LIDS Paths Ahead: address by Albert Benveniste

1. What makes the following systems complex?

There are many reasons for the above systems to be considered complex. But the most essential one (to my mind) is they exhibit a huge number of levels of granularity.

Designers must, at the same time, keep track of the overall system and its functionalities and performance, and pay attention to the finest details. In particular, a number of feedback loops will have to be designed by control engineers and distributed computing infrastructure and communication infrastructure will be cautiously designed by systems and computer engineers. Control is involved both as low level feedback and at the level of larger subsystems or even in the overall system, and sometimes as part of the interaction with the encompassing *system of systems* (cf. air traffic control, which will be, in the future, decentralized in part to the aircrafts).

2. Opportunities for control in systems engineering

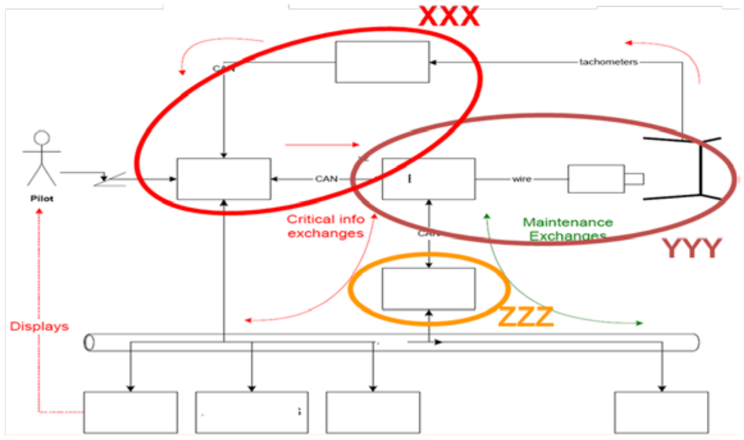
When I acted as a chair of IFAC Committee on Theory in the early 90's, I had the opportunity to perform a corporate audit in a large systems company. I found that there was a clear decline of control w.r.t. computer engineering in industry. Digital computing went embedded; hence software and architectures became the dominant issues. Simulink emerged that allowed physicists to handle modeling. And control engineers seemed confined to the back door.

This decade, however, has reversed the trend. The core of safety critical embedded software is about control. There are more functions, and more functions are distributed across the system, and across systems of systems. Seen from my dual background, I noticed that computer scientists got interested in control and even expressed respect.

There are clearly opportunities for control in systems engineering. But the nature of control problems and research is going to change.

3. Issues that must be addressed by control scientists

3.1 COMPONENTIZING, SUBCONTRACTING



The above diagram refers to a real system (part of a larger system) that consists of a feedback loop implemented over a distributed architecture. System requirements involve performance objectives of this feedback loop as a main issue. Still, this system has been subcontracted to three different suppliers as shown on the diagram. Would a sane control scientist consider doing this? However, large OEMs consider other reasons than the quality of control when deciding upon their business organization. Outsourcing again and again.

What does our community offer:

- For subcontracting feedback?
- For having control specifications as part of requirements?

  (Today we often see control heuristics formulated as rules, given as part of system requirements.) Should we build on top of MPC? Same for monitoring and protection (We often see rules, but rarely statements of statistical decision problems.)
- For component-based control design?

Did we devote enough effort:

- To make stability compose?
- To develop passivity based design?

3.2 DISTRIBUTING FEEDBACK

Computer scientists have devoted a great deal of effort to *System Architectures*:

- Fault tolerance, partitioning "distributed algorithms", Time-Triggered Architectures;
- With lots of theoretical effort: theory of concurrency, time-based distribution, and more;
- And effectiveness (despite problems remain).

Did the control community achieve the same?

- Did we develop control architectures?
- Did we think of how to blend control with other systems aspects? It is my understanding that it is more and more difficult to *isolate,* in complex systems, a control problem that a control engineer can cleanly state and address.
- Did we develop enough mathematical frameworks for distributed control? A great deal of effort is being currently devoted to the impact of delay, the cost of transmitting information, and the penalty of losing messages in distributed control systems. This is important. But actual distributed architectures used for control generally exhibit other artifacts that may, at times, be more important than the above ones. Not looking closely at computing infrastructures is not the way to go.
- Even though we may have nice mathematics, did we think of how to expose them to the designer? System designers often are trained to other tools and techniques than the ones control scientists use to deal with.

The comeback of control in the early 2000's must in large part be acknowledged to the success of Matlab-Simulink, *despite the dominance of aspects of computing over aspects of control.* In the 90's, this tool has put modeling and system design at the center of its objectives. It has largely won the competition against software dominant technologies in system engineering, for the referred sectors.

Can we replay Matlab-Simulink success story for the larger and more complex systems and systems of systems in the future?