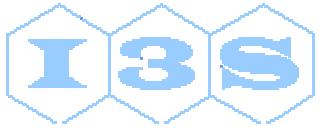
Towards a "Synchronous Reactive" UML Profile?

Robert de Simone

Charles André



Synchronous hypotheses

S/R stands for Synchronous Reactive

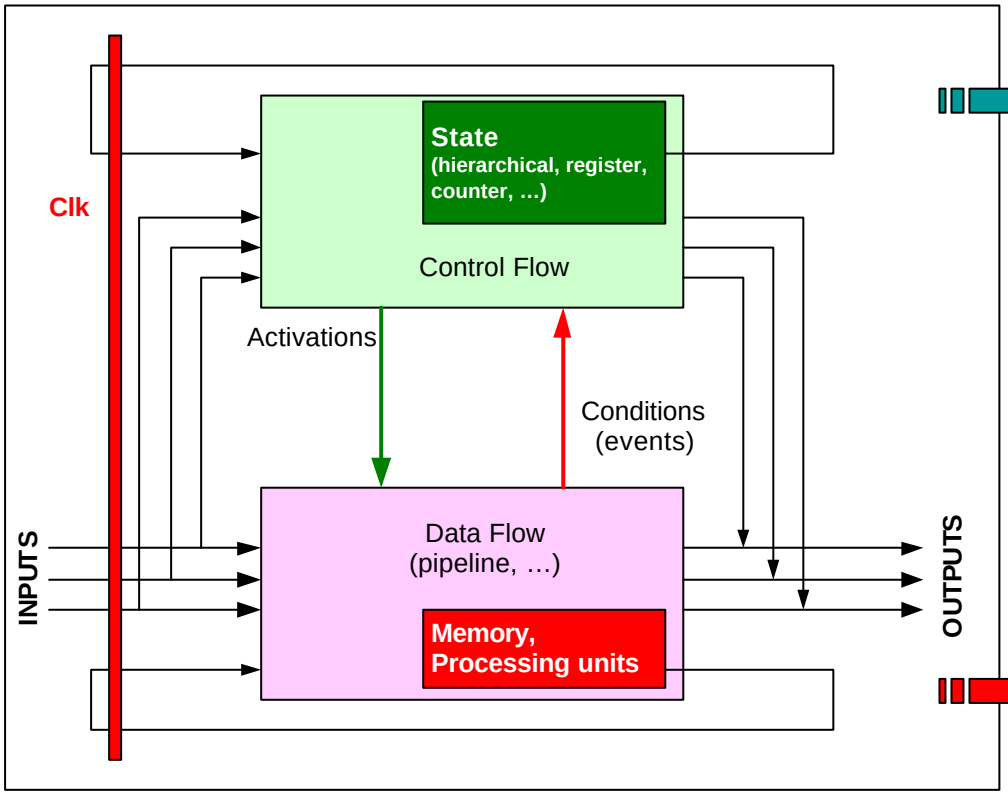
- Logical division of time into instants
- At each instant: execute a synchronous cycle (a reaction)
  - Acquisition
  - Compute (a **global** run-to-completion)
  - Actuate
- Signals are the unique support for communication

Where S/R should be used

- Applications:
  - Embedded controller, HMI, HW/SW, …
- Formalisms:
  - Strictly synchronous
    - SCADE, Esterel Studio, Block Diagrams
  - Almost synchronous
    - Statecharts, VHDL/Verilog, Simulink, Scicos

Circuits CAD
DSP, autom. Control
simulators

Control-flow / Data-flow



Esterel/SyncCharts:

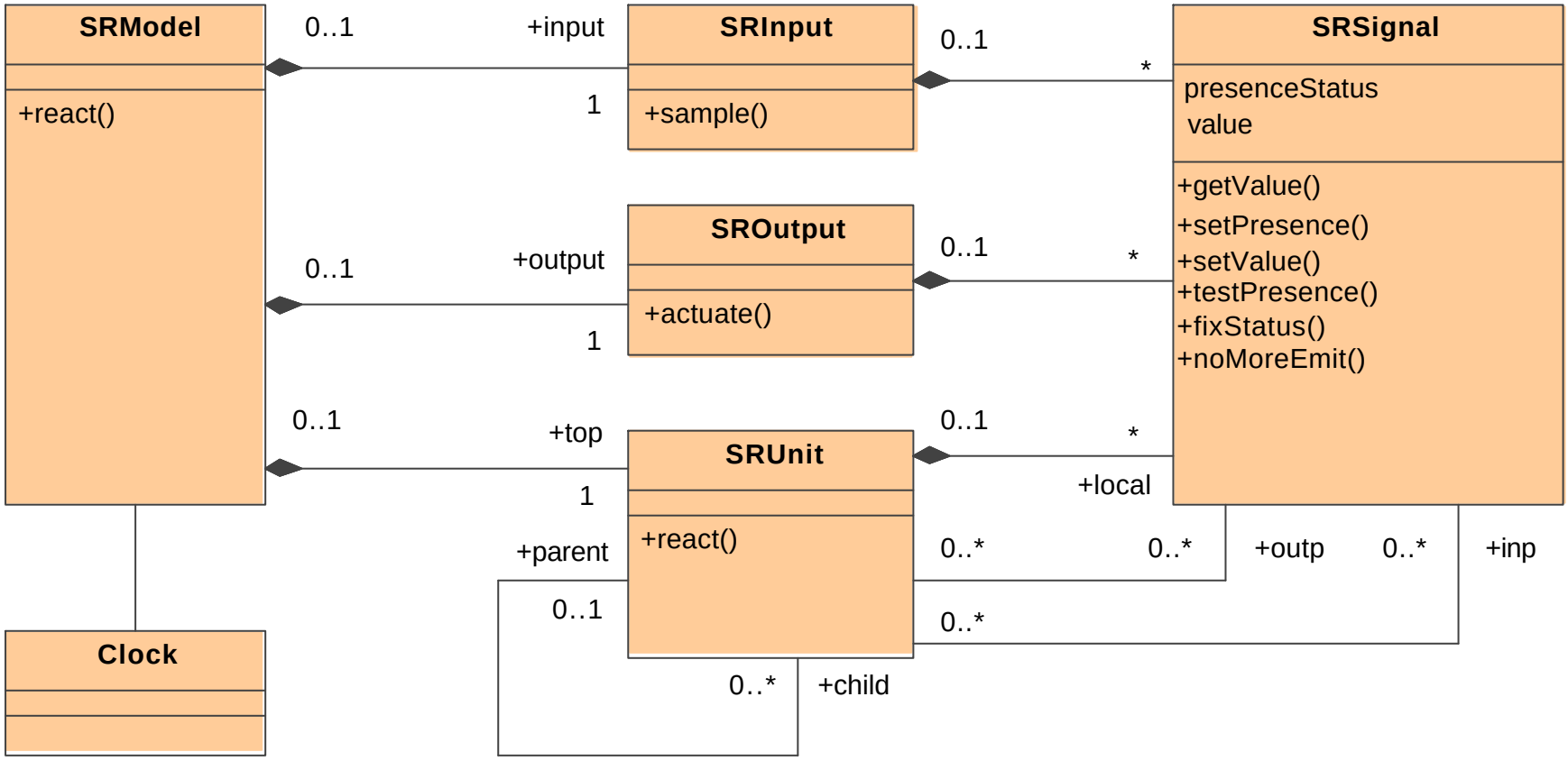
state-oriented

S/R capsules

Lustre/Scade:

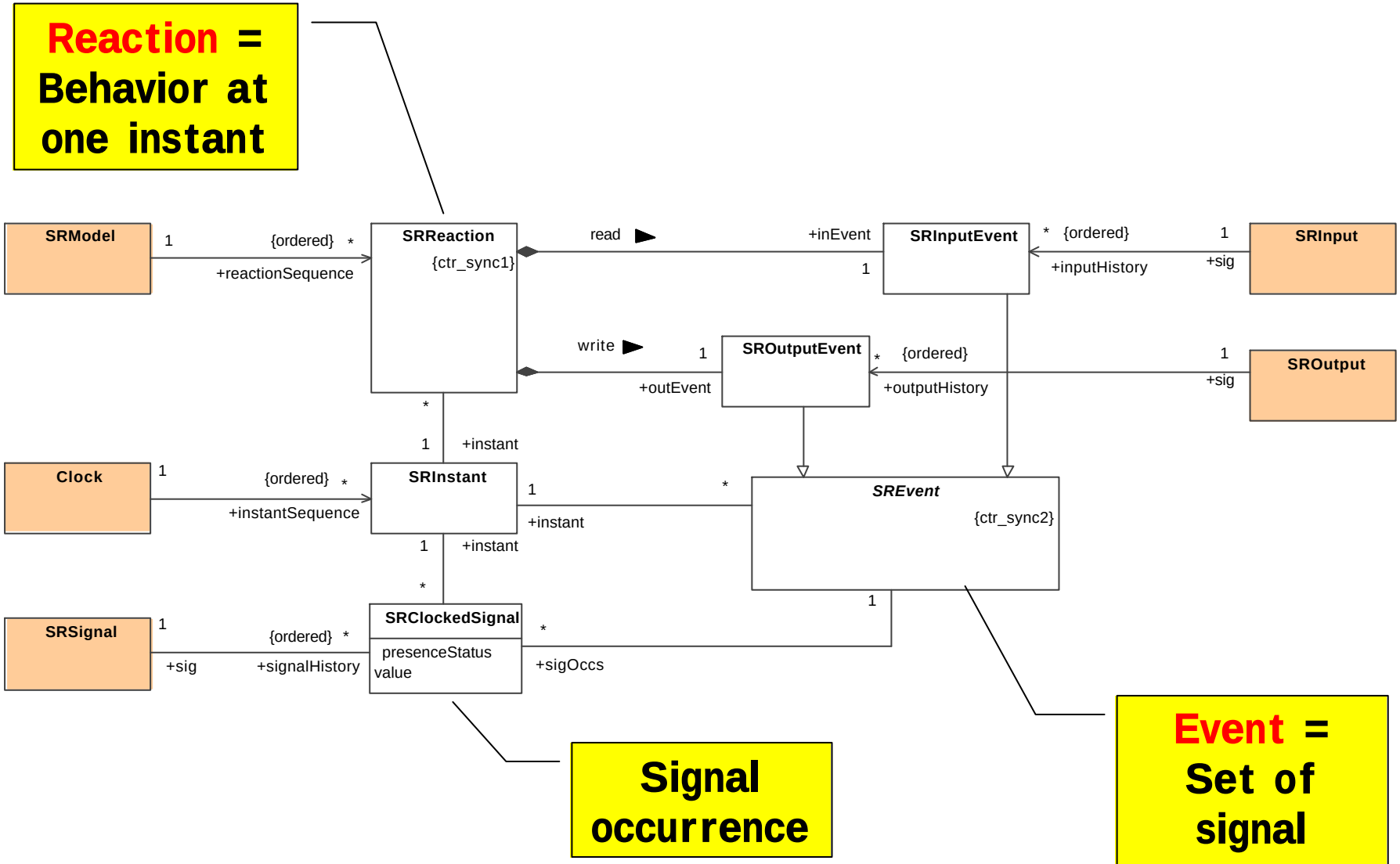
activity-oriented ?

