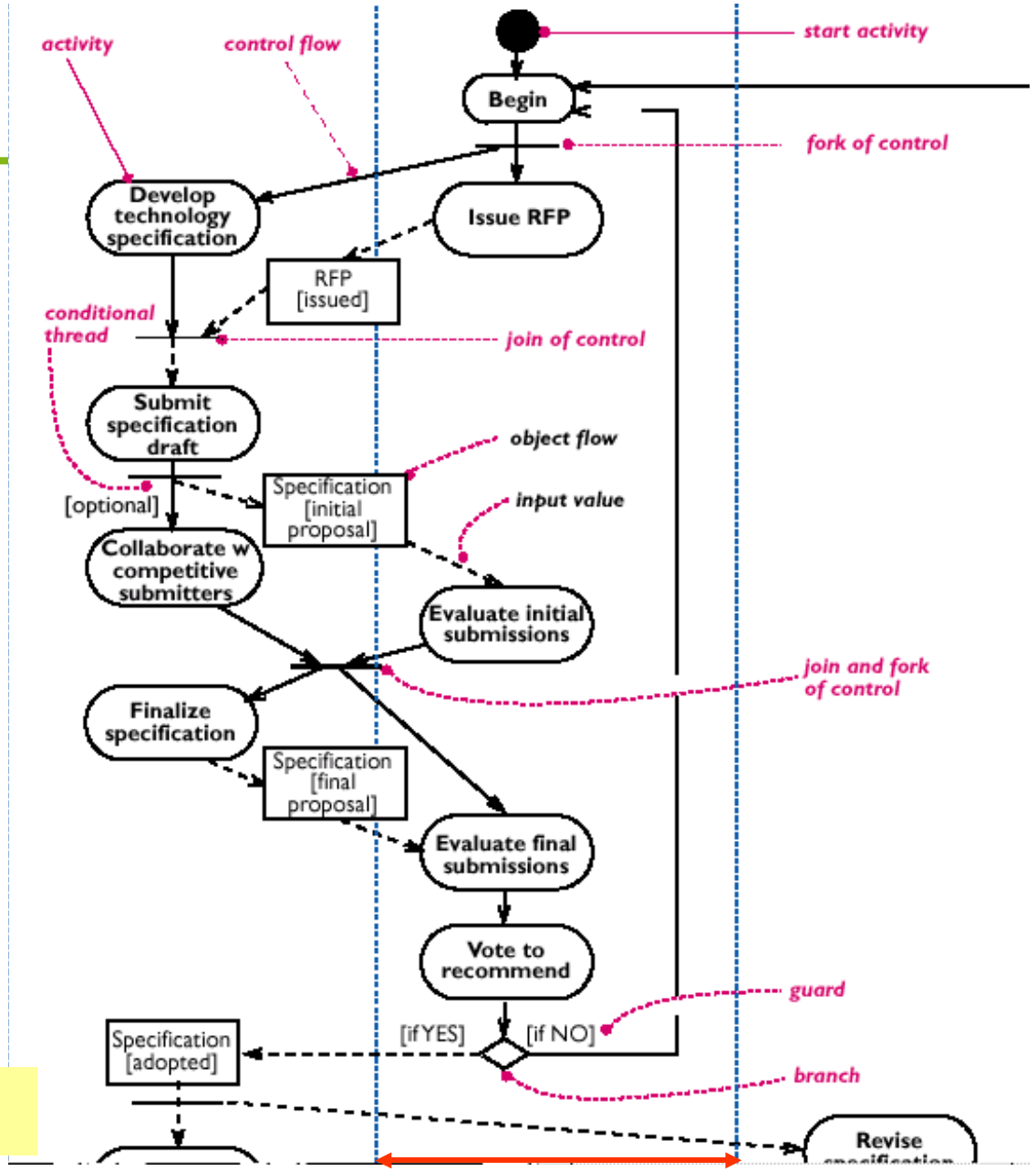
CONS:

- describes just one case, no timing tolerances: "What *does an MSC specification mean: does it describe all behaviors of a system, or does it describe a set of sample behaviors of a system?"* *

* H. Ben-Abdallah and S. Leue, "Timing constraints in message sequence chart specifications," in *Proc. 10th International Conference on Formal Description Techniques FORTE/PSTV'97*, Chapman and Hall, 1997.

