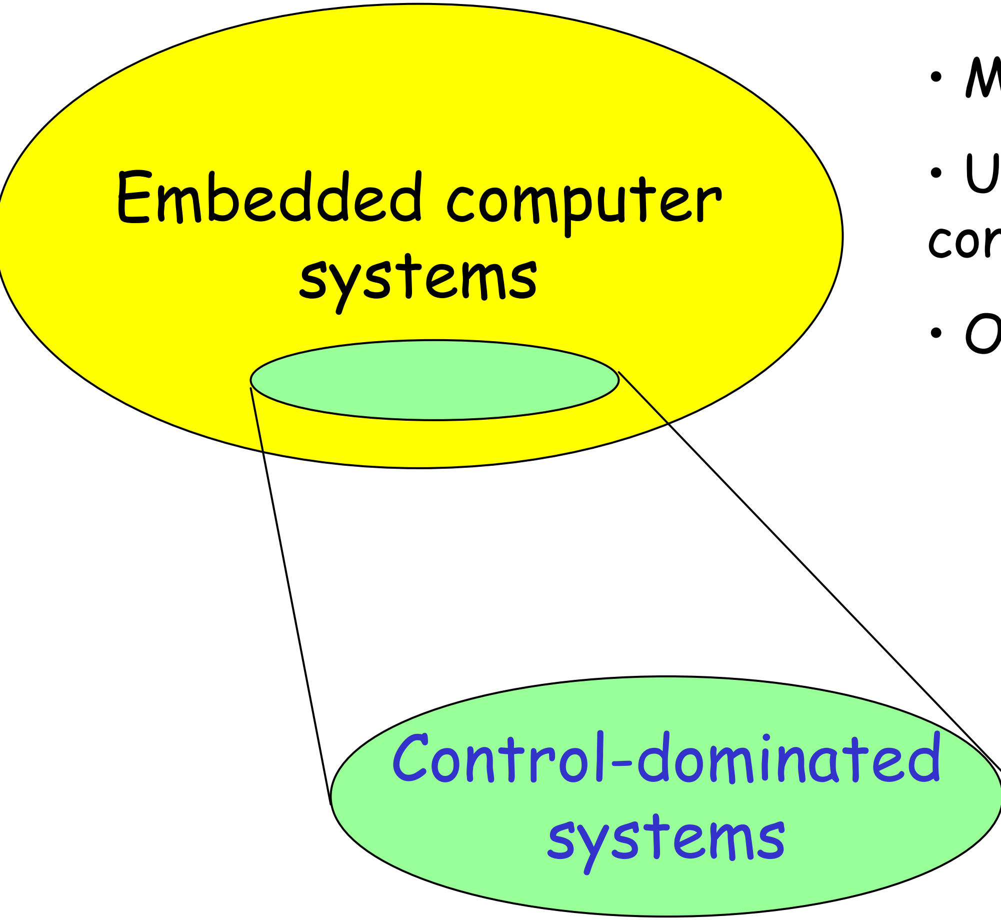
A Synchronous Approach to Reactive System Design

Charles ANDRE
University of Nice-Sophia Antipolis

Introduction



- More and more numerous
- Usually subject to real-time constraints
- Often safety-critical

A special class (controllers)

- discrete-event
- mostly control
- highly reactive

Introduction (Cont'd)

Controllers must react to all stimuli in a predictable and timely way.

- Demand for precise and powerful models

Sequential and concurrent evolutions

Hierarchical description

Exception handling

Introduction (Cont'd)

Controllers must react to all stimuli in a predictable and timely way.

- Demand for precise and powerful models
- Mathematical foundation

Formal analysis of the model

Guaranteed properties

Introduction (Cont'd)

Controllers must react to all stimuli in a predictable and timely way.

- Demand for precise and powerful models
- Mathematical foundation
- Efficient and safe implementation techniques

High quality

Dependability

Introduction (Cont'd)

Controllers must react to all stimuli in a predictable and timely way.

- Demand for precise and powerful models
- Mathematical foundation
- Efficient and safe implementation techniques
- Reusability

Design Once, Use Many

Educational Objectives

- Theoretical background

  Teaching fundamentals (Boolean Algebra, Finite State Machines, Petri Nets, ...)

- Skills

  In Design, Programming, Testing.

- Intellectual Curiosity

  Innovative concepts: modern design, synchrony, objects, model-checking, ...

