The Second IEEE International Conference on Robotic Computing
IRC 2018

Towards a well-founded software component model for cyber-physical control systems

Jacques.Malenfant (at) lip6.fr

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6

Introduction

- How to allow/ease CPCS and robotics systems specification, implementation, test, verification, validation?
- Main proposals:
  1. Build over strong behavioral models (BM): stochastic hybrid systems & hybrid automata.
  2. Implement jointly the software and the simulator, using modular simulation models derived from the BM.

$\Rightarrow$ *Component model integrating software, BM and simulation.*

Introduction

- How to allow/ease CPCS and robotics systems specification, implementation, test, verification, validation?
- Main proposals:
  1. Build over strong behavioral models (BM): stochastic hybrid systems & hybrid automata.
  2. Implement jointly the software and the simulator, using modular simulation models derived from the BM.

  $\Rightarrow$ *Component model integrating software, BM and simulation.*
- Goals:
  1. Test, validation and verification through MIL, SIL and HIL simulations.
  2. Allow a progressive approach (unit $\Rightarrow$ integration $\Rightarrow$ full system).
  3. Provide a strong basis for a (large-scale distributed) CPCS software development process.
  4. Develop more reliable CPCS with less resources.

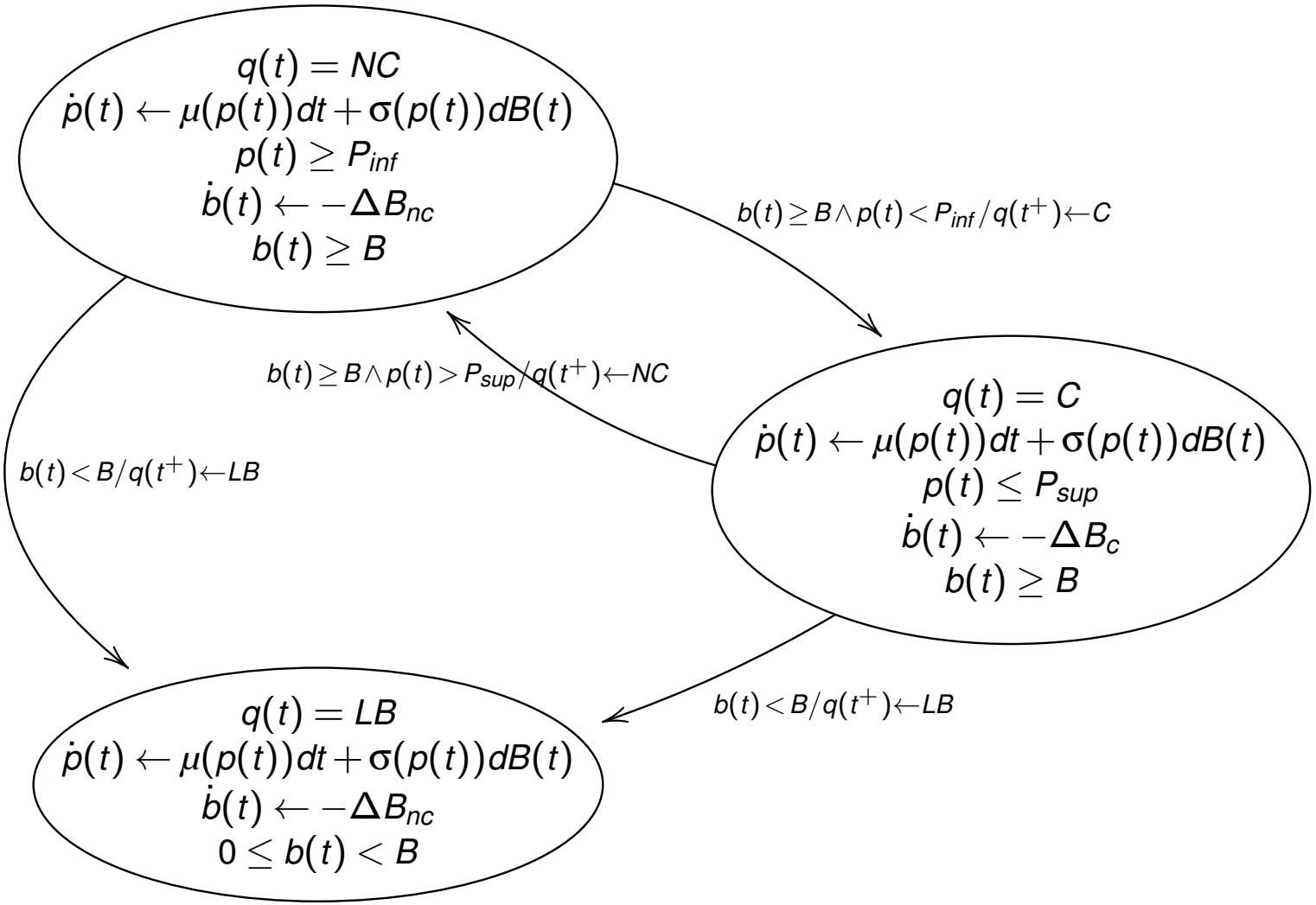
Conceptual framework developed in four steps

