# Chapter 4 – COMPLEXITY, AUTOMATION AND AUTONOMY

## 4.1 INTRODUCTION

This chapter addresses complexity, automation and autonomy. The advent of portable and powerful micro-processors has allowed control systems to become more sophisticated and complex. Integrating multiple control systems together can yield highly automated systems that exhibit autonomous capabilities. An autonomous system can make decisions without human guidance, thus the long term goal of autonomous control is to impart human-like capabilities. To achieve human-like capabilities, an autonomous system may require the ability to learn from experiences.

For both single vehicles and multiple vehicle systems, the design of the system needs a good grasp of the capability that is required to meet the desired performance requirements. For autonomous systems this requires a precise description of the environment in which the system must operate, together with the desired behaviours of the autonomous system. This in turn demands precise definition of the behaviour that the system is required to exhibit and under what conditions. Hence some time must be spent in defining autonomous behaviour and the complexity of both the environment and the complex system structure that is usually associated with autonomous systems.

This chapter discusses the following important topics such as the complexity of systems, man-machine interfaces yielding higher levels of automation, autonomous vehicles and associated problems, and artificial intelligence.

## 4.2 COMPLEXITY

As the control systems and computers are becoming more and more sophisticated and complex, they require a high degree of reliability and maintainability and they must have fault accommodation in order to operate successfully over long periods of time. Reconfigurable controller has to achieve the following goals:

1) Keep the system performance within acceptable boundaries during operation;
2) Increase the performance of the process; and
3) Achieve the goal for fault accommodation.

Reconfigurable control(ler) is a critical technology to detect the fault and recover the functionality of the faulty system as same as that of the nominal system. Various methods are used for reconfigurable control to cover the requirements of different applications. The behavior of the reconfigurable control depends upon whether the approach is passive or active. Such control ideas have been implemented on a variety of military and commercial applications in last two decades to accommodate faults, for example on flight control systems on space technology in and on unmanned underwater vehicles.

The steady increase in complexity of modern systems and infrastructures has placed strong demands on the requirements for control systems technologies. New advances in computing technology, microelectronics and intelligent devices (MEMS, self-validating sensor and architectures) have facilitated the development of powerful scientific and engineering methods in control. Almost all embedded systems now involve Control particularly at the high levels of embedding. Control technologies go significantly beyond the scope of traditional or classical solutions and now have the capabilities to address new challenges arising from large-scale networking, distributed service provision and sophisticated safety-critical applications. Complex Systems emerge in many disciplines and domains and have many interpretations, implications and problems associated with them. In addition to real-time requirements

these systems have to be dependable which most often requires fault-tolerance. It is well known that the development of fault-tolerant real-time systems is a very challenging topic.

An interesting characteristic of complexity is that higher levels of complexity tend to result in increased involvement of the human operator. Humans are difficult to model and their interaction with the plant, process or vehicle induces additional complexity and uncertainty. An interesting characteristic of complexity is that higher levels of complexity tend to result in increased involvement of the human operator. As systems increase in complexity this is becoming more important as more "pervasive" system tools become available. In recent years Automatic Control has been branching out in the directions of Software Engineering and Cognitive Science, giving rise to a discipline that can now be called "Embedded Cognitive Control", as shown in the figure below.

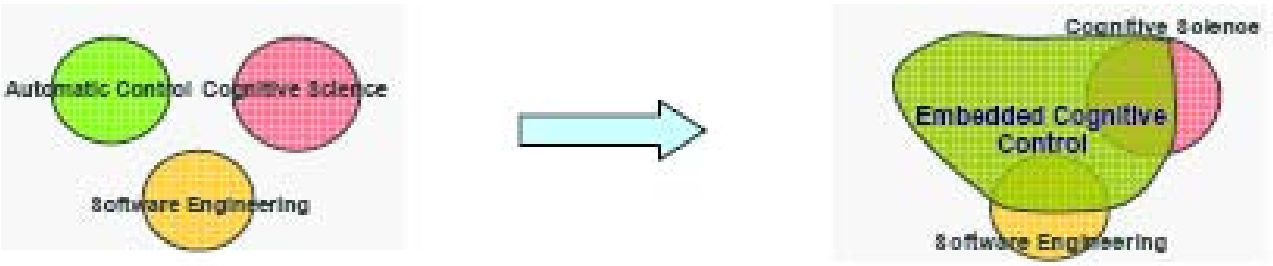


**Figure 4.1: Embedded Cognitive Control Engineering Discipline.**

The "human self" can be represented much more in terms of reasoning, as a cognitive agent and as a "digital system" (as a part of the embedded computing structure of the system, etc.). The human self can even be regarded as an "executable" system. It is apparent that as systems increase in complexity these emerging issues and concepts involving the role of the human operator become more and more important. In some aspects humans will be dominant in the sense that there are some abilities that cannot be replaced by an automated system (e.g., in the analysis of many decision situations, where a comprehensive analysis of diversified elements is not possible to be formalized), want to keep decision-making sovereignty while using a decision support system for analysing the problem. On the other hand in some situations a dynamic and/or complexity of system may require automatic control. New activities in advanced control face the challenge of the complexity (characterized by size, structure, irreducible uncertainty, risk, diversified performance measures, etc.) of modern engineering and business systems and enterprises, spanning bio-technology, information technology, space and aeronautics, vehicle systems, process and manufacturing systems, life sciences, the economy, etc.

Lui Sha, from the University of Illinois at Urbana-Champaign, in his article entitled:" Using Simplicity to Control Complexity. He has indicated his "call this approach using simplicity to control complexity". Computational complexity is modeled as the number of steps to complete the computation. Likewise, we can view logical complexity as the number of steps to verify correctness. Logical complexity is a function of the number of cases (states) that the verification or testing process must handle. A program can have different logical and computational complexities. For example, compared to quick sort, bubble sort has lower logical complexity but higher computational complexity. The wisdom of "Keep it simple" is self evident. We know that simplicity leads to reliability, so why is keeping systems simple so difficult? One reason involves the pursuit of features and performance. Gaining higher performance and functionality requires that we push the technology envelope and stretch the limits of our understanding. Given the competition on features, functionality, and performance, the production and usage of complex software components (either custom or COTS) are unavoidable in most applications. Useful but unessential features cause most of the complexity. Avoiding complex software components is not practical in most applications. We need an approach that lets us safely exploit the features the applications provide.

The notion of using simplicity to control complexity ensures the critical properties. It provides us with a "safety net" that lets us safely exploit the features that complex software components offer.

"Control of Complex Systems" by Karl J. Aström et. al has extensive analysis of complex systems. This book is an example of the types of approach that European researchers are using to tackle problems derived from systems' complexity. It has grown out of activities in the Control of Complex Systems (COSY) research program the goals of which are to promote multi-disciplinary activity leading to a deeper understanding and further development of control technologies for complex systems and if possible, to develop the theory underlying such systems. The material in this book represents a selection of the results of the COSY program and is organised as a collection of essays of varying nature: surveys of essential areas, discussion of specific problems, case studies, and benchmark problems. A selection of the results of the Control of Complex Systems research program, COSY, and is organized as a collection of essays. Topics include modeling and complex physical systems, control design, learning control, satellite attitude control, and passivity-based fault identification and fault tolerance.

## 4.3 MAN-MACHINE SYSTEMS

The knowledge about human behavior in man-machine system design for the vehicle with high level of autonomy can be used for:

- Design of operator's station or other means in cabin of manned vehicles for monitoring of such vehicles.
- Design of intelligent systems by use of knowledge on human operator behavior.

These aspects of knowledge are discussed below.

### 4.3.1 Supervisory Control in Man-Machine System

The level of vehicle autonomy influences on role of a man in control of the vehicles. For the lowest level a human-operator acts as an active controller in man-machine system.

In such manual control tasks human-operator each moment reacts actively on perceived stimulas (visual cues, vestibular cues, etc.) by deflecting a manipulator for transition of control signals to the vehicle. The brief analysis of pilot behavior in manual control and usage of this knowledge for integration of human-operator and vehicle is given in Chapter 2. The scheme characterized the human-operator activity in man-machine system for manual control is given on Figure 2.1. The increase of level of autonomy changes the role of human-operator. He acts in that case as a supervisor. There is supposed that for case of supervisory control the task is accomplished by automatically and human-operator ability is to check its fulfillment and to act as a monitor. It means that operator ability is the monitoring of the control process. His active participation will take place in case when the task (mission) will be changed or system performance (for example, accuracy) will approach or increase the requirements. In principle pilot can fulfill the manual control and monitoring tasks simultaneously. The pilot workload influences considerably on distribution of pilot activity between control and monitoring. The decrease of pilot workload index will lead to increase of human-operator activity as a monitor. In other case, he has to be more active control element in control loop. The Figure 4.2 reflects man-machine system in supervisory control. The main design problem in supervisory control is optimization of interfaces provided minimum human errors in recognition of appearance and/or development of accident or failures with minimum operator workload.

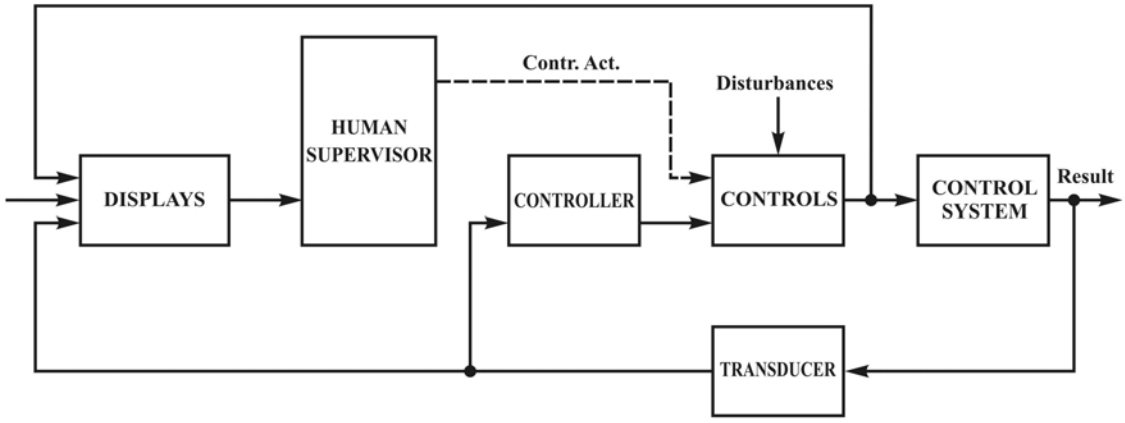


**Figure 4.2: Man-Machine System in Supervisory Control.**

The solution of this problem requires to develop the models of operator as a supervisor, to define the workload indexes (WI), and to select the secondary task (for evaluation of WI).

