# Facing the Challenge of Computer Science in the Industrial Applications of Control, a joint IEEE CSS – IFAC project

# SUMMARY OF ANALYSIS AND RECOMMENDATIONS

K.J. Aström, A. Benveniste*,
P.E. Caines, G. Cohen, L. Ljung

October 28, 1992

**Abstract**

Control, signal processing, and more generally "systems" industries ignore the boundaries we have in the academic world between control, signal processing, and computer sciences. Industries think of "hardware" (electronics or computers) and "software", making little distinction between algorithms development and implemention of them. Acting as a chairman of the IFAC Technical Committee on Theory for the triennium 1990-1993, Albert Benveniste proposed in the fall of 1989 this project to investigate some fundamental questions raised by the above mentioned facts. Since CDC'90 this has been approved as a *joint IEEE/CSS-IFAC project* managed by the above listed group of people. A detailed progress report of the project has been written in March 20, 1991, followed by an a brief update in October 10, 1991. This is a summary of the conclusions of the report. Additional detailed information on the project is found in the bibliography.

*A. Benveniste is the chairman of the project, he is with INRIA-IRISA, Campus de Beaulieu, 35042 RENNES Cedex, FRANCE; e-mail benveniste@irisa.fr

# Contents

1 EMERGING IDEAS 4
1.1 Process control, signal processing, and related industries, are Concurrent Engineering activities . . . 4
1.2 Modern progresses in Advanced Control reduce the relative cost of control design within the whole process of systems design 5
1.3 Application software development frequently builds up to more than 50% of the total cost of engineering in large systems development and is thus critical for engineering productivity . . 5
1.4 Complex systems modelling on one hand, and designing heuristics of high combinatorial complexity on the other hand, require good CASE tools . . . 6
1.5 A major recognized bottleneck is the task of collecting data (or more broadly knowledge) that are reliable and accurate for complex systems . . . 6
1.6 Algorithm design and their real-time programming should not be considered as mutually foreign activities any more . . . . . 6
1.7 New mathematical frameworks are needed to cope with the combinatorial complexity inherent in large systems . . . . . . 7
1.8 The background of control engineers has to be widened to cover major aspects of systems engineering, this includes real-time programming and software engineering . . . 9

2 RECOMMENDATIONS 9
2.1 Addressed to academic research community . . . 9
2.1.1 The control science community should be aware of the niche of current classical control science in industrial applications of control . . . 9
2.1.2 The control science community should note other aspects of industrial control engineering . . . 9
2.1.3 Providing new mathematical frameworks for dynamical systems to reduce the amount of heuristics, is among the best ways to increase productivity in application software development . . . 10
2.1.4 The control science community should learn from computer science how to handle complexity via modularity 10
2.2 Remarks for the industrial community . . . 10

2.2.1 The industry should not deduce from the current situation that above PID's and hardware only exists AI . 10
2.2.2 Excessive turnover may be the bad, but true reason for not investing on formal methods . . . . . . . . . . 11
2.2.3 Technical management in industry should pay more attention in providing the research community with global views on their main issues . . . . . . . . . . . . 11
2.3 Remarks for government organizations . . . . . . . . . . . . . 11
2.3.1 "Real-time" systems design should be recognized as a research area and supported accordingly . . . . . . . . 11
2.3.2 More generally, any effort towards opening boundaries between adjacent disciplines relevant to automation or control industry should be undertaken . . . . . . . . . 12
2.3.3 Joint academic/industrial efforts at the "systems design" level should be encouraged . . . . . . . . . . . . 12
2.4 Address to Education . . . . . . . . . . . . . . . . . . . . . . . 12
2.4.1 Control engineering courses should involve real-time programming and software engineering . . . . . . . . . 12
2.4.2 Computer engineering courses should involve real-time as an important topic with its dynamical aspects . . . 13

**A Other major deliveries of the project** **13**
A.1 Meetings with Industrialists . . . . . . . . . . . . . . . . . . . . 13
A.2 Joint Automatica — IEEE-AC Special Issue . . . . . . . . . . 14

# 1 EMERGING IDEAS

Here follow some opinions emerging from the project. These were in part collected from the reports by members of the project ([1], section 1.3). Since the advance report, these ideas have been discussed and further elaborated. The authors of these ideas are acknowledged on the fly and references given when available, where you will find detailed arguments supporting the claims to follow.