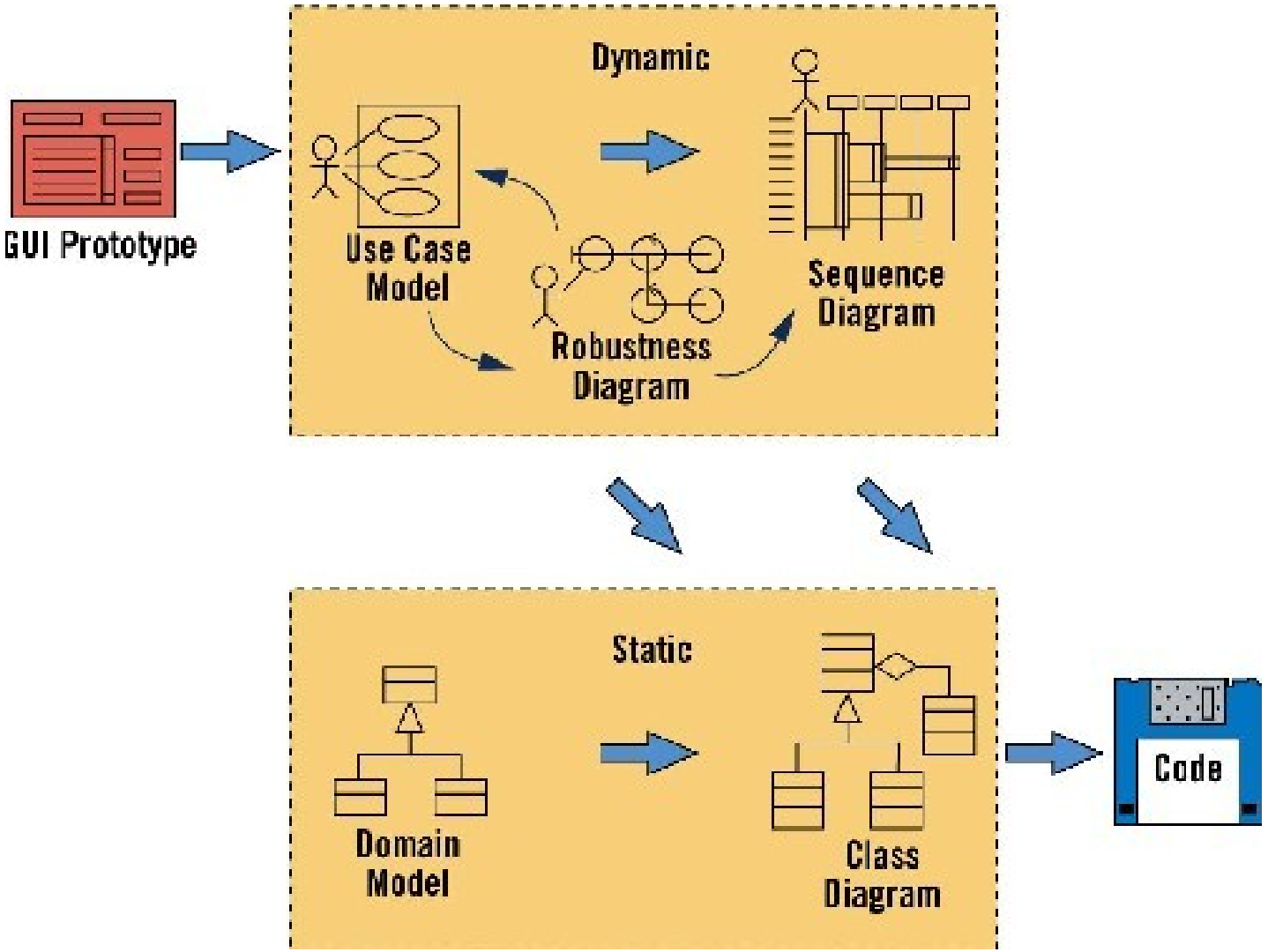
Computational Graphs @ UML: Activity diagram

Extended Petri nets.
Include decisions (like in flow charts).
Graphical notation similar to SDL.