The model of human operator as a supervisor can be presented with help of the scheme shown on Figure 4.3.

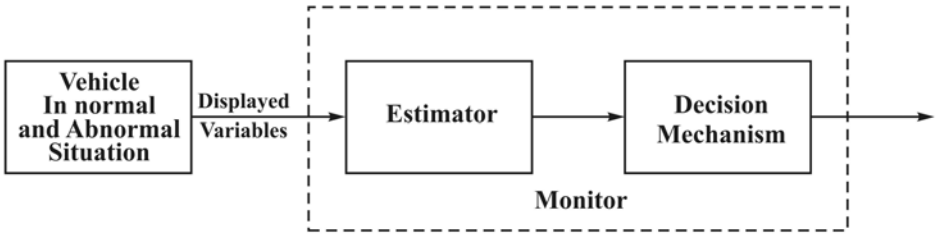


**Figure 4.3: Human Operator Schematic.**

It consists of two elements: a linear estimator and a decision mechanism. As a linear estimator they use Kalman filter estimates the state variables $\hat{x}(t)$ and the measurements $\hat{y}(t)$ as well as the measurement error (residual) $\varepsilon(t)$, where $\varepsilon(t) = y(t) - \hat{y}(t)$. The model takes into account the observation noise. For a number of instruments or metrics observed by operator the model takes into account the effect of sharing attention. If the operator observes more then one instrument his observation noise for each observation increases by a constant factor [Levison W.H., Elkind, J. and Ward J.,1971] and [Baron S., Kleinman D., Levison W., 1969]. The level of noise is inversely proportional to the fraction of attention that he spends monitoring that specific instrument.

Decision making mechanism can be based by different way. One of them proposed in [Gai E., Curry R., 1975] is based on sequential analysis. The last one uses the hood-like ratio $l(m)$ as a decision function after $m$ observations. The two criteria levels, A and B are chosen, and the decision rule is given by the sequence:

if $l(m) \geq A$ choose failure

if $l(m) \leq B$ choose normal

if $B < l(m) < A$ take another observation.

Where A and B are determined by desired probability of false alarm P(FA) and probability of miss P(MS) follows:

A=1-P(MS)/P(FA)

B=P(MS)/1-P(FA)

One of the problems in definition of the best way for presentation of symbols and their location, indicator design, is the low sensitivity of the criteria used for solution of the problem (for example, detection time) to the workload index used in researches. As an example, the dependence of detected time as a function of normalized workload is shown on Figure 4.4.

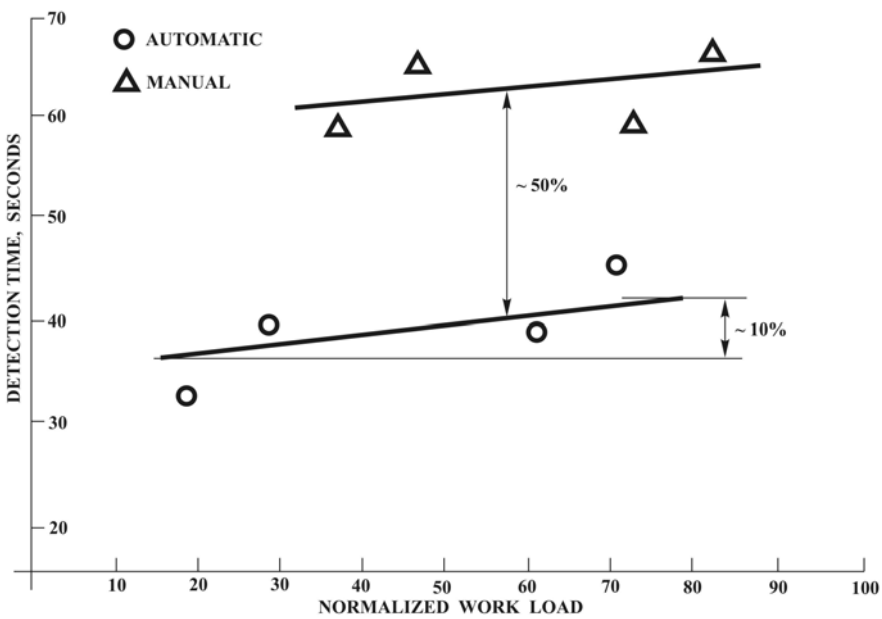


**Figure 4.4: Detected Time vs. Normalized Workload.**

The research [Ephrath A.R., 1975] was dedicated to the estimation of detection time required to define a failure of one of the instruments. The research was fulfilled for manual control when pilot uses the observed information actively acting as controller and for automatic regime when pilot's role was a monitor. The research was fulfilled for different pilot workload, associated with fulfillment of the secondary (no control) task. There is seen that in average the increase of normalized workload in its wide range causes the insignificant increase of detection time (up to 10% only). As for manual control the detection time was 50% higher in comparison with automatic control.

This result leads to idea to use the manual control task as a secondary task for definition of detection time. It might be more sensitive test for solution of applied design problem. As an example the first order unstable plant $\left(W_c = \frac{\lambda}{p-\lambda}\right)$ with different level of unstable root can be used for the secondary task and the value $\bar{\lambda} = \frac{\lambda_s}{\lambda_c}$ as a WI (here $\lambda_c$ maximum value of root achieved in single loop manual control task), $\lambda_s$ – value $\lambda_s$ corresponded to the specific level of error.

### 4.3.2 Adaptive and Intelligent Systems Grounding on Human Operator Experience as a Basis for System-Level Integration of Control

#### 4.3.2.1 Intelligent Control Systems as a Reflection of Experienced Human Pilot Skills

The intelligent control systems are the systems having an ability to emulate human capabilities, such as planning, learning and adaptation**.** Intelligent control systems may be considered as a reflection of experienced human pilot (operator) skills in some artificial media. We would like to generate a control system, which will be comparable with a test pilot from the point of view of control skills.

It is necessary to define what is meant above by intelligent, a term used here to refer to a specific class of problem solving. The technical committee on intelligent control of the IEEE Control Systems Society has defined the general characteristics of intelligent control systems as having an ability to emulate human capabilities, such as planning, learning and adaptation [Linkens, D.A. and others, 1996]**.** Learning and adaptation especially are essential characteristics of intelligent control systems and, while adaptation does not necessarily require a learning ability for systems to be able to cope with a wide variety of unexpected changes and environments, learning is invariably required.

As it is shown below, intelligent control systems must allow solving of control tasks, which are too difficult or unsolvable by means of traditional control techniques.

#### 4.3.2.2 Control Problems for Advanced Manned and Unmanned Aircraft – General Description

There are many tasks associated with flight control for modern and advanced aircraft including unmanned aerial vehicles (UAVs), which are not solved (or solved very unsatisfactorily) with traditional tools. It has been recognized in recent years, that realization of more flexible and effective control systems requires to combine other elements, such as logic, reasoning, heuristics etc., with the algorithmic techniques provided by conventional control theory, and such systems are known as intelligent control systems.

Evidently uncompleted list of such tasks includes: flight control for agile and post-stall aircraft; flight control in complicated cases (influence of atmospheric turbulence, wind shear, flight and landing in complicated weather conditions, landing on aircraft carrier, refueling etc.); flight control with possibility of sharp or smooth changing of mathematical models for aircraft motion caused corresponding changes in vehicle shape and parameters (dropping internal and/or external stores, damages of aircraft, engines, avionics – sharp foreseen and/or unforeseen changes; maintenance of aircraft with taking into account smooth changes in its shape and parameters – icing of planes, wear of aircraft systems, expenditure of fuel from tanks etc.); and flight management and control in case of group of vehicles. These tasks can be characterized with such features as:

- Wide range of conditions (flight modes, motion parameters, external disturbances etc.) needed to take into account;
- Presence of many uncertainty factors belonging to various classes;
- Essential nonlinearity of aircraft characteristics;
- Essentially non-steady nature of processes realized with aircraft as controlled system;
- Broad using of supercritical flight regimes to implement aircraft supermaneuverability;
- Possibility of many abnormal modes caused by various kind of failures and damages in aircraft structure and equipment as well as with some external effects;
- Unmanned advanced aircraft autonomy challenge; and
- Necessity of collective operations for vehicles (cooperative actions, collision avoidance etc.) including a case of dissimilar vehicles.

#### 4.3.2.3 Semi-Soft Computing as a Basis for Implementation of Intelligent Control Systems

Research in the intelligent control field is based mainly on soft computing methods and tools as well as on extensions of these ones. It can be divided on three levels of the methods and tools needed to solve problems mentioned above [Brusov, and others, 1996, Tiumentsev Yu. V., 2002, 2004a, 2004b] (Figure 4.5):

- Soft computing (SC) methods and tools: artificial neural networks (ANN), fuzzy logic (FL) systems, evolutionary techniques (genetic algorithms (GA), genetic programming etc.), uncertainty management techniques;
- Extended soft computing (ESC) methods and tools: soft computing methods and tools together with knowledge-based systems (KBS) and multiple-agent technologies; and
- Semi-soft computing methods and tools: extended soft computing methods and tools together with mathematical modeling (MM) techniques.