S/R semantic domain

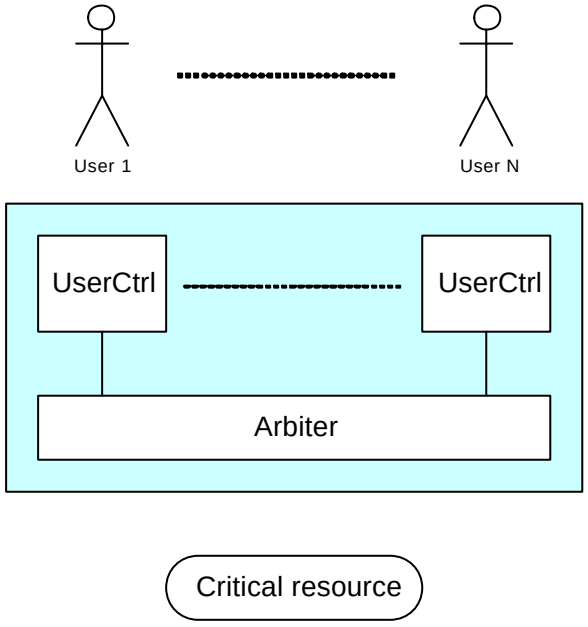


Not a metamodel – Only to represent main concepts

Dynamic semantics

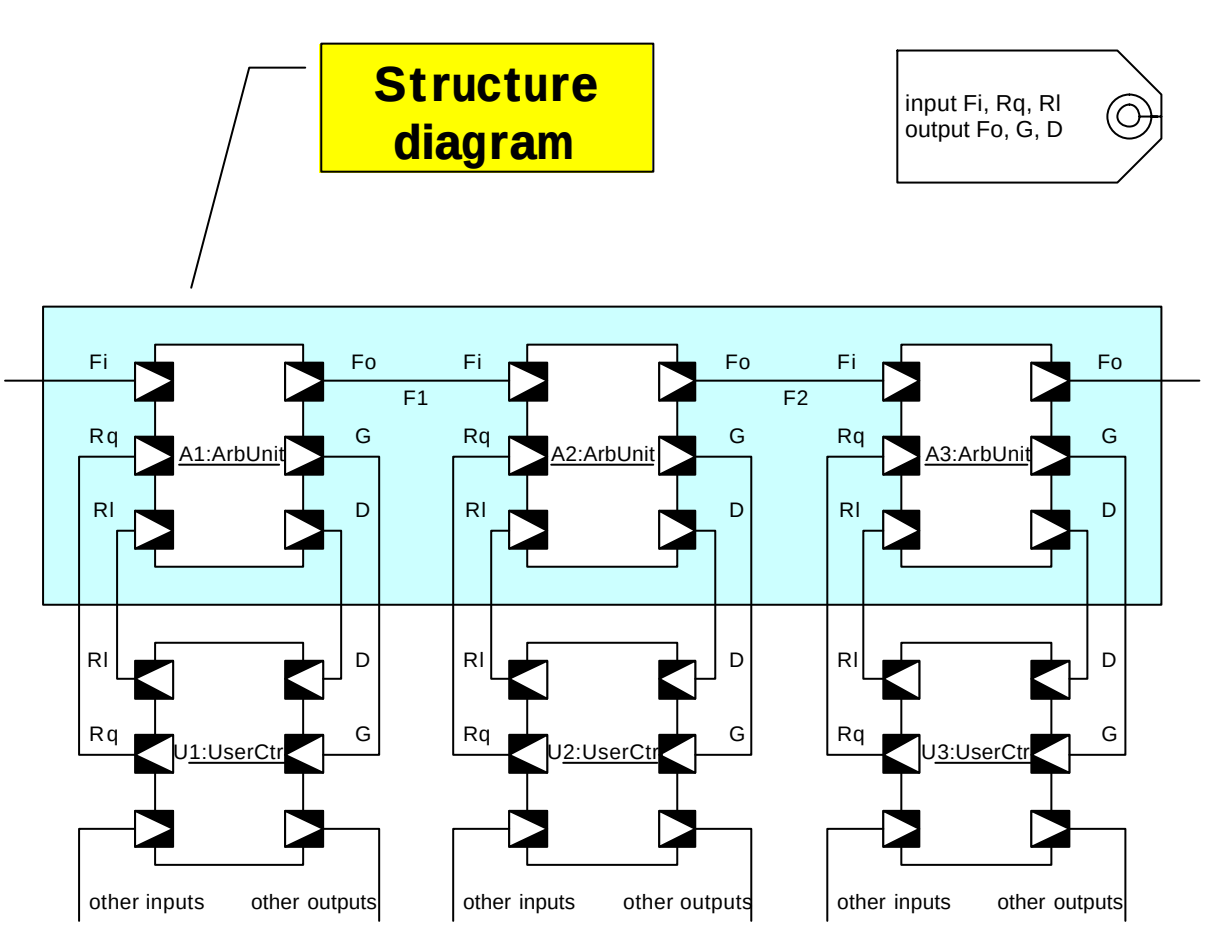


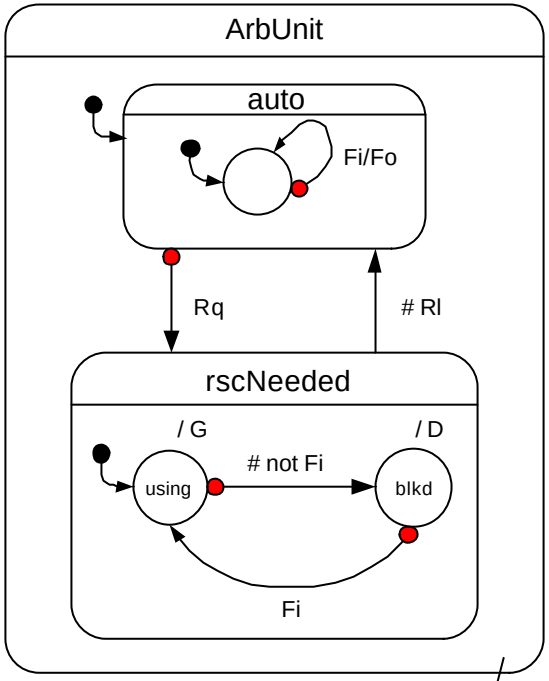
Example of an Arbiter



- Users: Rq and Rl
- Arbiter: G, D
- Exclusive access
- Static priority
- D valued with the nb of candidates for the resource with higher priority