© Cris Kobryn: UML 2001: A Standardization Odyssey, CACM, October, 1999

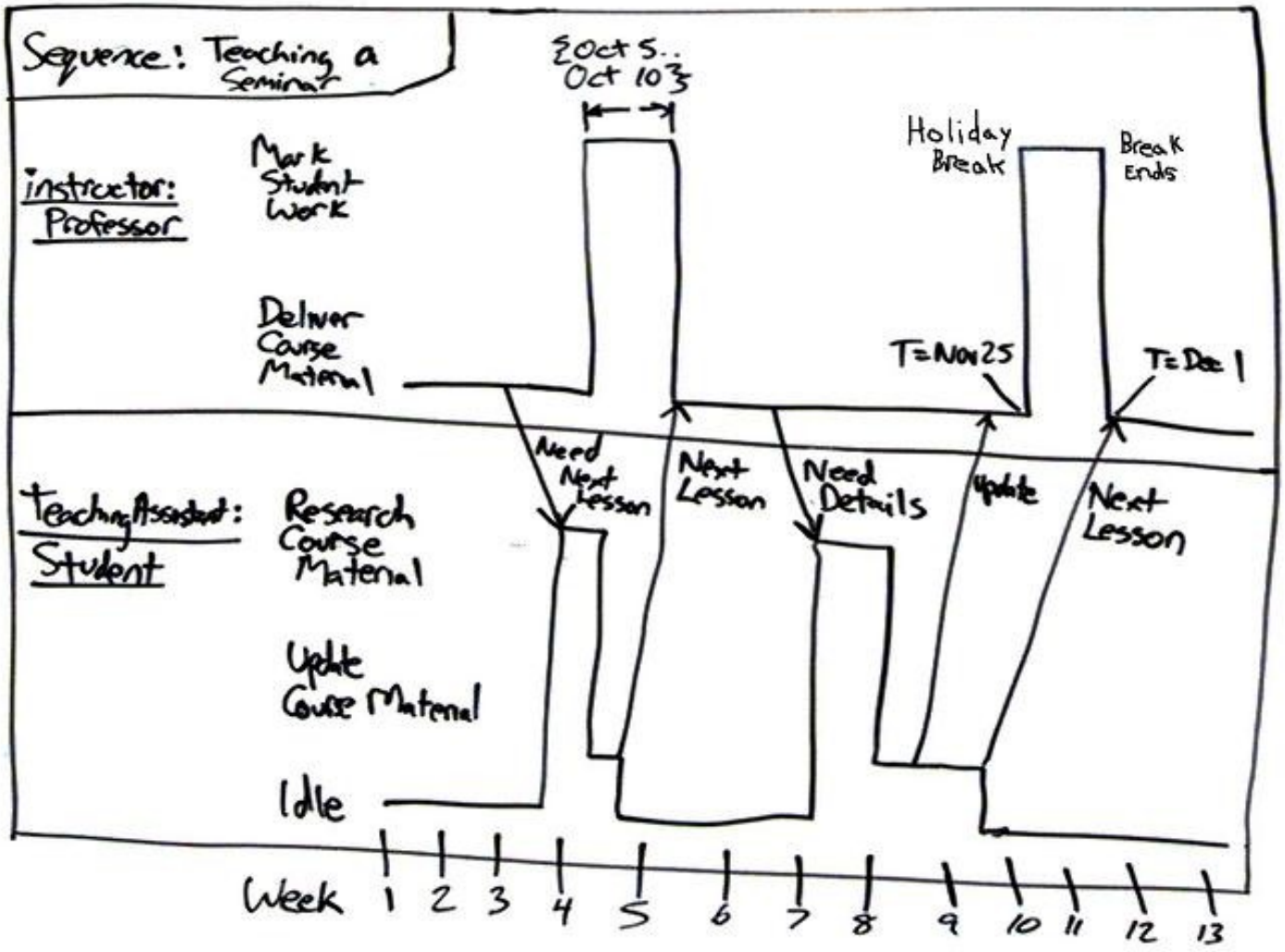
UML diagrams



From: www.sdmagazine.com/documents/s=815/sdm0012c/

Timing diagrams

Can be used to show the change of the state of an object over time.



© Scott Ambler, Agile Modeling, //www.agilemodeling.com, 2003

UML for real-time?

Initially not designed for real-time.

Initially lacking features:

- Partitioning of software into tasks and processes
- specifying timing
- specification of hardware components
- Projects on defining real-time UML based on previous work
- ROOM [Selic] is an object-oriented methodology for real-time systems developed originally at Bell-Northern Research.
- “UML profile for schedulability, performance and time“ http://www.omg.org/cgi-bin/doc?ptc/2002-03-02

UML Profiles Relevant for SoC

Existing (OMG)	▪ SPT (Schedulability, Performance, and Timing Analysis) ▪ Testing Profile ▪ QoS and Fault Tolerance
	▪ SysML (System Modelling Language) ▪ UML Profile for SoC
Upcoming (OMG)	▪ MARTE (Modeling and Analysis of Real-Time Embedded Systems)
non-OMG	▪ UML/SystemC (STMicroelectronics) ▪ SPRINT Profile (ST, NXP, Infineon, …)