## 1.1 Process control, signal processing, and related industries, are Concurrent Engineering activities

The whole process of "automation" consists of the following items (A. Benveniste [1, 3]):

**Systems analysis and customers requirements** is an important item: informal interactions between customer and designer occur. For them to be handled efficiently, some feedback is required from other items (design issues for instance should not be ignored).

**Systems specifications elaboration and implementation** is in fact a permanent activity within the whole process: each stage of systems design consists in transforming specifications into some "effective implementation" of these, which in turn may be the specification for a subsequent stage (for instance control loop design, and then computer implementation of it). This item uses information from other ones, but also influences them. Among the various stages "specification → implementation", one may for instance encounter

- Planning design, i.e. defining medium level requirements from higher level ones such as availability, safety, performance... Results from this stage are typically: design specifications for control loops (robustness, performance,...); sensor, actuator, and process fault detection and associated predictive maintenance and process reconfiguration; monitoring and diagnostics; software and hardware safety and performance requirements; interfaces to the human operator; etc...
- Control design.
- On-line monitoring, fault detection, and systems reconfiguration.
- Diagnostics.

- Software, communication, and hardware architecture design.
- Setting the human operator in the system.
- Application software development and implementation.

**Verification and/or certification** is a very important item in complex systems.

## 1.2 Modern progresses in Advanced Control reduce the relative cost of control design within the whole process of systems design

Is has been found that (K.J. Aström, A. Benveniste [1]), thanks to the recent availability of modern control techniques and associated CACSD[1] tools, building control laws from available specifications and models has a relative cost which is frequently less than a percent of that of the whole systems design. Designing good controls is a sine qua non condition for good systems design, but, due to the little percentage of this activity within the total cost, major savings in productivity are expected to be found elsewhere[2].

## 1.3 Application software development frequently builds up to more than 50% of the total cost of engineering in large systems development and is thus critical for engineering productivity

Application software is the place where major savings are expected (A. Benveniste [1, 3]). Among the above listed items "specification → implementation", a few of them rely on mathematical founded frameworks. This is for instance the case for control design, especially when models of the plant are used rather than rules of thumb for PID tuning. But most of application software is implemented as poorly structured heuristics, which may result in up to 1 million lines of ADA code for large, but not exceptional, systems. In the same way, typical distributed control system have a similar size but the code is often written in several languages. (K.J. Aström, [1]).

[1]CACSD: Computer Aided Control Systems Design

[2]Comment from E.J. Davison, IFAC Technical Board Vice-Chairman : *This is not nessarily true. For instance Canada is putting up a 3rd generation spacecraft called MSAT. The limiting factor in the design was the control problem (everything else was minor). Most of the effort/time has been directly related to this control problem. Another example is the Canadian space-arm for the US space station - 99% of the problem is control (in the standard sense). See [7] for related information.*

## 1.4 Complex systems modelling on one hand, and designing heuristics of high combinatorial complexity on the other hand, require good CASE tools

We[3] just discussed the case of real-time programming of loosely structured heuristics. But simulators of complex systems often require handling models or descriptions that are *hybrid* in nature: PDE's or ODE's can be mixed with dynamical systems of discrete event type. While popular CACSD tools allow the user to describe and simulate ODE's, hybrid models are seldomly considered, mainly due to the lack of proper mathematical framework which would be needed to support such tools. To summarize, the current status of both areas calls for aids to complex systems handling (K.J. Astrom [1]): this is currently achieved by relying on software technologies such as object oriented programming, knowledge based systems, intelligent data bases, expert systems, and a great deal of effort is devoted to progress in these areas in process control industries with support from government organizations.

## 1.5 A major recognized bottleneck is the task of collecting data (or more broadly knowledge) that are reliable and accurate for complex systems

Data or knowledge is used both for building models or directly within heuristics. It has been recognized as an advantage of the expert systems technology that attention is paid from the very beginning to these aspects (G. Cohen [1, 4], P.E. Caines [1]); furthermore, while using such a method, plant and associated system are frequently condidered as a whole and this is an advantage. This is maybe in contrast to the more traditional control engineer point of view, where aspects that seem at a first glance irrelevant to control design are left aside, but might in turn have an unfavourable impact on other stages of the systems design, e.g., systems architecture or maintenance.