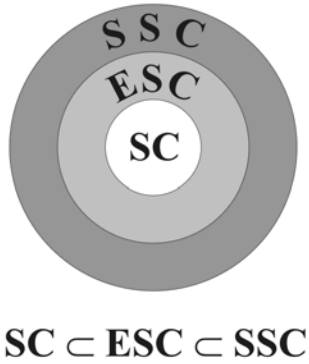


**Figure 4.5: Relationships between Soft Computing (SC), Extended Soft Computing (ESC) and Semi-Soft Computing (SSC).**

It should be emphasized that only presence of system elements based on soft and semi-soft computing techniques **do not cause** this system to be intelligent. After all, these elements can merely "replace" some other elements based on conventional techniques. For example, we can take a controller (a control channel in particular) and approximate its functional dependencies for gains in control law by means of some artificial neural network. After that the ANN-based representation can be used to compute gains related to some particular situation. A representation form is changed here but the essence of control law remains permanent. The controller becomes "ANN-based" but it still does not become "intelligent". Therefore "neural control", "fuzzy logic control", "neuro-fuzzy logic control" and other similar terms are not synonyms at all for the "intelligent control" term. Some system will be "intelligent" only if it is capable to adapt and to learn (Figure 4.6).

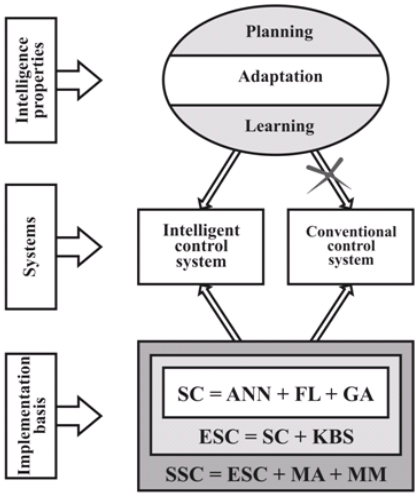


**Figure 4.6: Intelligent and Conventional Control Systems versus Soft and Semi-Soft Computing.**

It means intelligent systems can reconstruct effectively their behavior depending on current situation as well as accumulate solution experience for various tasks and use this experience to solve earlier unknown tasks. It's quite another matter that a system named as "intelligent" can be hardly implemented without soft and semi-soft computing tools. So then realization of a control system basing on semi-soft information technology is most likely a necessary condition and is not a sufficient one.

Based on soft and semi-soft computing, intelligent flight control are directed to reply the demands solving such main problems as: enhance the mission capability of aircraft; improve aircraft performance by learning from experience; make aircraft less dependent on proper human actions for mission completion; increase flight reliability and safety.

#### 4.3.2.4 Adaptive and Intelligent Systems as a Basis for System-Level Integration of Control

##### *4.3.2.4.1 Adaptation in Controlled Systems*

A need for adaptation and adaptive systems arises as a rule in case of multiple interconnected task solved under uncertainty conditions and severe resource constraints.

Adaptive systems play very important role as a part of all kinds of complex systems especially air vehicles both piloted and unmanned as well as vehicles for space, land, and sea domains. Also intelligent systems demonstrate growing significance in recent years.

Adaptive systems are thought in some broad sense namely as systems which are capable to modify their behavior according to varying conditions of their existence (environment, goals etc.). In other words adaptive system is a system, which has some mechanisms providing a possibility to live and to act in conditions of various and numerous uncertainties. One of most important mechanisms of this kind is intelligence, although it is not a unique one.

Four kinds (hierarchical levels) of adaptation can be distinguished:

- Parametrical adaptation (adjustment, self-adjustment);
- Structural adaptation (reconfiguration and/or restructuring);
- Object adaptation (correction of system composition); and
- Goal adaptation (adjustment of demands).

*4.3.2.4.2 Parametrical Aadaptation*

Is supported with variation of control system adjustment parameters, for example, controller gains.

*4.3.2.4.3 Structural Adaptation*

A required versatility for a controlled system not always can be reached only with varying values of control system parameters. Next hierarchical level for adaptive systems are systems, which are capable for structural adaptation, i.e. for a modification of the system structure (it includes a set of control system components as well as links between these components) in regard to changing situation and goal. Simplest case is a control system equipped with a set of alternative versions of control laws. Only one of the versions operates at some particular instant.

*4.3.2.4.4 Object Adaptation*

A case is quite possible when no variation of control system structure and parameters exists to satisfy goals of the controlled system. It is quite natural because of potentialities are restricted for any system; the potentialities are constrained with the system "structure". If such a case arises then we can involve next adaptation level, i.e. object adaptation.

The principal idea of this adaptation level consists in solving of a required task by means of a set (group, constellation, swarm) of interacting systems instead of some separate controlled system as it was for two previous cases.

The essence of this approach can be illustrated for a task of intercept of aerial targets, including multiple ones. If airspace area and number of targets are relatively small then the task can be solved often with a single interceptor equipped with missile weapon and multi-channel system for finding and tracking of targets. If these conditions are not satisfied then capability of sole aircraft is not adequate to the task. In that case we can form some heterogeneous group of systems intended to solve cooperatively the mutual task. Such group can include vehicles of several types, ground-based equipment and so on.

*4.3.2.4.5 Goal Adaptation*

If object adaptation level is not adequate to the task similarly two previous levels, i.e. the level does not provide achievement of the goal, it is quite possible the stated goal is not achievable at all. We can change control goal to make it achievable.

The essence of this adaptation level can be illustrated in the following way. Let a task be surveying of particular object for some self-propelled vehicle delivered onto a celestial body. It can be revealed that solving of the task demands too many resources. This circumstance impends to defeat the plans of the expedition. If that's the case, the system can replace one goal with other following some general purposes (for example providing highest possible extraction of knowledge about the celestial body). The system can search for that an object "similar" to excluded one or to refuse this element of the exploration plan at all and to switch over to other plan elements.

*4.3.2.4.6 Implementation Basis for Adaptive and Intelligent Systems*

There are some advanced research and engineering areas associated with various sorts of adaptive and intelligent systems. As it was mentioned above list of these areas includes first of all soft computing technologies which combine into one such approaches as artificial neural networks, fuzzy systems and evolutionary techniques (genetic algorithms, genetic programming etc.). If we add knowledge-based systems to the list then we get extended soft computing. The addition of numerical simulation techniques to the extended soft computing leads to semi-soft computing.

During past 12–15 years methods and tools involved in soft computing, extended soft computing and semi-soft computing have been developed very intensively. Many interesting results have been obtained and accepted [Samarin, A.I., 2005], [Brusov, V.S., and others, 2004], [Stengel, R.F., 1993], [Suykens, J.A.K., and others, 1996], [Sontag, E., 1993], [RayChaudhuri, T., and others, 1995], [Ronco, E., and others, 1997], [Haykin, S., 1994], [Berthold, M., and others, 2003], [Michalewicz, Z., 2001], [Domany, J.L., and others, 1992], [Piegat, A., 2001], [Pal, S.K., and others, 2003], Polkowsky, L., and others, 2000]. Among these results we have those related to all aspects of management and control problems for complex systems: observation, sensor fusion, estimation, control, guidance, navigation, mission planning, situation assessment, decision making and so on.

The results obtained allow to assert that extended soft computing and semi-soft computing techniques provide possibilities to solve management and control problems with conditions unacceptable for traditional control theory methods, for example, for a case with large abrupt change of controlled system configuration.

These possibilities seem to be very important for piloted vehicles as well as for unmanned vehicles, especially to solve problem of system-level integration of control. Such possibilities could be helpful for many subjects and efforts including integration of control for both stability and maneuvering, integrated flight and fire control, integrated flight and propulsion control, automatic ground collision avoidance, swarms of unmanned vehicles, cooperative actions of dissimilar vehicles, safe mixing of manned and unmanned vehicles, automated mission performance of unmanned vehicles, automatic air collision avoidance and many others.

As it was mentioned already it hardly makes sense to search some “miracle cure”, which is some kind of information technology (IT) alternative with respect to the traditional imperative-kind IT, to overcome revealed obstacles. Each IT picked up from the long list of modern traditional and advanced information technologies has both virtues and shortcomings. However as experience demonstrates capability of any single IT is not sufficient to solve all the problems arising during development and maintenance of complex systems especially adaptive and intelligent ones. It could be more productive to combine various information technologies for the purpose of using their strong features and to compensate mutually their inherent shortcomings. Based on these reasons it is not rational also to oppose traditional information technologies and advanced ones as well as some advanced technology to each other.

We have to work for *synthesis* of concepts, techniques and tools consisted in these information technologies. It is the foundation that allows to solve the problems mentioned in previous sections. The most important goal of such synthesis is to define some kind of sufficiently general scheme that will be called as semi-soft computation (SSC) model. This model must allow to obtain partial (special-kind) models for various branches of the SSC namely for artificial neural networks (ANN), fuzzy logic (FL), genetic algorithm (GA), knowledge-based system (KBS), multiagent system (MA) as well as conventional mathematical modeling (MM) by putting into consideration appropriate requirements and conditions.

Let us outline an approach to realization of the semi-soft computation model.

One of the key elements of the SSC approach is an idea of neural network as a special kind composition of mappings and, on the other hand, as a dynamical system what is especially important for feedback neural networks. Such concept makes possible to reveal deep relationships between artificial neural nets and other fields of the SSC as well as their interrelations with conventional mathematical modelling. Similar approach is proved to be effective in respect of other SSC parts namely FL, GA, MA and KBS. Most generally the semi-soft computation model can be interpret as a composition of some special-kind mappings having either static (for instance perceptron-like feedforward networks) or dynamic (for example recurrent neural networks) nature. This model can be reduced to certain “pure” model based solely on the FL, GA, KBS, MA, MM concepts or to some “truncated” hybrid model (e.g. ANN+FL,

GA+ANN, ANN+KBS, GA+MA etc.) applying corresponding conditions. Such kind of representation for the SSC model opens up possibilities to rigorous mathematical analysis of the model and its various specialized versions as well as to define appropriate theoretical foundations for the SSC.