DATE 2007 UML Workshop – W. Mueller 7/17

Example: Activity diagram with annotations

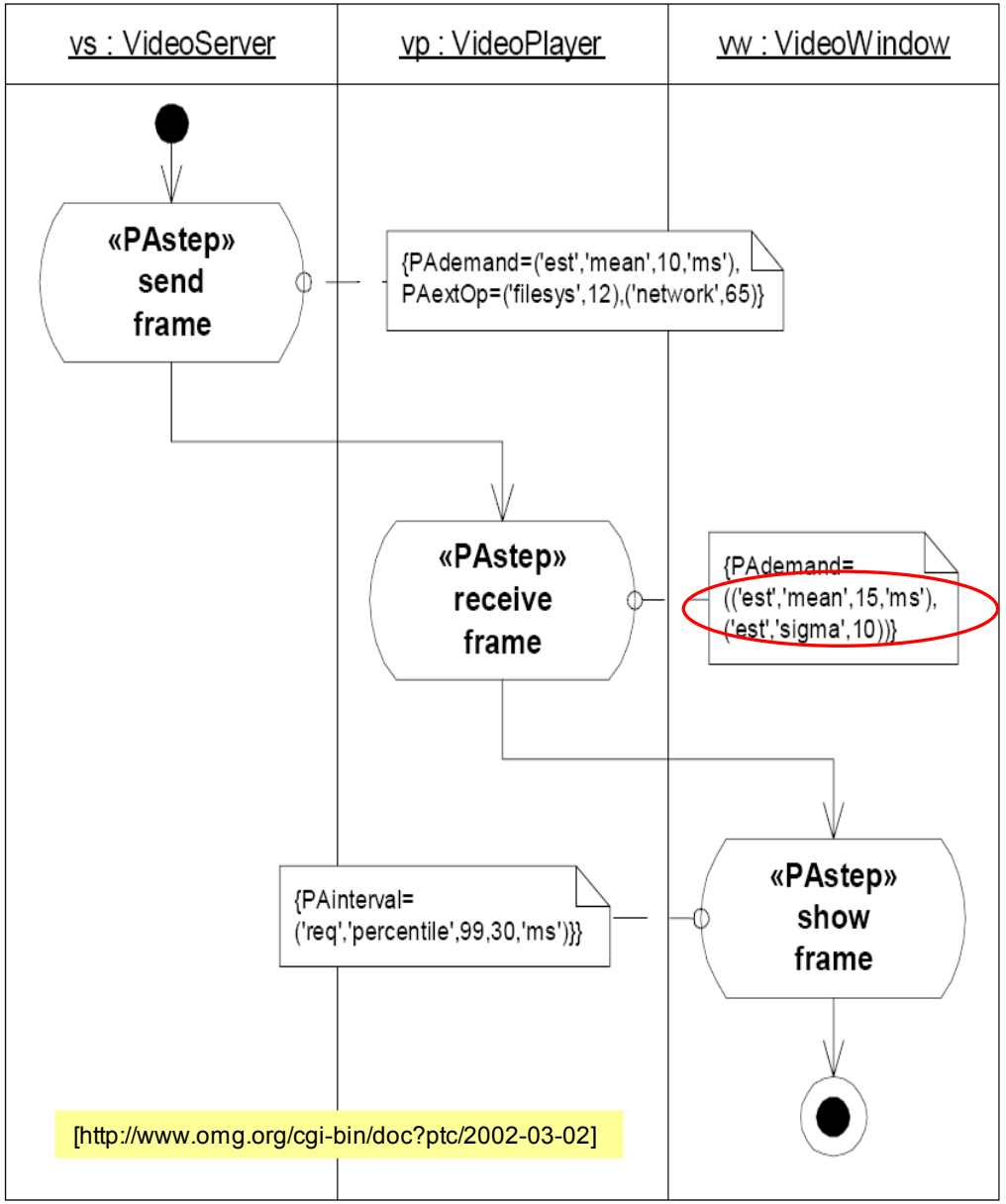


Figure 8-10 Details of the "send video" subactivity with performance annotations

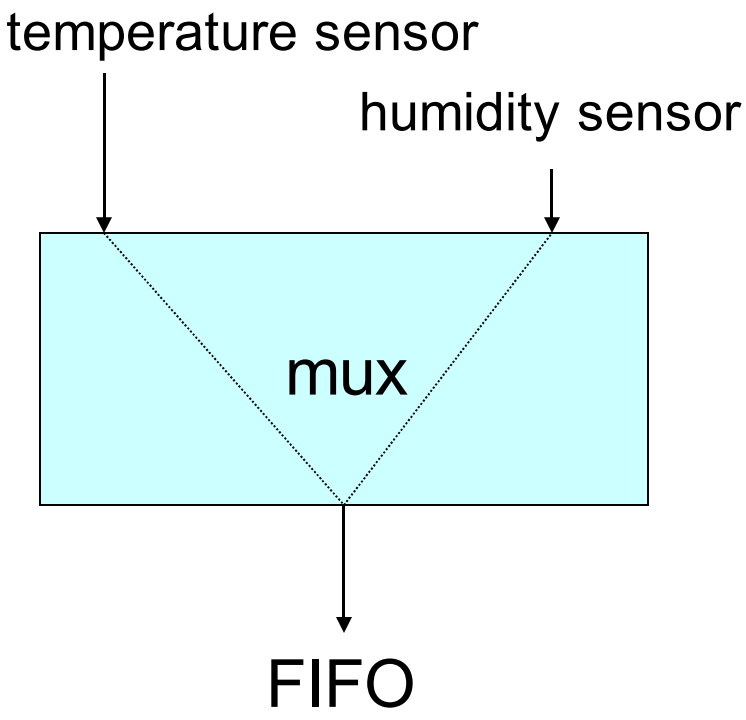
See also W. Müller et al.: UML for SoC, http://jerry.c-lab.de/uml-soc/

UML Profile Summary

- UML Profile comes as class diagrams with constraints, textual outlines (semantics), icons, diagram symbols, …
  - Constraints and behavioral semantics typically leave several issues open (variation points)
- Different OMG profiles of related domains may not be compatible!
- Current OMG UML Profiles are mainly for modelling
- UML Profiles do not come with a formal semantics
- … but Hardware Design is not just modelling
- HW verification and synthesis requires a well-defined and precise behavioral semantics
- Several UML tools already support UML profile definition

Task graphs

Many applications can be specified in the form of a set of communicating processes.
Example: system with two sensors:



Alternating read

```
loop
 read_temp; read_humi
until false;
```

of the two sensors no the right approach.