## 1.6 Algorithm design and their real-time programming should not be considered as mutually foreign activities any more

It has been recognized (A. Benveniste [1, 3]) that, like other areas of modern computer science — especially when parallelism and concurrency are involved —, real-time programming has to move quickly towards a well-

[3]CASE: Computer Aided Software Engineering

founded scientific activity. Formal methods which provide proofs, verifications, and possibly synthesis from specifications, are required for large software certification and safety requirements. Today, formal methods from advanced computer science mainly handle discrete aspects (synchronization and logic) of real-time systems. But real-time programs are dynamical systems and their theory shares many fundamental aspects with system theories that are developed in control science. Thus real-time programming should not be considered as a foreign research activity by hard core control scientists: it is possible for them to get technical interest in it, and even possibly to provide contributions.

## 1.7 New mathematical frameworks are needed to cope with the combinatorial complexity inherent in large systems

It has been found that, in most cases, the largest part of application software (say, about 90%) is devoted to handling the unstructured aspects of processing. It consists of loosely structured heuristics and is the major cause of the *"software errors"*, known as the project manager's nightmare. Software errors may result from both a flawed design of the heuristics and/or their flawed programming. In real-time systems development, the occurrence of such heuristics has been the main motivation for relying on modern software engineering technologies such as object oriented programming, expert systems, etc... On the other hand, it has been noticed that *relying on mathematical frameworks typically result in software that is of one magnitude order smaller and is much easier to validate and debug* (A. Benveniste [1, 3]). Unfortunately, most of the currently used mathematical frameworks in control design, real-time information processing, and related tasks do not handle discrete event aspects satisfactorily. New mathematical frameworks are needed with the following features:

**Modularity:** such frameworks should provide us with models that are inherently modular in nature: the key to complex systems handling is the ability to build complex models from small primitives and a few assembly rules. The traditional background of the control scientist, which is linear systems oriented, does not give any account of modularity. Lack of such a prospective has probably been the main reason for "large scale systems theory" having little success in practice[4]. In

[4] Comment from E.J. Davison, IFAC Technical Board Vice-Chairman : *There are some classes of industrial systems, where large system theory is very effectively applied. Exam-*

contrast, coping with modularity is the everyday life of computer scientists: *modularity is one major feature of programming.*

**Hybrid nature:** such frameworks should be *hybrid* in nature, i.e., they should encompass diverse aspects of systems (continuous, discrete, uncertain,...) and their interconnections. It has been a long standing way of thinking in control science that a model class should be developed together with its associated formal theory. For instance, linear systems with linear control design, etc... Today, there is little hope to develop frameworks that are at the same time *universal* in the above sense, and equipped with an associated full power formal system. *One should better distinguish between descriptive power — which should be "maximum" —, and formal manipulations — which have to remain partial in order to be effective —.* The interest of having a universal framework just for description is to provide a common framework to support various formal systems, each of them handling different features of the general model. We know that switching from one formalism to another one within the design process is one of the key difficulties.

**With associated formal systems:** the power of classical and modern linear control is the richness of linear systems theory, which makes it possible to *perform directly a very exact and explicit design of linear controllers from high level specifications.* It is generally accepted that formal theories are the key to breakthroughs in productivity. This is where object oriented programming and related technologies fail: they allow to handle complexity in a nicely ordered fashion, but do not remove possible complexity.

The "programming" nature of "fuzzy control" is likely the major reason for its success: "fuzzy" rules (i.e., programming) help to handle the unstructured part of system as well as regulation. In contrast, the issue of combinatorial complexity is not handled per se via connectionnist approaches.

*ples are given now. The control of MSAT [7] is a decentralized/centralized configuration based on a model with* 11 *outputs,* 9 *inputs, of order* $\gg 100$. *The control of large power facilities is a success story (Ontario Hydro uses all aspects of large scale system theory extensively — e.g. the basic model of the hydro grid system in Ontario is of order approximately* 2000 *with hundreds of inputs/outputs). The control of chemical plants again uses such large scale system techniques extensively ... etc.*

### 1.8 The background of control engineers has to be widened to cover major aspects of systems engineering, this includes real-time programming and software engineering

In analysing the way teams are built in industry, we have found the following (K.J. Aström [1], A. Benveniste [1, 3]). Engineers with traditional background in control or signal processing usually rely on a way of thinking which is close to analog systems. They are not trained to master modern large software systems such as one encounters in real-time programming. A resulting consequence has been sometimes that control oriented companies have found it more profitable to hire engineers with hard core computer science background rather than a control background. This results in having complex control systems developed by engineers who pay a great attention to human interfaces and data bases, but who are hardly able of managing the question of dynamics properly, not to speak of sophisticated control design. Human interfaces are among the major difficulties in complex systems, but proper design of safe control systems should not be underestimated.