One exists however another aspect of the discussed problem. A computing experiment is needed to evaluate efficiency and capability for approaches and techniques offered. Therefore it is quite important to represent the SSC model not only keeping its mathematical analysis in mind but also taking into account peculiarities of its computing implementation. It seems very promising to accomplish this purpose by using an extended semiotic model which is the second key element of the SSC approach. This model is based on a generalization of the semiotic model concept suggested by Dmitry Pospelov in [Pospelov, D.A., 1986]. The generalization is accomplished as a result of "composition-style" semiotic model reformulation corresponding to the SSC approach.

A value of suggested theoretical (formal) models will be not so much without appropriate procedures which allow to reassert various kinds of application problems in terms of the SSC. In this connection the third key element of the SSC approach has to be a development of relevant tools ensuring the reassertion.

The fourth key element of the SSC approach consists in development of suitable problem solving techniques. These techniques are based on concepts and tools of the SSC branches (ANN, FL, GA, KBS, MA and MM) transformed to keep the SSC-style of model representation, i.e. using special-kind composition of mappings plus extended semiotic model.

Available results demonstrate very deep interconnections between such semi-soft computing areas as ANN and MM, ANN and FL, ANN and MA, FL and KBS, FL and GA, ANN and GA, KBS and conventional imperative technologies etc. Therefore one might conclude that it is very important to reveal, study and develop such kind of relationships because this approach promises to integrate really achievements from various SSC fields into comprehensive whole system. Besides a possibility emerges to formulate corresponding methodological principles and to derive suitable theoretical foundations for the SSC. These results are quite needed to build an information technology based on the SSC concepts and models. This technology, in turn, makes possible to increase considerably a level of complexity for application problems, which are available to solve them effectively.

#### *4.3.2.4.7 Multiagent Architecture for Adaptive and Intelligent Systems*

Adaptive and intelligent systems are very complicated ones. Complexity of these systems prevents their effective implementation for useful applications by means of conventional software architectures based on pure imperative information technologies. There is an alternative way to implement adaptive and intelligent systems involving *multiagent models* and corresponding *multiagent systems*.

The multiagent system is a system consisting of numerous autonomous agent modules interacting with an environment (Figure 4.7) and having such properties as:

- Each agent possesses an autonomy;
- There is no centralized control for agents;
- Data sources and data access are decentralized; and
- Agents operate in asynchronous mode.

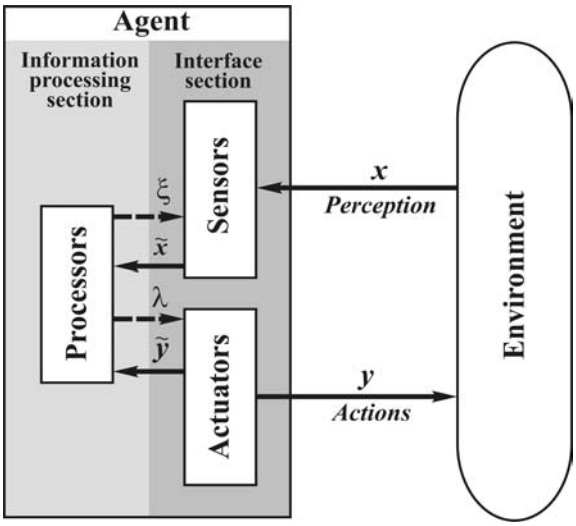


**Figure 4.7: Generalized Structure of Autonomous Agent and its Interaction with Environment.**

The multiagent model is spreading now more and more widely including complex system development processes. Conventional approach to solve the complex system development problem is based on representation of the system as some subsystem hierarchy with subsystems rigorously subordinate to each other as well as with explicit structurization of all interconnections and interactions between subsystems.

The new approach introduces a concept of autonomous system called agent instead of the subsystem concept. The agent has a high autonomy degree and it is independent of other agents. The rigorous set of relations for conventional-style model is replaced with a set of rules (protocols) defining interactions between agents. Besides some set of procedures is usually introduced to allow agent interactions if we are needed cooperative behavior for the agents solving certain common problem. This approach ensures to reduce considerably complexity and expenditures for the systems developed including adaptive and intelligent ones. Complexity of interactions between parts of the systems can be also reduced significantly. Therefore a true possibility appears to create really useful application systems.

Two examples of multiagent architectures are shown in Figure 4.8: multiagent model based on direct interaction with the environment and without any coordination between agents (Figure 4.8a); and multiagent model involving agent cooperation via a blackboard (Figure 4.8b).

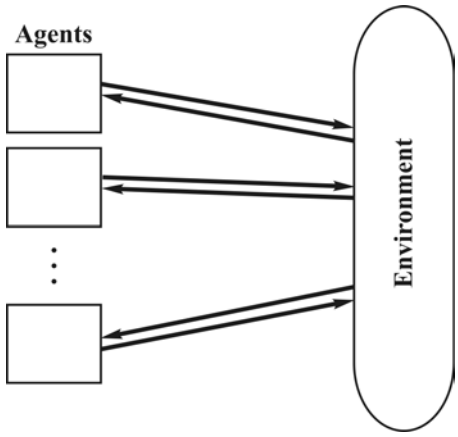


**Figure 4.8a: Multiagent Architectures Based on Direct Interaction With the Environment and Without any Coordination Between Agents.**

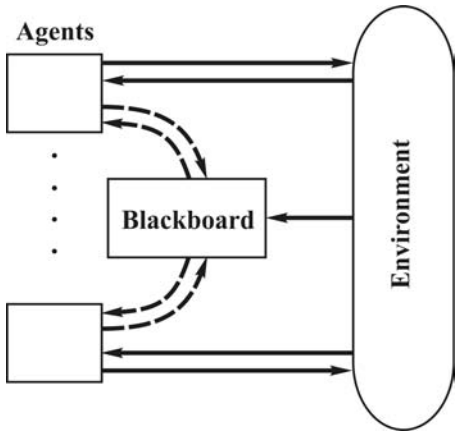


**Figure 4.8b: Multiagent Architecture Involving Agent Cooperation via Blackboard.**

*4.3.2.4.8 Intelligent Autonomous Vehicle as a Typical Semi-Soft Computing Application Example*

There are developed and widely used manual control and automatic control techniques as well as automated control techniques combining manual and automatic ones. Development of advanced piloted vehicles as well as highly autonomous unmanned aerial vehicles demands some new ways to solve automatic and automated control problems with the regard conditions listed above.

Grounding on soft and semi-soft computing techniques we can solve problems associated with highly autonomous vehicles including a case of intelligent autonomous vehicles (IAV). The IAV are systems which are capable:

- To reach specified goals within highly dynamical environment taking into account various and numerous uncertainties;
- To modify specified goals and to generate new goals and goal sets basing on some motivations;
- To extract (to mine) a new knowledge, to accumulate and generalize experience in solving of diverse problems, to learn basing on the experience, to modify system behavior basing on the new knowledge and experience;
- To adapt to problems which are needed to solve including some problems not presented at the original system design; and
- To form "collectives" ("communities") made up from IAVs directed to cooperative solving of some common complex problem.

An activity of IAV within some environment can be divided into three main parts:

- Perception of a current situation (situation = external-situation + internal-situation), i.e. revelation of a challenge – sensor functions;
- Producing of a system reaction ("system's reply") for the current or predicted situation (kinds of possible reactions are system state evolution, reconfiguration, restructuring, adaptation of goals, unsupervised learning, self-organization etc.) – decision making functions; and
- Implementation of reaction produced for the current or predicted situation – effector functions.

A capability to learn and to accumulate (and to generalize) experience is provided for a controlled system a very high adaptability level in regard to variations in activity conditions.

In particular, a possibility arises to solve such problems as:

1) System-level integration of control (one-vehicle-level) for piloted and unmanned air vehicles:
   - Integration of control for both stability and maneuverability;
   - Integrated flight and propulsion control;
   - Integrated flight and fire control;
   - Automatic ground collision avoidance;
   - Reconfiguration and restructuring of control as a reaction on large abrupt changes of controlled system configuration;
   - Structure adaptive and intelligent control (smart materials and structures, adaptive structures);
   - High-autonomous intelligent vehicles; and
   - … many others.

2) System-level integration of control (multiple-vehicles-level) for piloted and unmanned air vehicles:
   - Automatic air collision avoidance;
   - Cooperative actions of dissimilar vehicles;
   - Safe mixing of manned and unmanned vehicles;
   - Tasks for communities of high-autonomous intelligent vehicles;
   - Swarms of unmanned vehicles; and
   - … many others.

#### 4.3.2.5 Intelligent Control Techniques Based on Human Operator Experience

Let us consider in more details the second element of the IAV activity structure because of key role of the decision-making (control) functions.

There are many ways to solve the problem. One of these ways is based on using of rich control experience accumulated in piloted aviation.

There are three ways to generate intelligent control laws needed to solve tasks indicated above:

- "Mimic" approach based on using of some neural or fuzzy-neural network or some ensemble of networks to imitate control actions produced by human pilot (operator); this approach essentially uses experience accumulated by pilots to generate control laws implemented with automatic systems; the pilot experience needed for "mimic" approach realization can be revealed by using of flight simulators;
- "Formal" approach based on learning of some neural or fuzzy-neural network according to certain set of indices describing required behavior of controlled system; this approach do not use any pilot's experience at all; and
- "Combined" approach which merges "mimic" and "formal" approaches; in accordance with this approach some controller is constructed and learned at the beginning by means of "formal" approach tools, then such controller is refined using "mimic" approach tools using flight simulator and knowledge base containing pilot experience.