The case for multi-process modeling

```
MODULE main;
 TYPE some_channel =
         (temperature, humidity);
      some_sample : RECORD
                     value : integer;
                     line  : some_channel
                    END;
 PROCESS get_temperature;
 VAR sample : some_sample;
 BEGIN
  LOOP
   sample.value := new_temperature;
   IF sample.value > 30 THEN ....
   sample.line := temperature;
   to_fifo(sample);
  END
 END get_temperature;
```

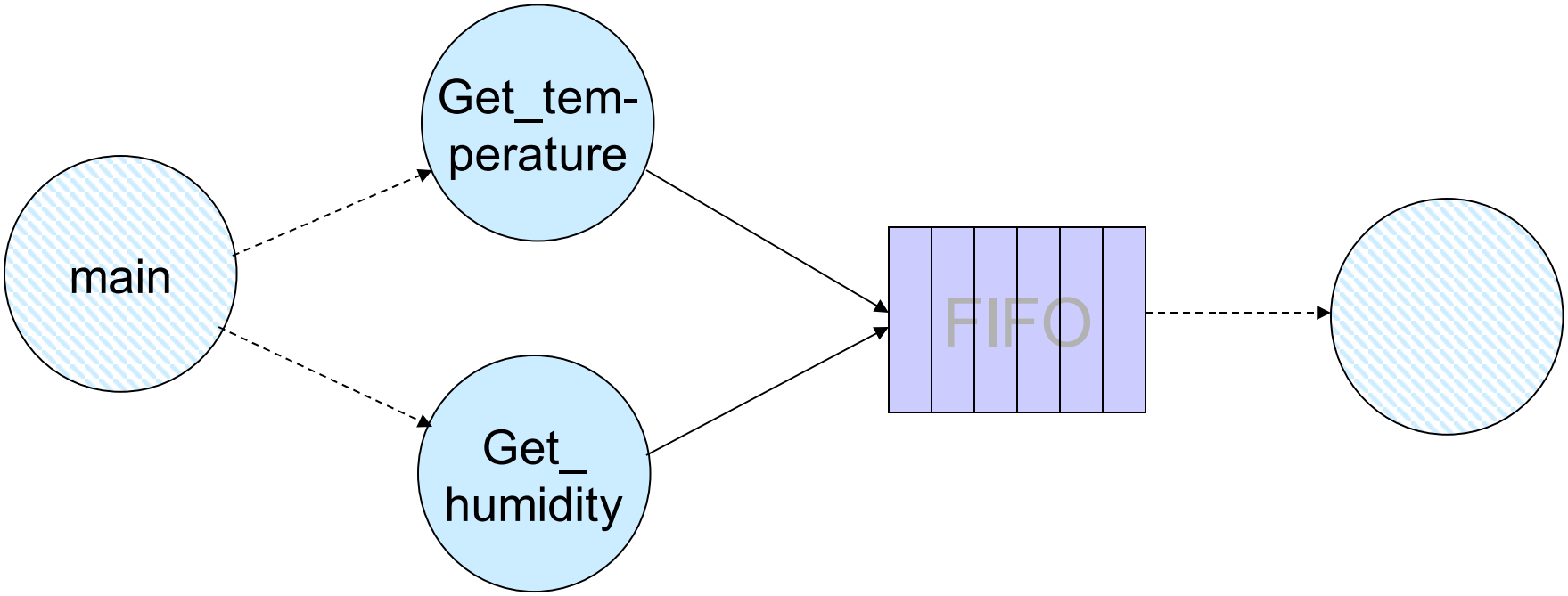
```
PROCESS get_humidity;
  VAR sample : some_sample;
  BEGIN
   LOOP
    sample.value := new_humidity;
    sample.line := humidity;
    to_fifo(sample);
   END
  END  get_humidity;

 BEGIN
   get_temperature; get_humidity;
 END;
```

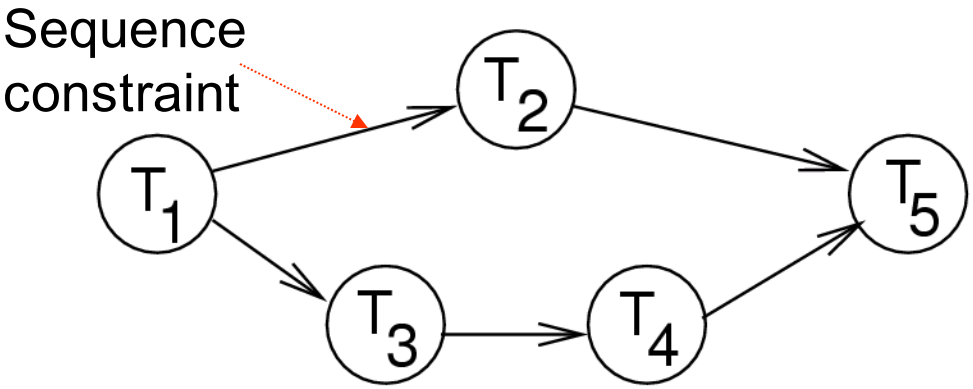
- Blocking calls new_temperature, new_humidity
- **Structure clearer than alternating checks for new values in a single process**

How to model dependencies between tasks/processes?