- Linear description vs. the nb of users

State-based synchronous modeling





syncCharts

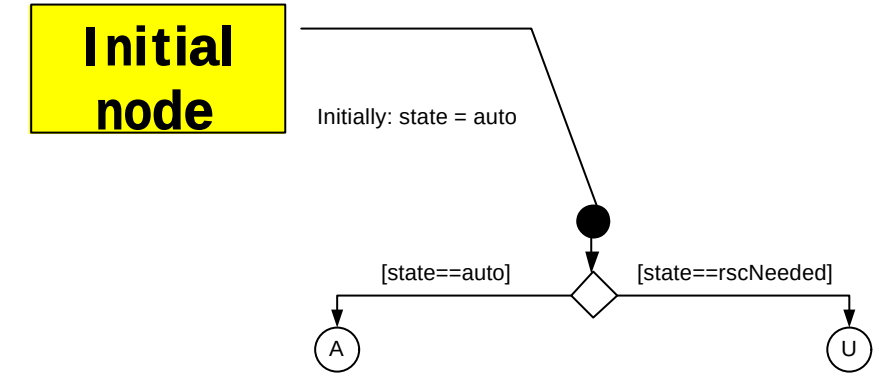
Limitations of the UML State Machines

- Object-based variant of statecharts
- Semantics described in terms of operations of a hypothetical machine
- Event queue + dispatcher
- Events are dispatched and processed one at a time
- Run-to-completion assumption
- Poor support for concurrency