Contents of the Course

- Classical Design
  - Logical design (combinatorial and sequential circuits)
  - Complex synchronous systems (sequencer+ F.Us)
  - Programmable systems (PLC, DSP, μControllers, μProcessors)
- From Circuits to Programs
  - Increasing complexity
  - Co-design
  - Systems (including RTOS)

Unfortunately, average E.E students lack knowledge and skills in Software Engineering !!

Rationale for Teaching SL

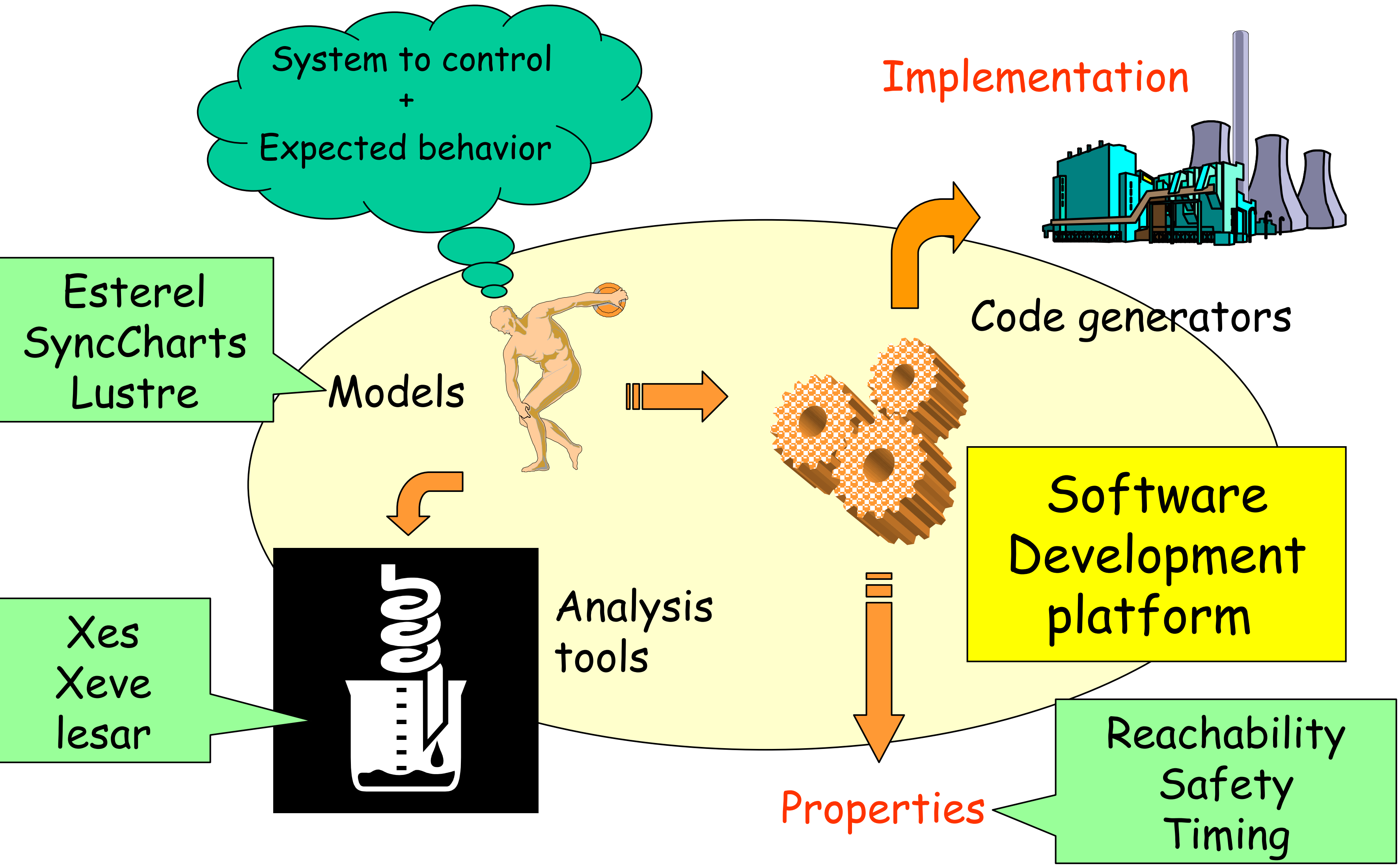
Synchronous languages:

- Simple languages
  - Restricted instruction sets
  - Static structures
- Tailored to control: special purpose languages
- Execution: sequence of stimuli/reaction
- Mathematical semantics