Dependences between processes/tasks



Task graphs

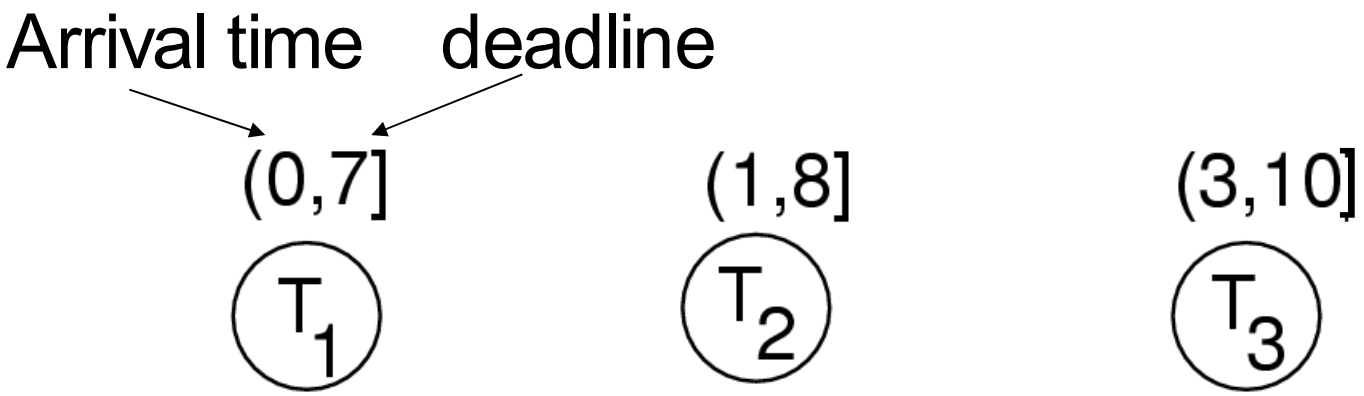


Nodes are assumed to be a “program“ described in some programming language, e.g. C or Java.

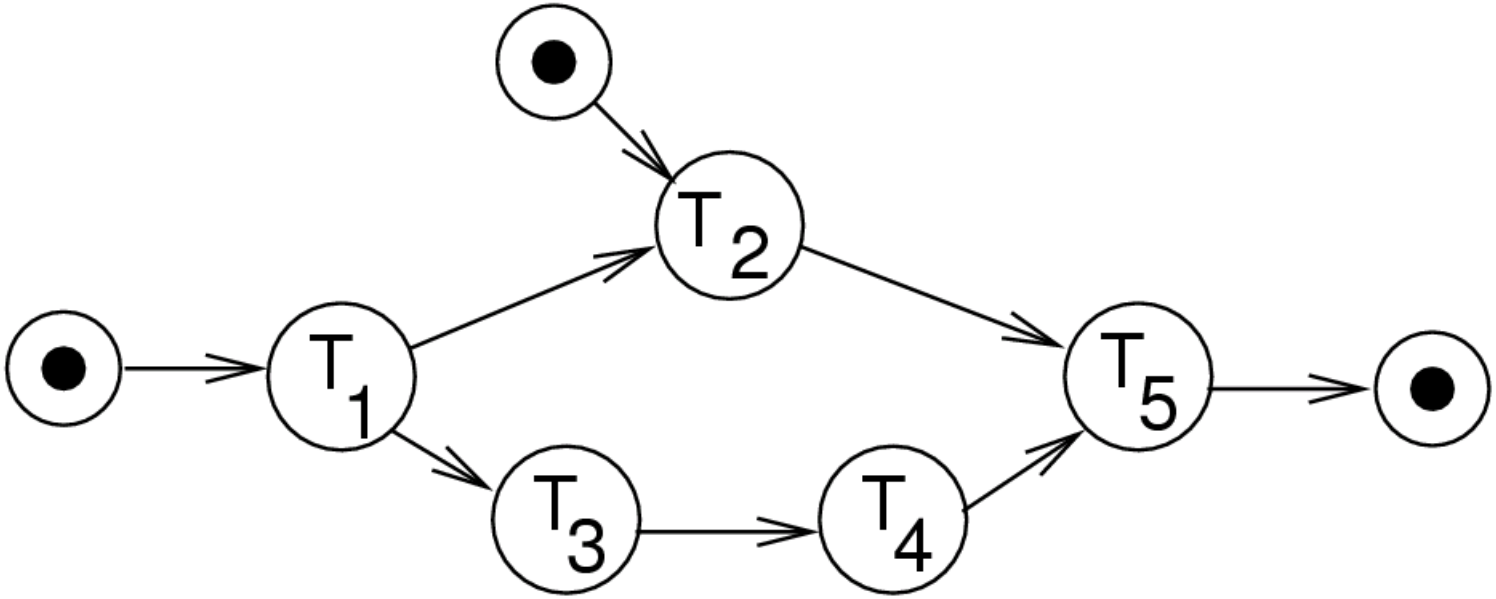
Def.: A **dependence graph** is a directed graph $G=(V,E)$ in which $E \subseteq V \times V$ is a partial order.
If $(v1, v2) \in E$, then *v1* is called an **immediate predecessor** of *v2* and *v2* is called an **immediate successor** of *v1*.
Suppose E^* is the transitive closure of E.
If $(v1, v2) \in E^*$, then *v1* is called a **predecessor** of *v2* and *v2* is called a **successor** of *v1*.