A human operator experience and skills can be used in the context of mimic and combined approaches by means one of such two ways as:

- Imitation of operator's behavior for specified conditions in solving some control problems; and
- Synthesis of a control law implementing human operator's experience and skills by means of approximation of dynamical model, which describes operator's activity in solving some control problems.

Implementation of mimic or combined approach causes some problems. The problems arise as a result of necessity to generate an approximate representation of human operator as a dynamical system.

The human operator as a modeled object is characterized with such peculiarities as:

1) Operator's model is generally very complicated, nonsteady and nonlinear especially for cases of multi-channel control tasks, influence of complex external impacts and disturbances, control tasks in the event that dynamic of controlled object changes in sharp and unpredictable manner (for example because of structural damages and equipment failures of controlled object).
2) Operator's model depends essentially on a task which needed to be solved. There are many of such tasks in the case of complex multiple-mode controlled object. Accordingly, a lot of different models are needed to describe appropriately human operator activity as a whole.

Thus it is necessary to solve approximation problem for dynamical systems realizing complex and multiple-variant behavior.

Suppose a considered dynamical system realizes a transformation (mapping) $F$ of input signals $x$ into output signals $y$. It is necessary to find an approximate representation for $F$ such that a behavior of the dynamical system with mapping $F$ would be "similar" (in some predefined sense) to a behavior of a source system, i.e. some system with a human operator as a controller.

Mathematically and computationally this problem is rather difficult.

Let us assume for the sake of definiteness that inputs $x$ and outputs $y$ are vectors with elements, which belong to one of two spaces, $R$ and $C[a,b]$: $R$ is the space of real numbers and $C[a,b]$ is a space of real-valued continuous functions on the $[a,b]$ interval, where $a, b \in R$.

Problems of approximate representation for mathematical objects like $F$ can be divided into four types depending on a kind of inputs $x$ and outputs $y$:

1) Problems with $x \in R$ and $y \in R$; they are traditional approximation problems for some function F.
2) Problems with $x \in C[a,b]$ and $y \in R$ ; they are approximation problems for some functional F defined on functions $x \in C[a,b]$ and possessing real number values $y \in R$.
3) Problems with $x \in R$ and $y \in C[a,b]$ ; they are approximation problems for some differential operator F, which depends on real-valued parameters $x \in R$ and possessing function values $y \in C[a,b]$.
4) Problems with $x \in C[a,b]$ and $y \in C[a,b]$; they are approximation problems for some integral operator $F$ defined on functions $x \in C[a,b]$ and possessing its function values $y \in C[a,b]$.

First type of problems is traditional approximation problems for functions. A vast majority of works associated with approximation problems for mathematical objects including neural network based investigations is given up to such kind of problems.

Second and third types of problems are some problems related to approximation for systems of differential equations. There are traditional ways to solve these problems (difference schemes, functional expansions etc.) as well as non-traditional ways based on using of artificial neural networks, for example. However quantity of papers realizing non-traditional approaches to the problem is relatively small now.

Fourth type of problems is certain problems associated with approximation of integral equations. The state of things in this area is about the same as for the second and third type problems.

Approximation problems related to dynamical systems like human operator concern to the second and third type problems mentioned above. Conventional (traditional) techniques to solve these problems are hardly applicable because of their inconvenience, inflexibility as well as large requirements in computational resources. Techniques based on artificial neural networks are more preferable here. They allow handling considered problems.

Neural network based implementation of intelligent control laws by means of approximation of appropriate dynamical models causes some problems:

1) There are many control tasks realized by human operator for sufficiently complex controlled object. It would be ineffective to generate a separate neural network for each individual task, especially for a case of expandable list of tasks. It will be better to use a single network or a system of interconnected and interacted neural networks instead of a set of individual networks.
2) An additional on-line learning can be required in many cases, for example if dynamic of controlled object changes in some sharp and unpredictable manner. Under the circumstances arise very hard limitations on operating speed of adaptation mechanisms.

There are two ways leading to solution of these two problems:

- Neural networks with a dynamical preliminary adjustment; and
- Ensembles of neural networks.

Neural networks with a dynamical preliminary adjustment are directed to realization of complex and very complex implicit functional dependencies. These networks include dynamically adjustable work elements (neurons) as well as so-called interneurons, which influence on tune parameters of the work neurons depending on values of input network signals and it possibly subject also to values of some additional parameters. The network is capable to absorb multiple models simultaneously by means of combinatorial interneuron inhibition process. The interneurons may receive their input signals from some other network solving a classification problem. A solution of this problem represents a name (or a number) of the model, which is adequate to current work conditions. These networks can be learned additionally to react on an expansion of the associated task list. Previous experience of the network is preserved and enlarged, i.e. additional training do not destruct its preceding capabilities.

Some different approach is ensembles of neural networks. An ensemble of neural network is used for implementation of required task set according to this approach. Each of these networks associates with some tasks from the task set. Two architecture version of the ensemble is possible depending on type of interaction between networks involved:

- Architecture version based on a model “master–slave” (it can be named more exactly as “conductor–performer” model); and
- Architecture version based on multiple-agent model.

For the “conductor–performer” model there are $N$ neural networks execute required tasks and some $(N+1)th$ network, which “conducts” the other nets: “conductor” orders and “performers” obey.

In case of the multiple-agent version of the network ensemble there are *N* relatively autonomous and relatively independent neural networks associated with the task set. These networks interact with each other according to some rules. Such interaction is organized to solve all tasks entered in the task list.

It may be stated concluding the discussion that human operator experience and skills together with techniques of semi-soft computing allow to solve various important tasks associated with the problem of system-level integration of control.

## 4.4 AUTONOMOUS VEHICLES

### 4.4.1 Automation and Autonomy

The Free Dictionary (www.tfd.com) defines automation as:

- The automatic operation or control of equipment, a process, or a system.
- The techniques and equipment used to achieve automatic operation or control.

Humans have been developing automated systems for centuries. With the advent of modern computers, the complexity of automated systems has risen exponentially and has resulted in systems that exhibit autonomous capabilities.

Autonomous is defined as:

- Not controlled by others or by outside forces; independent, and
- Independent in mind or judgment; self-directed.

Due to the generic nature of autonomy, individual disciplines have defined their own unique meaning to autonomy, with is specific for the context of their research. The following sections reflect the diverse meanings that researchers have applied to the words “Automation” and “Autonomy”.

### 4.4.2 Autonomy for Robotics Systems

Under the Wikipedia definition, autonomous robots “are robots which can perform desired tasks in unstructured environments without continuous human guidance.” Given this definitions, one route to achieving autonomous operation is the SMPA paradigm [Brooks], where robots sense, model, plan and act. Thus, the components of autonomy may be itemized as follows:

1) **Sensing:** Via proprioception, the robot senses its own internal status, while exteroceptive sensors allow the robot to sense the external environment.
2) **Modelling:** The robot constructs a world representation using sensor data. The representation’s fidelity must allow obstacles and hazards to be identified. Commonly used world representations include: occupancy grids [Moravec], 2½ D representations [Broten 2007a] and, potentially, 3 D representations. Into this representation, the robot may insert other types of information such as known landmarks and features.
3) **Planning:** In order to accomplish a task, the robot must plan a course of action. This plan must account for the robots internal status, the current environment, and the task or mission goal.
4) **Acting:** With a plan in hand, commands must be sent to the actuators and motors that physically propel the robot. Historically, a monolithic SMPA implementation resulted in poor performance as modelling and planning caused the system to be slow and unresponsive. In order to make the act stage responsive, a hybrid architecture, that is composed of both reactive and deliberative control strategies, is commonly implemented [Rosenblatt, Simmons].

In addition to the traditional SMPA paradigm, some leading edge robotics systems have adding learning to enhance autonomous capabilities. Learning can be invaluable as it allows a robot to adapt to environment changes such as variations in lighting conditions and seasonal environmental changes [Thrun].

### 4.4.3 Complexity for Autonomous Systems

For autonomous systems, complexity manifests itself in different ways:

1) Complexity of the operational environment.
2) Complexity of the task or mission to be performed.
3) Co-operation between multiple autonomous systems also leads to complexity.

The following sections provide more details with respect to the complexity encountered by each of the above items.

#### 4.4.3.1 Environmental Complexity

The environment encountered by an autonomous system varies greatly. Both UAVs and UUVs operate in relatively simple and forgiving environments. Once above the ground plane, a UAV's environment is almost completely obstacle free. While operating in this benign environment, a UAV has only minimal requirements to sense its world, and no need to create a world representation. Although a world representation is not required, there are environmental conditions that add other forms of complexity. This complexity can be lumped into two large groups:

1) The atmosphere directly affects the vehicle's motion. These effects include turbulence, shear, and vortices. These atmospheric effects may have a considerable influence on the vehicle's linear and angular motion, and are potentially catastrophic in terms of accidents (recall wake effects caused by large vehicles on smaller ones, for instance).
2) Other contributors to complexity include: boom motion during aerial refuelling, deck and optical system motions during carrier landing, target maneuvering and landing on extremely short runways.

UUVs operate in environments that are similar to those encountered by UAVs. A UUV, operating in deep water, can also assume a benign and almost obstacle free environment. As UAVs and UUVs approach ground terrain the environmental complexity increases significantly and eventually approach the complex, unstructured environments in which UGVs must operate.

UGVs must operate over a wide range of environments, from those with low complexity, to those with high complexity. The following items are example UGV environments.

1) A low complexity environment as shown in Figure 4.9 (Parking lot or highway, open fields with no or very little vegetation).

**Figure 4.9: A Low Complexity Environment.**

2) A medium complexity environment, which is shown in Figure 4.10 (Open fields with vegetation, gravel or dirty roads, urban streets).

**Figure 4.10: A Medium Complexity Environment.**

3) A high complexity environment is shown is Figure 4.11 (Heavily vegetated terrain such as forests or jungles, terrain featuring rubble whether it be in a natural or urban setting, building interiors, actively hostile areas).