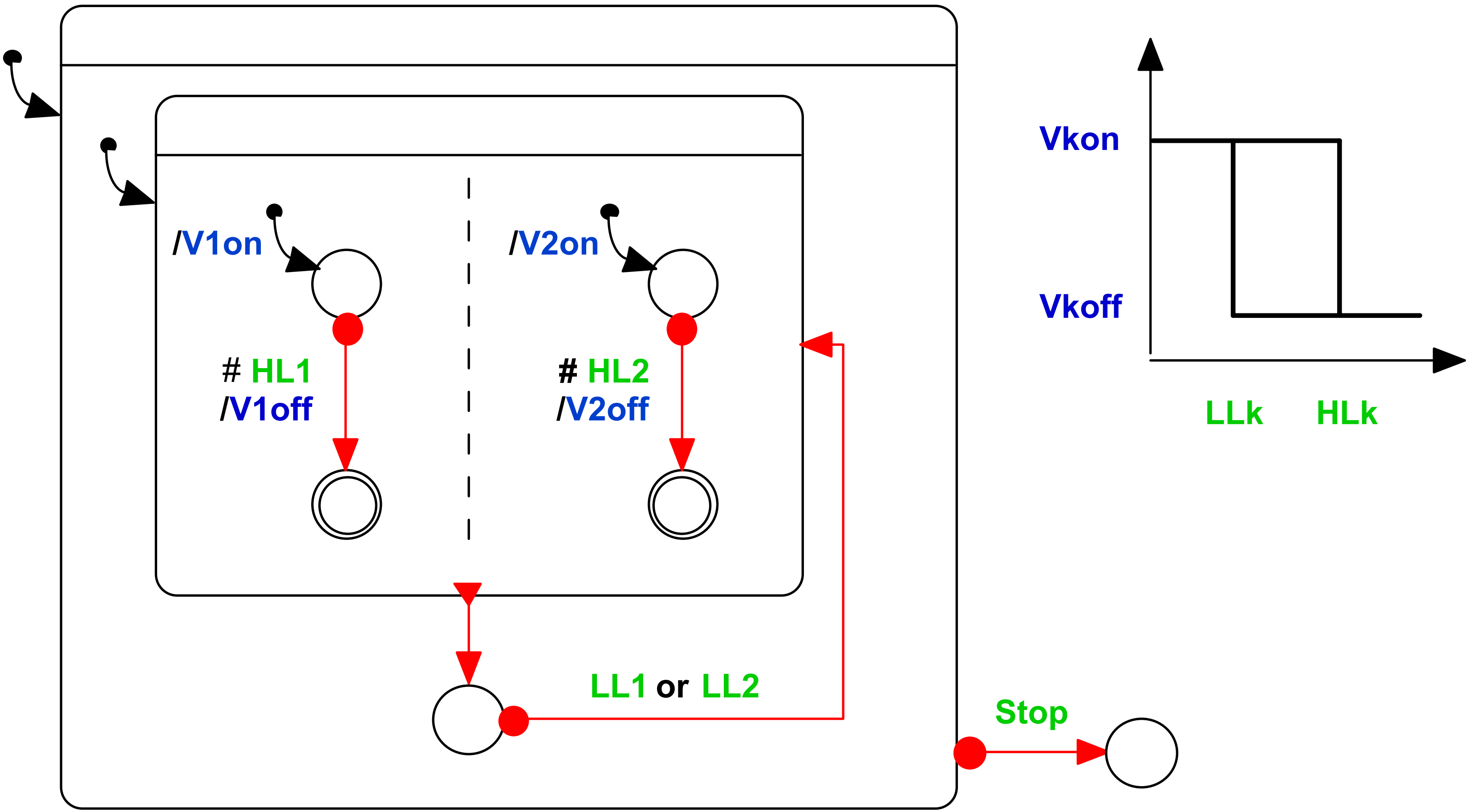
  FSM, set of Boolean equations, recurrence equations

  Formal verification

Software Environment



Example: a Simple Regulation



Example: an Esterel program

Declarative part

```
module Regul:
input HL1,LL1, HL2, LL2, Stop;
output V1on, V1off, V2on, V2off;
```

Reactive part

```
abort
 loop
  [
   emit V1on; await immediate HL1; emit V1off
      ||
   emit V2on; await immediate HL2; emit V2off
   ];
   await [LL1 or LL2]
  end loop
when Stop
end module
```

Future

- Introduction of OO-concepts
  - With an EE point of view
  - UML-RT, ROOM
- UML models
  - Use cases (functional view)
  - Dynamic modeling (statecharts, scenarios)
  - Objects and Classes (static structure)
  - Collaboration
  - Deployment
- Components and Architecture