Task graphs
1. Timing information -

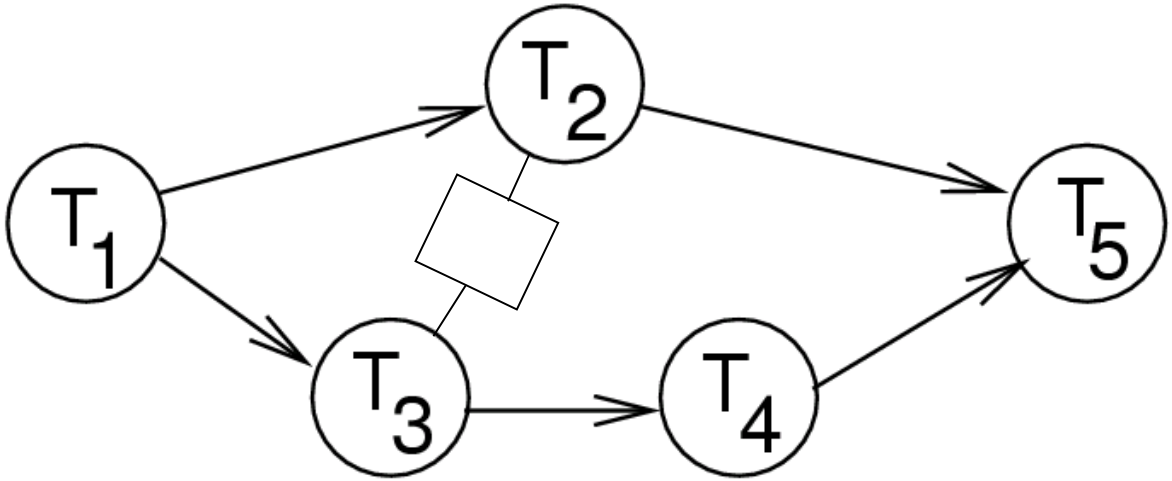
Task graphs may contain additional information,
for example: Timing information