1. Well-founded BM with stochastic hybrid systems
2. Modularity and composability with hybrid automata
3. Concrete operational semantics using modular simulation
4. Integration in a software component model with full composability.

Hybrid systems

- Mathematical discrete/continuous behavioral (dynamic) models.
- Hybrid state space: $\mathcal{S} = \bigcup_{q \in \mathcal{Q}} X_q \times \{q\}$
- Discrete states: $\mathcal{Q} = \{q_0, q_1, ...\}$ (denumerable) with discrete transitions upon events:
  - value changes of discrete variables
  - conditions (frontiers) met by continuous variables
- Each discrete state has a continuous evolution model (*e.g.*, differential equations).
- Stochastic hybrid systems:
  - stochastic discrete transitions,
  - stochastic differential equations.

Hybrid system baseline model for the data transfer use case



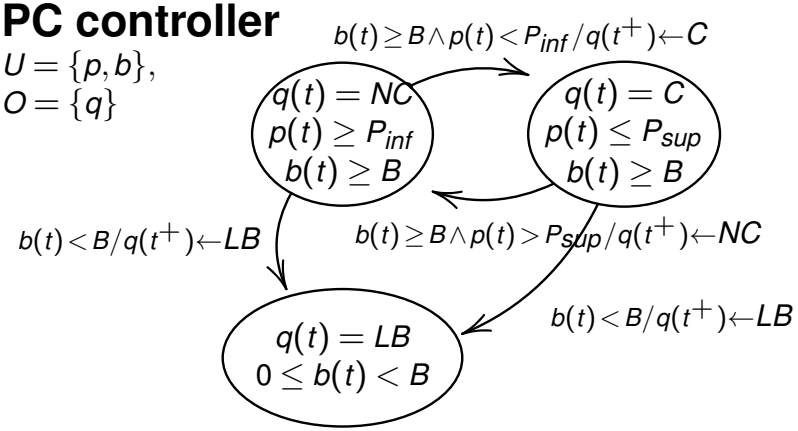
From monolithic hybrid systems to modular hybrid automata

- How to use hybrid systems in practice?
- Two major lines of work: Henzinger and Lynch.
- Hybrid automata:
  - continuous variables and discrete events
  - discrete transitions and continuous trajectories alternating to give the system overall trajectory
  - partition between internal and external variables and events
  - composition by sharing external variables and events
- Lynch's *et al.* Hybrid Input/Output automata (HIOA):
  - external events and variables are partitioned between input and output ones
  - when composing, only one producer (output) for each
- Lynch's *et al.* Timed Input/Output automata (TIOA):
  - no external continuous variables *i.e.*, no sharing thereof