# 2 RECOMMENDATIONS

## 2.1 Addressed to academic research community

### 2.1.1 The control science community should be aware of the niche of current classical control science in industrial applications of control

As has been said, in most applications, control design is responsible for a small percentage of the total engineering cost. This explains why control industries expect major savings in cost from improving software development for the part of processing which is non mathematical in nature. Control design is however often critical, i.e., a system may fail if a poor control design is implemented.

### 2.1.2 The control science community should note other aspects of industrial control engineering

Studies on supervision, monitoring, failure handling, diagnostics, and system reconfiguration is now (slowly) being considered as a part of control science. Further research efforts should be developed on these aspects, however. Discrete Event Dynamical System theories are now emerging as a new area of concern: it is not clear, however, that researchers are really aware of which issues in the industry may be relevant to this research area. In particular, issues of complexity, modularity, and availability of an effective formalism for complex system description, are perhaps not well appreciated. Crossfertilization with computer science is obviously needed here: real-time has also been recognized recently as a deep, difficult, and hot topic in computer science, see for instance [6]. It should be borne in mind that real-time systems and programs are dynamical systems.

### 2.1.3 Providing new mathematical frameworks for dynamical systems to reduce the amount of heuristics, is among the best ways to increase productivity in application software development

Control scientists always knew intuitively the importance of mathematical approaches to control: modern control design is an engineering activity (intuition and practical experience are needed) which is strongly supported by mathematical theory of control. Thanks to this mathematical theory, actual controls can be designed from high level specifications, e.g., robustness and performance criteria. *This desirable situation has to be translated to other aspects of systems design.* This is for instance what computer scientists do in promoting the use of formal methodologies in real-time systems development and programming. Further effort is needed therefore to enlarge the area of formal methods to handle new aspects of dynamical systems.

### 2.1.4 The control science community should learn from computer science how to handle complexity via modularity

This is perhaps the most significant idea for the control community. Considering modularity from the very beginning in developing a theory is definitely a new way of thinking.

## 2.2 Remarks for the industrial community

### 2.2.1 The industry should not deduce from the current situation that above PID's and hardware only exists AI

Coping with combinatorial complexity and the lack of relevant or cost effective mathematical modelling is often performed by relying on AI oriented approaches, with emphasis on data bases, objects, rules, and inference engines.

But, despite recent efforts, such (useful) notions are inherently non dynamical in nature, their use in real-time systems raises some difficult problems, and they do not remove complexity but just handle it better. Cutting costs may be better achieved by switching whenever possible to *formal* methods of systems design, which may originate equally well from control or computer science. Models with associated formal methods encompass prior knowledge much more efficiently than heuristics: modelling becomes the main part of specification, then implementation follows based on associated formal design techniques, and formal "proof" systems help verification.

#### 2.2.2 Excessive turnover may be the bad, but true reason for not investing on formal methods

Formal methods, both from control or computer science, require well trained and skilled people. Modelling and control design is a difficult but highly profitable activity provided appropriate people are available. Similarly, relying on formal methods of real-time systems development requires trained and skilled people. Excessive turnover may prevent having such people available. Heuristics in turn are easier to handle with little training.

#### 2.2.3 Technical management in industry should pay more attention in providing the research community with global views on their main issues

Concurrent Engineering is currently the magic rule. But Concurrent Engineering implies globality. In contrast, industrialists usually interact with academic researchers on very narrow topics for they feel a specialist should focus. But global problems require global views, and such global views should be shared with the academic community in some way. This project has been the very first opportunity for its authors to have such an interaction, and this has been found highly inspiring.

### 2.3 Remarks for government organizations

#### 2.3.1 "Real-time" systems design should be recognized as a research area and supported accordingly

This would break the otherwise strong boundary between control and computer sciences. Such a "real-time" topic would be first built on discrete aspects of dynamical systems, e.g., automata and related formalisms with associated model checking techniques, together with discrete event dynamical systems as studied in control science. A valuable connection about dynamics and modularity would result, and this is the key objective. Then extensions of this topic may occur towards handling problems with more "hybrid" aspects: this is where control and computer people have to cooperate. Real-time should not be a new closed lobby with its journals, conferences, etc, but should remain open to existing communities with their backgrounds and proper vehicles. We may indicate at this point that a joint industrial/academic research and development effort is already organized in France along these lines which is conducted by CNRS and INRIA.