**Figure 4.11: A High Complexity Environment.**

Currently, autonomous systems usually operate in low complexity environments. Research robots have a limited ability to operate medium complexity environments. Operations in high complexity environments are not possible given the current state of autonomous systems development.

### 4.4.4 World Representation and Understanding

Due to the complexity associated with ground environments, autonomous ground vehicles must sense and, at some level, understand their environment. The process of representing and understanding the world may be considered from many points of view. A stationary sensor can create a world representation, but its inability to move means this representation has limited internal uses. A sensor mounted on a human piloted vehicle can create a world representation, which humans may use to enhance situational awareness, but it is the human who does the interpreting and understanding.

Tele-operated vehicles require a human in-the-loop, where there is a heavy reliance upon the human for input and guidance. Thus, the tele-operated vehicle has limited requirements for world representations.

For a system to exhibit autonomy, it must make arrive at it own conclusions and internally direct it actions. A fundamental prerequisite to planning and decision making is representing and understanding one's environment.

#### 4.4.4.1 A Ground Vehicle Perspective

Robotics researchers, especially those focused on ground vehicles, have expended significant energies in developing applicable world representations. Due to the inherent complexity of a UGV's environment, researchers in this field were forced to confront the world representation issue at an early stage in their research. Over the years, this research has yielded solutions that allow UGVs to successfully navigate unstructured terrain, but only terrain of a low to medium level of complexity [Broten 2007b, Herbert, Kweon, Bellutta, Goldberg, Lacroix, Kelly]. These solutions "represent the world", but they make little effort to "understand" it. In a representation such a 2 ½ D terrain map, all obstacles are equal. The classical terrain map implementation does not differentiate between true physical obstacles such as stones and concrete, and soft obstacles such as grass or shrubs.

Achieving "understanding" via learning is a new and emerging field of research. Understanding via learning allows a robot to learn from experience, thus it adapts to changes. Specific research has focused on adapting to environmental changes for unmanned ground vehicles. The Stanley robot, from Stanford University, uses probabilistic learning to enable high-speed operation in unstructured terrain [Thrun]. Other researchers have investigated how to differentiate between vegetation and true obstacles [Macedo]. Finally, NASA researchers have developed algorithms that predict wheel slippage from visual information [Angelova].

#### 4.4.4.2 Understanding from the Machine or Computer Vision Perspective

Scene understanding remains one of the most difficult problems for machines to overcome. An example would be automatic target recognition (ATR) where machines are good at looking for a precisely defined object, but can not recognize a general class of objects (i.e. they can not pick out the cup on the table unless the cup is precisely defined).

### 4.4.5 Single and Multiple Platform Systems

Complexity, automation, and autonomy appear within single as well as multiple platforms. In this respect, either system may be preferable depending on the mission requirements. A problem that would be well served by a single-asset solution could be identified for instance by the following characteristics:

1) Hard to separate into pieces:
   - Highly interdependent system dynamics.
2) Physical dispersion adds little benefit:
   - Simultaneous actions add little; and
   - Sequential tasking is adequate/optimal.
3) Information transfer is costly/inadequate:
   - Threats make communication undesirable;
   - Geographic separation makes communication difficult;
   - Terrain/environment make communication difficult; and
   - Time lags and latent data compromise stability or optimality.

A problem that would be well served by a multi-asset solution, on the other hand could be characterized by:

1) Easy to separate into pieces:
   - Dynamics are loosely coupled; and
   - Time-scale separation is apparent.

2) Physical dispersion can be used to great effect:
   - Simultaneous tasking has great utility; and
   - Sequential tasking is inadequate.
3) Information transfer is not costly:
   - A global information state can be maintained;
   - Local information is adequate; and
   - Lags and latency are acceptable.

All this with the caveat that the complexity of some problems is so overwhelming that separation is the only realistic option available. The benefits of having multiple assets add degrees of freedom to the solution of a problem, e.g. allowing a choice of vehicles to service a target. However, the flexibility comes with additional complexity that is imposed in the form of constraints, e.g. a target must be confirmed before attack, and attacked before battle damage is assessed. So, the meaning of "complexity and automation" for multiplatform systems perhaps implies different concepts from those associated with single platform systems.

Other key factors that make a multi-asset solution different from a single-asset solution are:

1) Problem division; and
2) Information availability.

The former includes actions/items such as Order of precedence (Kill chain), Coupling of tasks, Performance, Computations. The latter deals primarily with Communication, Centralization of processing, Correlation of targets, and Moving targets.

### 4.4.6 Co-operative Control Operations

In this section the challenges of cooperative control operations are outlined. To fix ideas, the discussion is focused on a simplified surveillance scenario. A team consisting of a flight of Unmanned Air Vehicles (UAVs) is tasked with searching for and recognizing multiple targets in a battle space with many false targets. Allowing for an unstructured environment in cooperative control operations is essential. The presence of false targets/clutter forces one to consider a probabilistic setting. While the dynamics of unmanned air vehicles and, in particular small air vehicles are important, considerations relevant to resource allocation and optimization, the prevailing information pattern and unstructured environment/ uncertainty dominate the analysis. The potential benefits of cooperative team actions, and, in general, the benefits of cooperation among distributed controlled objects are addressed.

#### 4.4.6.1 A Taxonomy of Teams

A team is here defined as a loose collection of spatially distributed controlled objects, a.k.a., Unmanned Air Vehicles (UAVs), that have some objectives in common [Ho et al., 1972; Marschak, 1972]. Air vehicles may be too restrictive a term – generically, a team consists of members, or agents, and the team can (and generally will) include humans as operators, task performers (think of target recognition), and/or supervisors. The presence of a common objective forges a team and induces cooperative behavior. If the air vehicles are working together to achieve a common objective, then they are considered cooperative. Different degrees are possible: coordinated; cooperative; and collaborative. At the same time, additional individual objectives of the team members can encourage team members to opt for a weak degree of non-cooperative, competitive, or adversarial action.

When team decision and control problems are discussed, it is important to address the unstructured environment/uncertainty, the organizational structure, the information pattern, and task coupling. Individual operational scenarios can be dominated by one of the above, but will contain elements of all of them. The interaction of these different facets of cooperative control of a team cannot be ignored.

#### 4.4.6.2 Team Coordination

This is the strongest degree of cohesive team action. Consider a set of UAVs that have been designated to be part of the team, and they all share a single team objective and thus strive to optimize a single payoff function. The team could have more than one payoff function that it wishes to optimize, which would then entail multi-objective optimization [Luce, 1989]. Oftentimes, the different payoff functions can be assigned weights and rigorously combined into a single objective function. There is no conflict of interest among the team members, otherwise an incentive scheme [Groves, 1973] would need to be devised. The important distinction here is that particular team members do not have individual objective functions: a team member is a resource that can be over-utilized or under-utilized, if that will best achieve the team objective. The team members are obligated to participate and any assignments, tasking, or agreements are binding; they cannot opt out. At the same time, the forming of coalitions is not possible. The team may be geographically distributed, but it operates as a single unit.

#### 4.4.6.3 Team Cooperation

Each of the team members has a private objective function which he strives to optimize, in addition to the team objective function. The private and team objective functions are weighted such that $0 \leq w \leq 1$. A weight of $w = 1$ on the private objective function means the member acts in its own self interest, in which case there is no team action. A range of $0 \leq w \leq 1$ on the private objective functions corresponds to an increasing level of cooperation of the team members, to where $w = 0$ entails strict coordination. There is a possibility for conflict of interest, but, due to the structure of the objective functions used, it is not generally dominant. In most cases, local objectives such as fuel conservation and survivability are not in conflict with the team objectives and can be jointly achieved. Thus, the multi-criteria optimization aspect of the problem is not dominant and a weighted sum of the objective functions yields a conventional optimization. If they are in conflict, the team objective takes precedence according to the weight used.

#### 4.4.6.4 Team Collaboration

This is a looser form of cooperation. In some cases this can be the result of the task assignment or resource allocation method used [Guo, 2001; Bertsekas, 1992, Bertsekas, 1991; Bertsekas, 1993; Bertsekas, 1993 b]. At the global (team) level, the focus is on task completion, a.k.a., feasibility. Each team member tries to maximize his local objective function consistent with team task completion while avoiding tasking conflicts. This requires that a protocol be designed for negotiation to arbitrate conflicts [Olfati-Saber, 2003; Lamport, 1998]; this connects with the currently developed theory of communication networks. Ideally, those tasks that are best for each member to perform according to their private objective function are tasks that need to be done for the team objective function. In addition, there is the additional implicit constraint that the selected tasks are not redundant. Here there are possibilities of significant conflicts of interest: if the team members have a set of equally valuable (to them) tasks, then likely the conflicts can be resolved (mutually agreeably), and collaboration can be as efficient as coordination. Obviously, the more tightly coupled the various team tasks are, the more difficult it is to achieve a collaborative solution. Strong coupling will occur if a homogeneous team of multi-role UAVs is employed, or if the battle-space is small. A coordinated or cooperative operations approach will be needed. Also, negotiation is not compulsory; the members are not forced to participate. If a solution for a particular team member cannot be found, then this team member can opt out and join an alternative team that has a better overall match of member objective with team objective.

#### 4.4.6.5 Goal Seeking Team Action

This is a further abstraction of a team decision and control problem. Here there are no a priori designated team members. The team is formed from a set of available resources that are loosely networked. Each UAV can simultaneously be a member of several teams. Once a team's objective is achieved, the team will dissolve. The goal in general is abstract and has to be converted to a sequence of intermediate objectives/milestones and the objectives in turn have to be converted into a set or sequence of tasks that are assigned to, and coordinated between, the team members. There might be an intrinsic conflict of interest between the goals of the teams that the UAV can simultaneously be a member of. The teams are therefore self organizing [Jadbabaie, 2003]. This results in a high level of autonomy, where the vehicles are independent agents that however work together in an ad hoc fashion, as needed, to achieve an overarching goal. Each vehicle also strives to optimize its utility function, which may be handled through coordination, as previously mentioned.