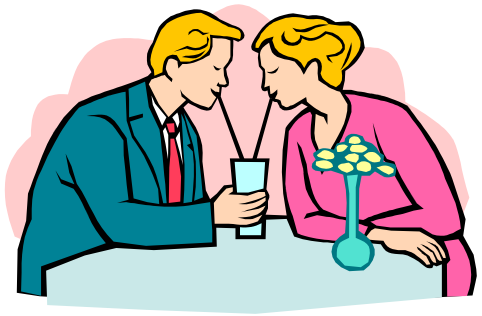


Task graphs
2. I/O-information

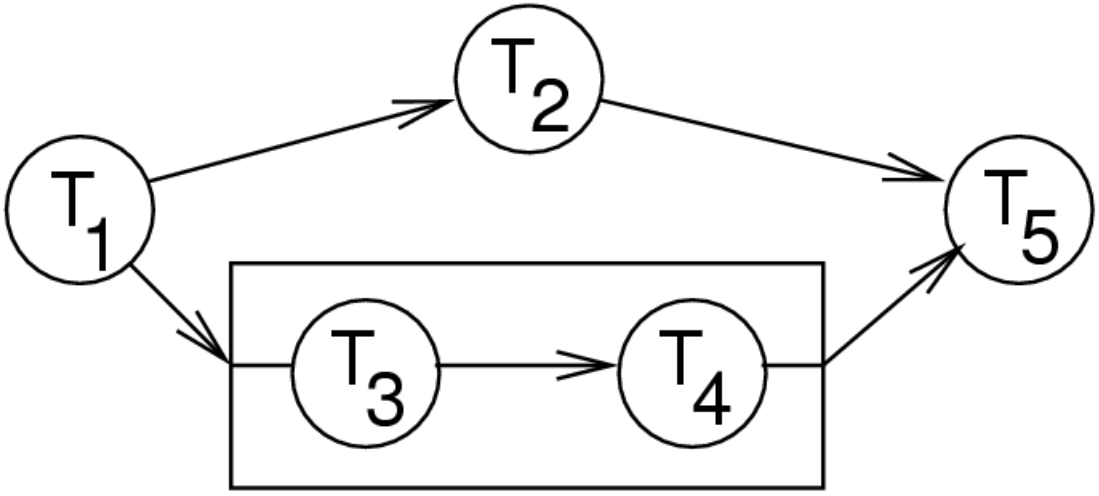


Task graphs
3. Shared resources





Task graphs
4. Hierarchical task graphs -



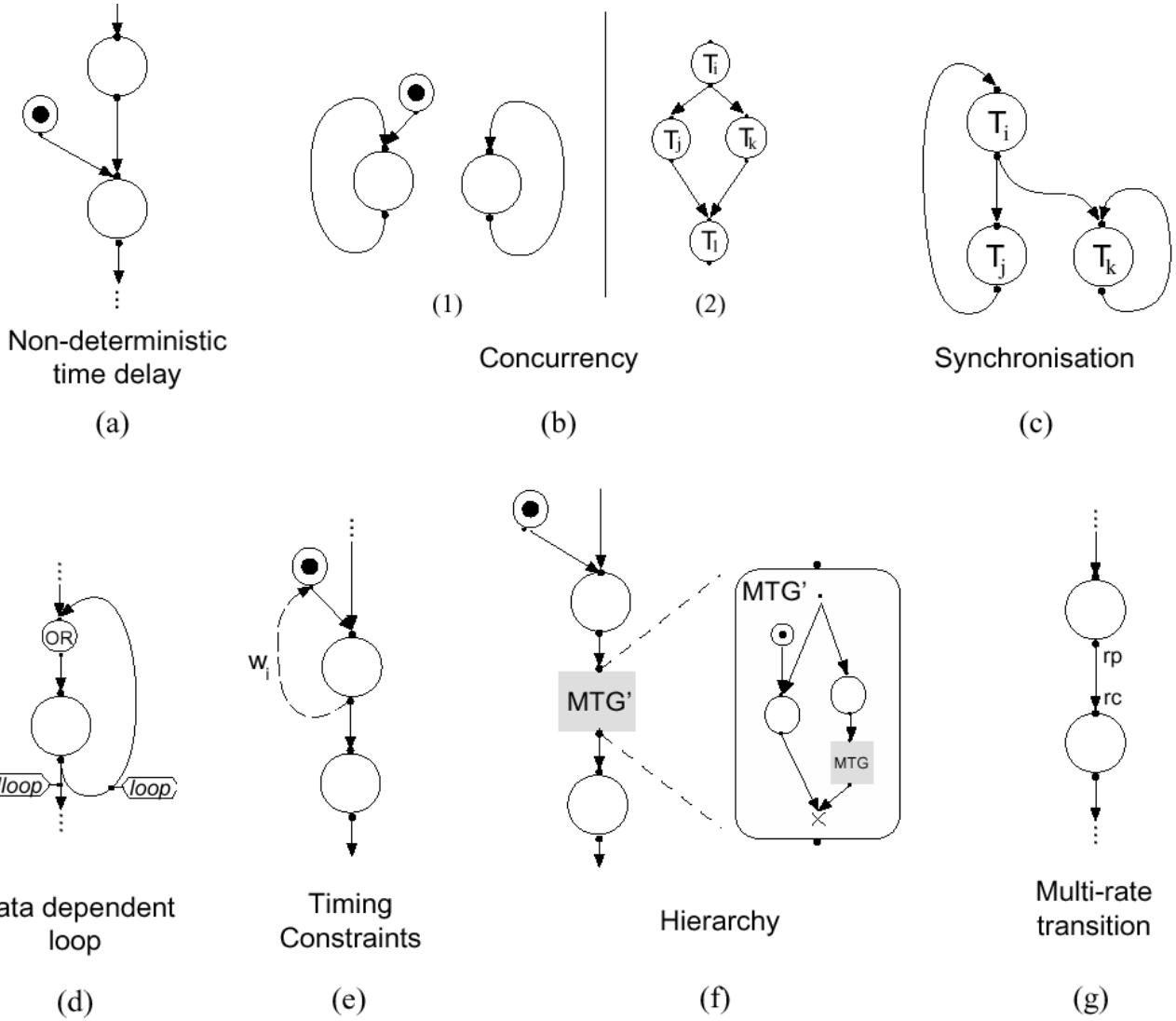
Distinction between task graphs and von Neumann computing somewhat blurring; precise semantics very desirable!

Multi-thread graphs (IMEC)

Def.: A multi-thread graph M is defined as an 11-tuple ($O, E, V, D, \vartheta, \iota, \Lambda, E^{lat}, E^{resp}, \nabla^{i}, \nabla^{av}$) with:

- O: set of operation nodes,
- E: set of control edges,
- V, D : refer to the access of variables,
- ι: is the set of input/output nodes,
- Λ: associates execution latency intervals with all threads,
- E^{lat}, E^{resp}, ∇^{i}, ∇^{av} are timing constraints.

MTG graphs: graphical notation



Summary

More computational graphs:

- (Message) sequence charts
- Time/distance charts
- Activity charts
- Other UML graphs
- UML profiles
- Real-time UML
- Task graphs