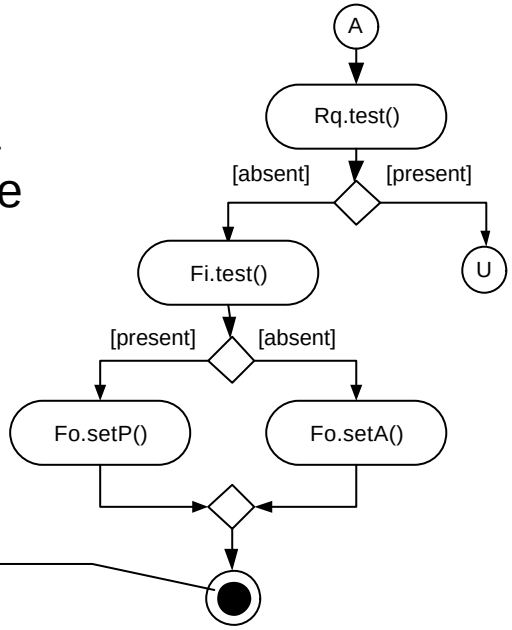
What is missing

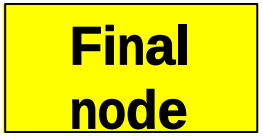
- Dealing with one event at a time is not acceptable for S/R models: many is the rule
- Combination of events (signals)
- Run-to-completion: to much restrictive. S/R have high degree of concurrency
- A notion out of the scope of classical asynchronous models: reaction to the absence
- Needs: a clear notion of instant