Factorising the baseline model into HIOA

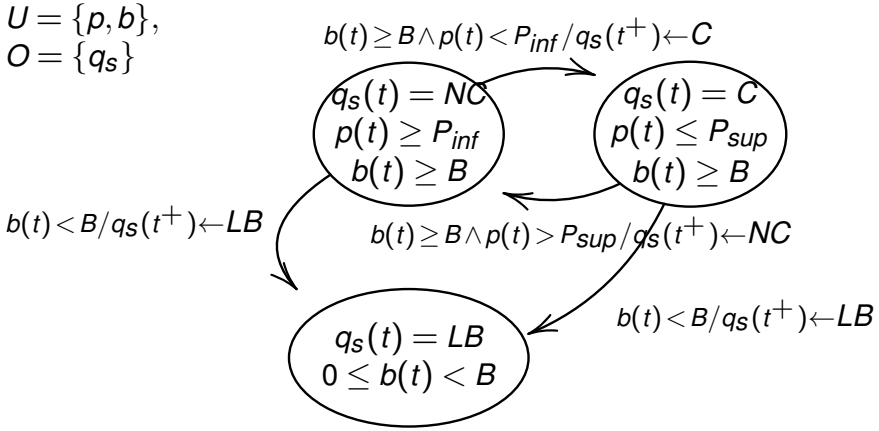
PC controller

$U = \{p, b\}$,
$O = \{q\}$



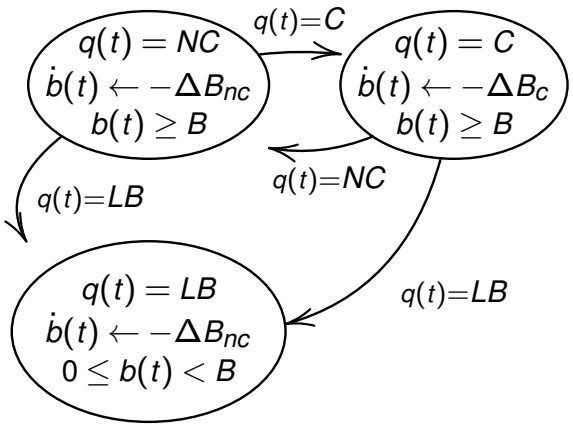
Server controller

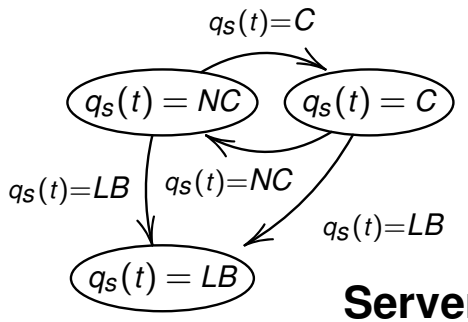
$U = \{p, b\}$,
$O = \{q_s\}$



PC

$Y = \{b\}$,
$I = \{q\}$





Server ($I = \{q\}$)