#### 2.3.2 More generally, any effort towards opening boundaries between adjacent disciplines relevant to automation or control industry should be undertaken

This is worth to be mentioned, since the natural way of doing is instead to create new areas with well recognizable boundaries so as to attract attention and then money. As instances we may mention "connexionnism" which has in fact a large overlapping with adaptive systems in control or signal processing areas, and also "intelligent control" which tries to identify itself as a separate discipline. Opening should be encouraged instead to promote global views on the issues (remember our discussion on Concurrent Engineering).

#### 2.3.3 Joint academic/industrial efforts at the "systems design" level should be encouraged

As discussed above, this would help sharing the needed global views on critical issues between research and industry.

### 2.4 Address to Education

#### 2.4.1 Control engineering courses should involve real-time programming and software engineering

Proper teaching should be provided. Laboratory mini-plants are often encountered and used in control courses. Opportunity should be taken of using advanced computer science methods for real-time systems development in conjunction with advanced control design.

#### 2.4.2 Computer engineering courses should involve real-time as an important topic with its dynamical aspects

This does not seem to be the case today. Laboratory mini-plants available in control courses may be the opportunity to exercise computer science and control engineering students on programming in interaction with physical systems.

# Appendix

## A Other major deliveries of the project

They are briefly listed and discussed below. These items provided the material that served as a basis for the present executive summary.

### A.1 Meetings with Industrialists

The following meetings have been organized, where the issues relevant to the project have been discussed:

- Fisher Control (control systems supplier), by K.J. Aström,
- Thomson-CSF-SDC, technical management (radar systems, air traffic control), by A. Benveniste,
- GEC-Alsthom, technical management of rail transportation systems (subway systems), by A. Benveniste,
- Siemens-AG-Automatisierung, technical management (rolling mill, expert systems for project management), by A. Benveniste,
- Hydro Quebec, power supply, expert systems division, by P.E. Caines,
- EDF-DER, nuclear power plant diagnosis system, by G. Cohen.

Approved reports of some of these meetings were provided in the Advance Report of the project [1]. The three reports by A. Benveniste and the one by G. Cohen have been published be *IEEE Control Systems Magazine* [2].

## A.2 Joint Automatica — IEEE-AC Special Issue

A joint special issue of *Automatica* and *IEEE Trans. on Automatic Control* is planed for spring 1993, with K.J. Aström and A. Benveniste acting as guest editors. The purpose of this special issue is twofold, namely

- illustrate that mathematical methods of control are useful in those areas where heuristics are mostly used today,
- attract good application papers from industry.

30 papers have been submitted, among them approximately 80% involve participants from industry. 4 submitted papers, however, involve industrial authors only. A large part of the submitted papers mainly report on classical control applications, with little consideration for the particular topic of the project ; some of these contributions, however, are of very high quality and will be selected for publication. To summarize, few papers are really addressing some of the issues discussed in the project, the overall impression is that the control community is not paying enough attention to these questions today ; note that purely industrial papers appear much more attractive from this point of view. These submissions are currently under processing, and publication is expected by the spring of 1993. It is likely that this special issue will contain several papers of outstanding quality.

# Bibliography

[1] K.J. Aström, A. Benveniste (chairman), P.E. Caines, G. Cohen, L. Ljung, P. Varaiya, "Facing the challenge of computer science in the industrial applications of control, a joint IEEE-IFAC project, advance report", IRISA, 1991, available upon request from A. Benveniste.

[2] K.J. Aström, A. Benveniste (chairman), P.E. Caines, G. Cohen, L. Ljung, P. Varaiya, "Facing the challenge of computer science in the industrial applications of control", *IEEE Control Systems Magazine*, vol 11, No 4, 86, June 1991.

[3] A. Benveniste, "Meetings held with Tomson-CSF, GEC-Alsthom, and Siemens-AG", *IEEE Control Systems Magazine*, vol 11, No 4, 86-92, June 1991.

[4] G. COHEN, "Meeting with Electricité de France", *IEEE Control Systems,* vol 11, No 4, 92-94, June 1991.

[5] K.J. ASTRÖM, A. BENVENISTE, Eds., *Meeting the Challenge of Computer Science in the Industrial Applications of Control,* joint special issue of *Automatica* and *IEEE Trans. on Automatic Control,* to appear in spring 1993.

[6] *Another look at real-time programming,* special section of the *Proc. of the IEEE,* vol. 79 n°9, Sept. 1991.

[7] DAVISON E.J., "Controller Design for MSAT- A Third Generation Spacecraft", 1990 IEEE CDC, Hawaii, Dec 1990, pp 2047-2053.