#### 4.4.6.6 Non-Cooperative Behavior

To this point we have been describing the different modes of how UAVs interact with other UAVs in a team. In a team, by construction, the objectives are basically compatible.

We now address the control of UAVs that are operating in teams that are competing and, possibly, adversarial. If there are two teams, this is the domain of a significant part of game theory research. This includes strictly competitive zero sum games and also non zero sum games, e.g., bimatrix games [Vorobev, 1977; Luce, 1989]. This is the field of much military research as in the Blue team against the Red team war game. However, the field is much richer than this, because in reality there can also be, e.g., a White team and a Green team. What's more, membership in a team can change fluidly, and the teams can form coalitions which can also dissolve. This is a rich area in which complementary and conflicting interests, collusion, hidden objectives, signalling, diversion, gambits, propaganda, and disinformation play a significant role.

#### 4.4.6.7 Conflict of Interest

Conflict of interest is brought about when the UAVs have different objective functions which, in addition, are structured such that joint action which simultaneously optimizes the different objective functions is not possible or feasible. Thus, consider the performance functionals $J1(u, v)$ and $J2(u, v)$ for UAV 1 and UAV 2, respectively; the decision variables of UAV 1 and UAV 2 are $u$ and $v$, respectively. The payoff for UAV 1 is a function of the actions (or decisions) $u$ taken by UAV 1, as well as the actions (or decisions) $v$ taken by UAV 2. Each affects the value of the other's objective function. If the objective functions were not coupled, that is, $J1(u)$ and $J2(v)$, then obviously there would is no conflict of interest. Alternatively, if $J1(u, v) = J2(u, v)$, there is no conflict of interest either.

If this is not the case and an ahead of time communicated plan (an agreement) is not in place, then this definitely is a non-cooperative scenario. The question is, what strategy does UAV 1 use to determine the "best" action $u$ to minimize his cost $J1$? Similarly, UAV 2 is faced with the problem of minimizing his cost $J2$. The actions $u*, v*$ are "best" if $J1(u*, v*) \geq J1(u*, v)v$ and $J2(u*, v*) \geq J2(u, v*)u$. This means that if UAV 2 deviates from $v*$, then UAV 1 will do better than $J1(u*, v*)$ and if UAV 1 deviates from $u*$, then UAV 2 will do better than $J2(u*, v*)$. Thus, $J1(u*, v*)$ and $J2(u*, v*)$ constitute guarantees for the respective players, no matter what the other player/vehicle does. This is a Nash (non-cooperative) equilibrium point [Vorobev, 1977; Luce, 1989].

Now consider the definition of "best" where $u*$ and $v*$ is the point where no further improvement (reduction) in $J1$ can be made without an increase in $J2$, and conversely, no further improvement

(reduction) in $J2$ can be made without an increase in $J1$. This is a Pareto (cooperative) equilibrium point [Luce, 1989].

There are two problems associated with the Pareto equilibrium concept. 1. Should UAV 1 choose another action $u_$ and deviate from $u*$ whereas UAV 2 sticks to his Pareto choice of $v*$, then $J2(u_, v*) > J2(u*, v*)$ and $J1(u_, v*) < J1(u*, v*)$. Now UAV 1 did better, at the expense of UAV 2. 2. In general, Pareto equilibria are not unique and therefore an agreement is needed or side conditions imposed on the actual Pareto equilibrium used.

If both vehicles cooperate and agree to play the Pareto equilibrium solution, then they both can do better than the outcome provided by the Nash equilibrium. This is the benefit of cooperation. However, the "agreement" to play Pareto must be rigidly enforced; else one side can choose an action that results in a one sided benefit to one of the teams, at the expense of the other. If the "agreement" cannot be rigidly enforced, then the players are better off playing Nash, because at least they'll get the guarantee. The latter can be computed ahead of time, before the game is ever played. One might then decide whether it's at all worth playing the game.

In the context of human behavior, Pareto games provide an incentive to cheat [Deissenberg, 2001]. Hence, the "contract" must specify a penalty for one of the parties breaking it. If the parties are strictly self interested, an expected cheating value calculation can be made which is a function of [Reward – Penalty $*$P(caught|cheat)]. Of course, the other party in the agreement can make the same calculation and both could violate the agreement, which means that both parties could end up with less than the Nash (non-cooperative) value. This much studied predicament is referred to as the "prisoners' dilemma" [Luce, 1989].

It is not at all clear if these considerations have much bearing on UAV team performance, but they abound in human teams. For example, in team sports each of the members share the objective of his team winning the game by accumulating more points than the opposing team. There are a series of plays and roles that are agreed on that could be considered a Pareto solution. However, one of the players might have a different objective that is not revealed upfront. That is, his objective is to maximize the points attributed to him, not necessarily to win the game. His team mates stick to the playbook, he scores more points, he wins (becoming more valuable), and his team loses.

Concerning adversarial behavior in an Unmanned Aerial Vehicles (UAVs) team, consider a network (team) of geographically distributed assets that have a range of overlapping capabilities. These assets service targets as well as provide services to other UAVs. These services have a variety of values and associated costs. Each UAV attempts to provide the highest valued services at the lowest cost [Rasmussen, 2003]. Still, if each of the vehicles, driven by its above stated self interest, engages in the pursuit of maximizing value and minimizing cost, then, under mild assumption and in realistic non-contrived scenarios, this should cause minimal churning and lead to maximizing the value for the team. In the context of UAV teams, situations where the UAVs have very different objectives and consequently self interest at heart, exhibit predatory behavior toward each other, as well as actively practice deception, are not envisaged.

#### 4.4.6.8 Distributed Decision and Control Systems

Figure 4.12 introduced a classification of distributed decision and control systems [Bertsekas, 1989; Smith, 1981]. The "centralized" quadrant represents classical centralized control [Dantzig, 1963; Bertsekas, 1995]. The complete state information from the distributed UAVs is sent to a center where the Decision Maker (DM) resides. No independent action is taken by the UAVs. This control concept can render optimal control action in so far as complex constraints and coupling/interactions between the

vehicles can be properly accounted for and not assumed away. Algorithms used here include dynamic programming [Murphy, 1999; Puterman, 2005, Ross, 1983], large linear programs [Dantzig, 1963] and nonlinear programming [Bertsekas, 1995]. This, in turn, causes centralized control not to scale well due to the curse of dimensionality. In addition, a centralized optimal control structure might suffer from fragility and a lack of robustness to e.g., missing data.

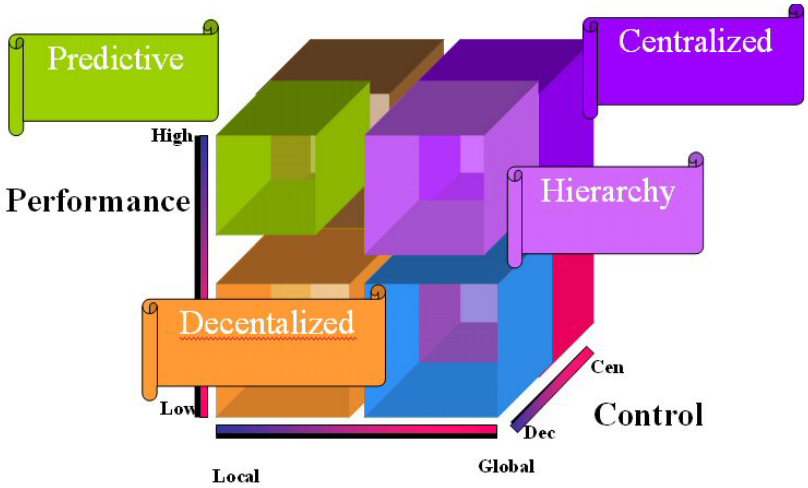


**Figure 4.12: Team Decision and Control Metrics.**

The "hierarchy" quadrant is where decomposition is prevalent. Motivated by the structure of organizations, hierarchical control [McLain, 2005] schemes are used – see Figure 4.13. The UAVs send the local vehicle state information to a higher level DM. The team members have limited global information, but they send their individual cost functions to the higher level DM. Consequently, an optimal assignment can be made by the DM that is beneficial to the team as a whole. While a degree of robustness is achieved, it is difficult to decompose the control problem and maintain high performance if there is appreciable task coupling. This approach is scalable, but optimality, as with the centralized solution, is not achieved.

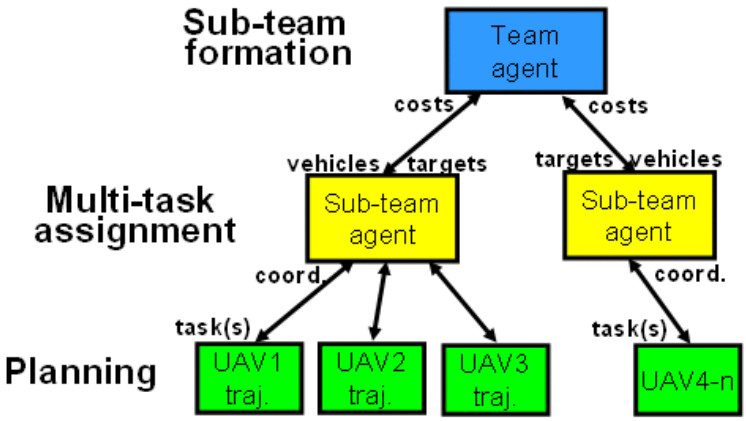


**Figure 4.13: Hierarchical Decomposition.**

Optimization techniques used here include network flow programming [Ford, 1962; Nygard, 2001; Bertsekas, 1992, Bertsekas, 1993, Goldberg, 1990], mixed integer linear programming [Schumacher, 2007; Richards, 2002; Alighanbardi, 2003; Schumacher, 2004; Richards, 2002], constraint satisfaction

[Modi, 2002; Yokoo, 2000], graph theoretic search algorithms [Nemhauser, 1999], set partition [Balas, 1976], the relative benefit method [Rasmussen, 2002], iterative auction [Bertsekas, 1992; Bertsekas, 1991; Wein, 1995; Bertsekas, 1989; Bertsekas, 1993; Kempka, 1991], negotiation, and consensus schemes [Olfati-Saber, 2003], e.g., the Paxos algorithm [Lamport, 1998].