Environment

$Y = \{p\}$

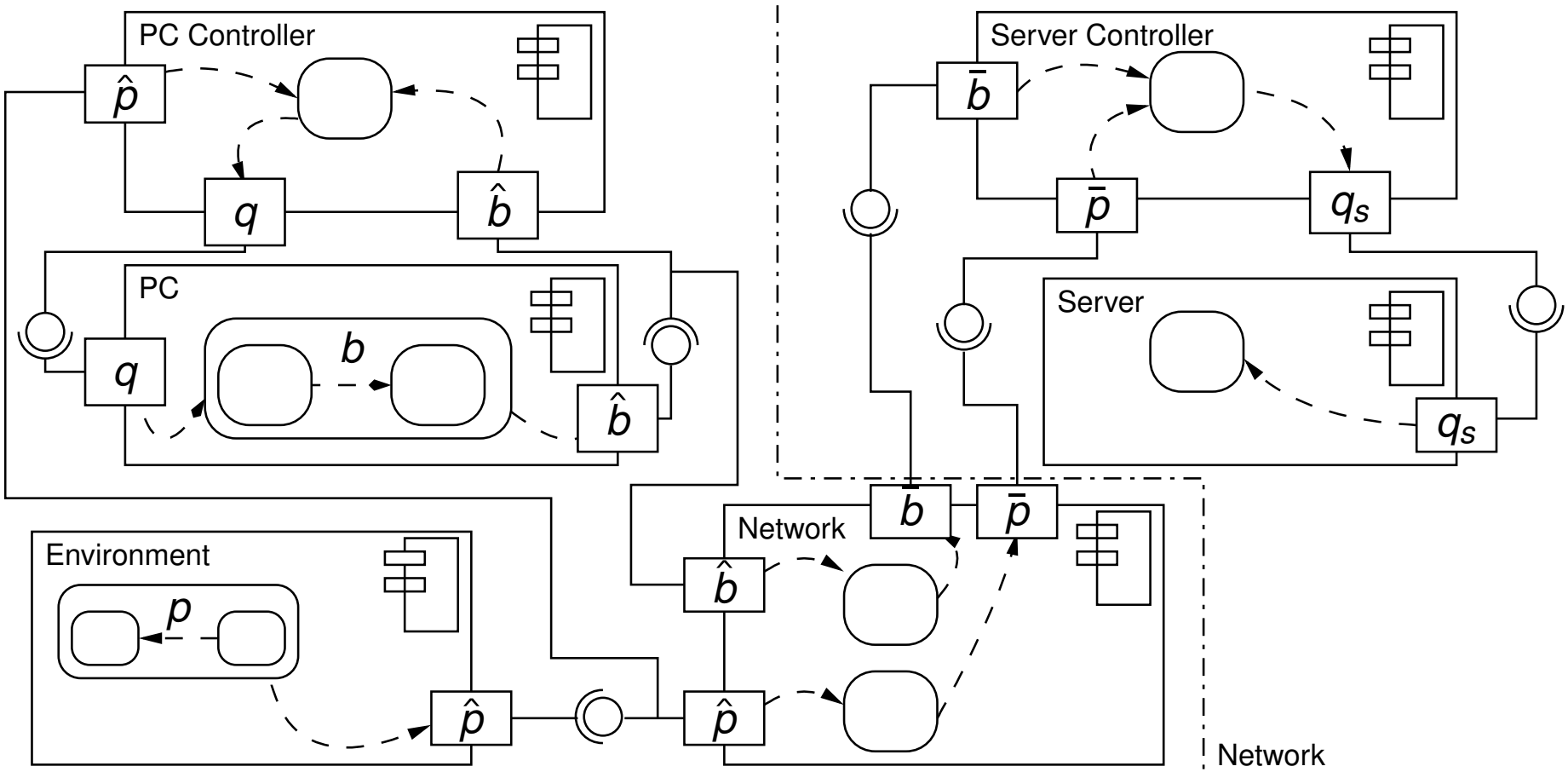
$\dot{p}(t) \leftarrow \mu(p(t))dt + \sigma(p(t))dB(t)$

(continuous variables: U = imported, Y = exported, X = internal; discrete variables: I = imported, O = exported, H = internal)

From modular hybrid automata to modular simulations

- What usage for BM?
  - test, validation, verification, ...
- Hybrid systems/automata = *declarative semantics*.
  Simulation models = *operational semantics*.
- Translate BM into simulation models
- Discrete EVent Systems (DEVS): de facto “standard” for modular discrete event simulation modeling and simulators.
  - Atomic models/simulators: core simulation engines, input and outpout of events.
  - Coupled models/simulators: composition and coordination (simulation clock).
  - Large variety of core simulation engines and distributed simulation implementations (*e.g.*, simulated versus real-time clock).
- Embed simulation models into software components.

Components, their embedded simulation engine and the exchanged events



Summary of the conceptual contributions

- A software component model with
  - stochastic hybrid systems BM,
  - modular modelling with HIOA/TIOA and
  - embedded DEVS-like simulation models,
  - and with composability at all levels

  that provide capabilities for:
  1. use cases and behavioral specification,
  2. model-in-the-loop simulation and validation,
  3. algorithms development and tuning,
  4. unit and integration testing through software-in-the-loop simulation,
  5. software verification and validation, deployment time system identification, control law synthesis and hardware-in-the-loop simulation for system validation and verification,
  6. run time verification,
  7. run time system self-adaptation.

Questions

Questions?

Related work

- Some related work about co-simulation
- But few with SIL or HIL
- And none aligning and integrating the software architecture with the simulation models.
- Two particularly interesting that adopt a software engineering point of view:
  1. Zoahib Iqbal *et al.*: systematic software testing with simulations, but only discrete.
  2. De Roo *et al.*: unit software testing with continuous simulations.

Perspectives

- Implementation in a Java distributed component model
- Integrate decision models for control law synthesis (*e.g.*, optimal stochastic control)
- Towards large-scale systems.