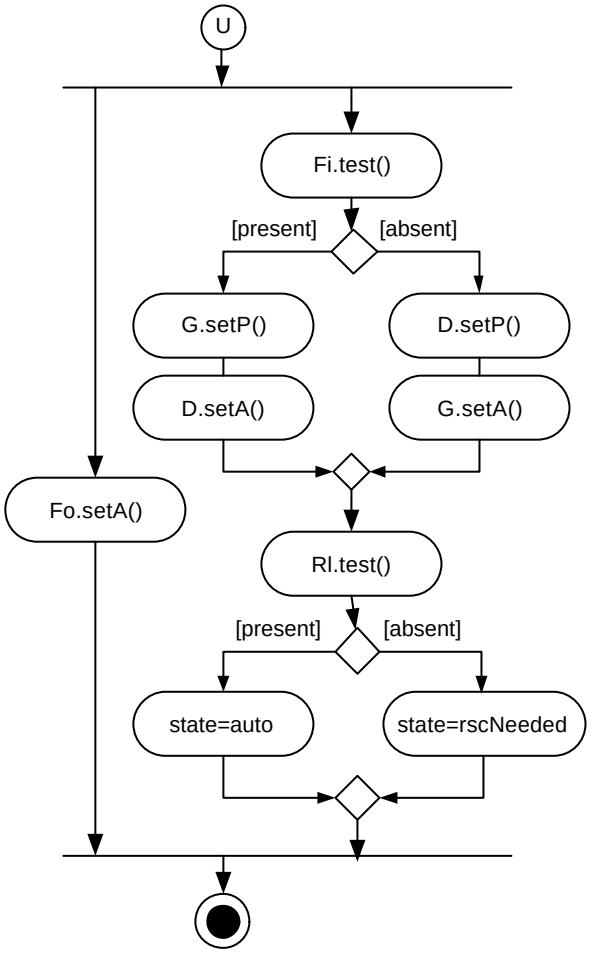
Activity-oriented approach



Activity diagram for a reaction of the ArbUnit







Activities from initial to final within one instant

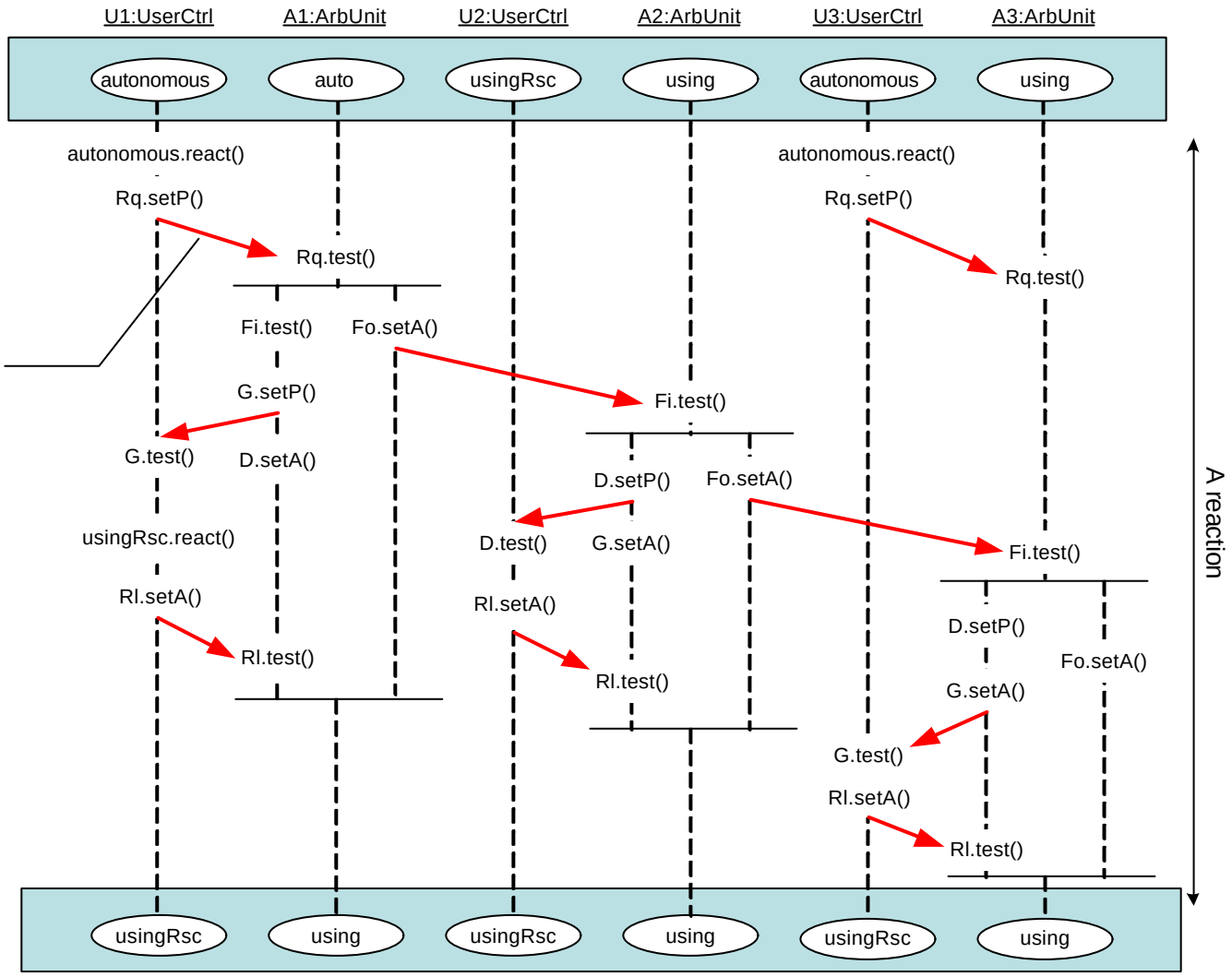
Emergent behavior

Finite and acyclic at each instant

Causality relation

A signal may be tested only after being set

Stable configuration



	SR	SPT
Purpose	High-level design & programming Implementation independent	Temporal analysis Performance analysis Real-time simulation Relevance of timing figures?
Time	Logical discrete time abstract causality within instant Strict instant: possibility to lose fleeting events	Reference clock (represents physical time) Causality appears as time precedence Signal buffering, queues

Scheduling in S/R

- Temporal and/or spatial mapping (SynDEx)
- Essential needs: respect causality
- Introduce chronometric time on architectural platform resources
- Often: a uniform scheduling for the whole application (i.e., a scheduling valid for all reactions).

Conclusion

- "Control system engineering" has specific needs, not all addressed by the UML
- The S/R approach answers some of them
- UML + …
- SPT OK for real-time cooperation of objects
- S/R well-suited for Control flow/Data flow tight interactions. But it demands relaxing some UML rules (especially for UML SM semantics)

S/R Mixed Architecture