The "decentralized" quadrant represents strongly decentralized control schemes [Bernstein, 2003] which rely on minimal global information. In the limit, there is no communication at all between the vehicles and information is only inferred about the other vehicles objectives by measuring their actions as seen through own-ship sensors. This line of work is often referred to as emergent behavior [Jadbabaie, 2003]. There is an assumption that the simple low level interactions of the vehicles will result in complex, but more importantly, optimal team behavior that meets a team objective. In general, however, this approach leads to a mediocre macro level response [Jadbabaie, 2003]. For example, the team maintains some loose cohesion, and a nominal center of mass trajectory. The team behavior appears to be complex and one marvels at the complexity brought about by a few simple rules. However, the achieved performance here is consistently low, due to the dearth of communication and consequently little or no global information. Some techniques used here include biological analogies/biomimetrics [Passino, 2005], mechanical analogies ala potential fields [Passino, 2005], and parallel auction [Bertsekas, 1991] which might cause churning.

The "predictive" quadrant is the domain of high decentralization and, at the same time, good performance. The individual team members are highly autonomous, capable of independent action, but share a common objective. Some mechanization concepts come from economics: negotiation [Wellman, 1998]; incentive games [Groves, 1973]; consensus voting [Lamport, 1998]; and distributed constraint satisfaction [Modi, 2002]. Since there is little global coordinating information, there is a high dependence on state estimation and predictive models. The better these models are at estimating future states, the higher the overall performance of the team. In general, the more coupled the various tasks the vehicles are to perform, the more complex and time consuming the arbitration to resolve the task conflicts.

#### 4.4.6.9 Complexity in Cooperative Teams

In cooperative teams an interactive decision making process between vehicles takes place, while individual vehicle autonomy is preserved. There is a continuum between centralized and decentralized control. If a fully decentralized team means no communication, then in a cooperative team the minimum level of globally communicated information that allows the desired level of team performance to be achieved, is required.

In general, the performance of cooperative control can be characterized by task coupling, uncertainty, communications delays, and partial information. The interaction of these dimensions renders cooperative optimal control a complex problem. Currently there is no working theory of cooperative systems that takes into account all these dimensions. A hierarchical decomposition is normally tried to reduce the problem to more digestible bits, but optimality is forfeited in the process. Some degree of coupling is ignored to achieve decomposition. This results in a suboptimal solution which is traded for solvability and some degree of robustness. Indeed, many times robustness comes at the expense of optimality, and vice versa. Indeed, the optimal operating point might be sensitive to changes in the problem parameters.

Team control and optimization problems are decomposed in space, time, or along function lines. Forming of sub-teams of UAVs and tasks can also be done by graph theoretic methods [Papadimitriou, 1982], set partition approaches [Balas, 1976], and relative benefit optimization techniques [Rasmussen, 2002], as well as by brute force search [Rasmussen, 2003]. The sub-team optimization problem (see Figure 4.13) then reduces to the multiple assignment problem; determine the task sequence and timing, for each team member, that satisfies all the constraints while minimizing an overall team objective function. The individual vehicles then perform their own task planning and send coordinating information,

preferably a sufficient statistic, around the network or to a team leader. Algorithms for constrained multiple task assignment include: heuristic search, e.g., branch and bound, Tabu search, or genetic algorithms [Balas, 1996]; generalized assignment [Burkard, 1998]; linear programming [Dantzig, 1963]; iterative network flow [Bertsekas, 1992; Goldberg, 1990]; and iterative auction [Bertsekas, 1988; Kempka, 1991]. One of the primary contributors to the complexity of multiple assignment is task coupling in the face of floating timing constraints – the latter brings in aspects of job shop flow optimization, or scheduling.

#### 4.4.6.10 Task Coupling

UAV team missions such as suppression of enemy air defences and wide area search and destroy are dominated by coupled tasks with floating timing constraints. There are a number of issues involved in solving the multiple assignment problem in a cooperative framework. Chief among these is the ability to decouple assignment from path planning for specific tasks. This means that tasks and path plans are generated to determine costs that are then used in the assignment process. The assumption is that these calculations are still valid after the assignment is made. This is even more so for tour sequences. Unless all possible tours are generated, sub-optimality is being ignored when chaining together tasks.

Also, decoupling assignment from timing is often done. For example, task tours are assigned first. Then the task order and precedence are enforced. This can be done myopically to set the task time using the earliest task that needs to be done, then the next, etc.; or the task times can be negotiated between the vehicles until a set of task times is arrived at that satisfies all the timing constraints. The assumption here is that these task times will not have a different, closer to optimal assignment. The decomposition assumptions to address coupling may lead to infeasibility, where all the tasks can not be assigned, as well as a significant degree of sub-optimality a.k.a., poor performance. If the task coupling is strong, decentralization is a bad idea [Rasmussen, 2003] – optimality is sacrificed, the algorithm might induce churning, and worse, feasibility is not enforced.

#### 4.4.6.11 Uncertainty

Some cooperative team problems can be dominated by uncertainty rather than by task coupling. This is true for those missions where the target identification, target localization, threat identification, and threat location are not known in advance. Some of this information may be known, while the rest is estimated using a priori given probability distributions. The challenge is to calculate the expected future value of a decision or action taken now. For example, if the UAVs use their resources on targets now, there may be no reserves left for targets that are found later and that have even higher value [Girard, 2006; Freeman; Ross, 1983]. At the same time, actions taken now might decrease the level of uncertainty. The latter can be gauged using information theoretic concepts. Possible choices are to myopically follow the decision path of least risk, or follow the decision path that maximizes the possible options in the future. Of course, the safest and middle of the road decisions are not generally the best. Furthermore, one source of uncertainty is associated with the actions of an adversary in response to an action taken by the UAVs. Possible approaches to account for uncertainty are stochastic dynamic programming [Puterman, 2005; Ross, 1983], Markov decision processes [Lovejoy, 1991; Puterman , 1994], Bayesian belief networks [Jensen, 1996], information theory [Gallager, 1968], and, in the case of no information – game theory [Vorobev, 1977; Luce, 1989].

#### 4.4.6.12 Communication

The basic premise of cooperative control is that the UAVs can communicate whenever and as much as they need to. All networks incur link delays, and if these delays are sufficiently long compared to the time between events (see Figure 4.14), they can completely nullify the benefits of team cooperation (cooperative control). Recall that control system delays in the feedback path are conducive to instability.

A critical choice here is whether the team decisions are synchronous or asynchronous. Synchronous implies that there is a common (and up to date) data base accessible to all the vehicles. If an event occurs locally, the event and all associated information is shared across the network and a decision based on the new event cannot occur until this happens. Under this protocol, a slow actor can slow down the whole team and compromise time critical tasks. Strategies are needed to maintain team coherence and performance. Asynchronous decision protocols, while more robust to delays, are much more difficult to verify in order to prove correct operation, are susceptible to inconsistent information across the network, and can lead to decision cycling/churning and, what's worse – infeasibility. The higher the rate of occurrence of events, the more difficult these problems become because the input's frequency exceeds the system's "bandwidth". Some useful protocols include consensus voting [Olfati-Saber, 2003], parallel computing [Bertsekas, 1989; Bertsekas, 1995], load balancing [Bertsekas, 1989], job shop scheduling [Sycara, 1996], and contract nets [Smith, 1981; Sandholm, 1993]. However, with false information and sufficient delays, consensus may never be reached. Indeed, false information strongly negates the benefits of cooperative control. The situation is somewhat analogous to the feedback control situation: feedback action is superior to open loop control, provided the signal to noise ratio in the measurement is high. One then takes advantage of the benefit of feedback. If however the measurements are very noisy, one might be better off ignoring the measurements and instead opt for open loop (feed-forward or model-based) control.

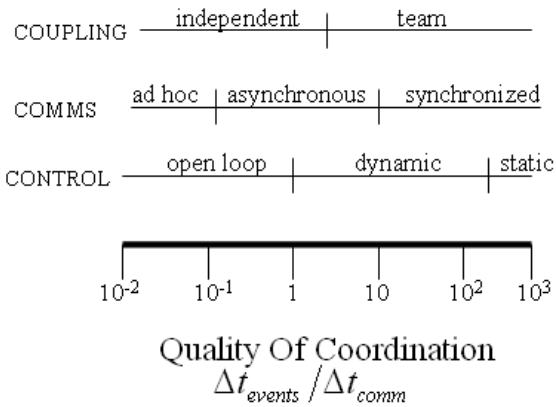


**Figure 4.14: Notional Coordination Metric.**

#### 4.4.6.13 Partial Information

Decentralized control [Kuhn, 1953] and cooperative teams are characterized by limited or partial information. The full information state is not available anywhere in the network. Worse, the information pattern is not nested. The challenge is to perform the tasks cooperatively and achieve a degree of optimality under limited and distributed information.

A specified level of team performance and team coherence requires a minimum level of shared information, e.g., the team objective function, a subset of the relevant events – e.g., pop up target information, and part of the state vector, e.g., the fuel state of the UAVs. There should be sufficient information to ensure that all the tasks are accounted for and the tasks are consistent.

This can be considered the "sufficient statistic". Also, the vehicles may have state estimation and prediction models to provide information that is not available locally. Interestingly, the vehicles may have different objective functions as well as inconsistent and delayed information that can result in conflicts that need arbitration or negotiation. False information (see Figure 4.15), in particular, can induce poor performance in a cooperative team – one may be better off using non-cooperative control.

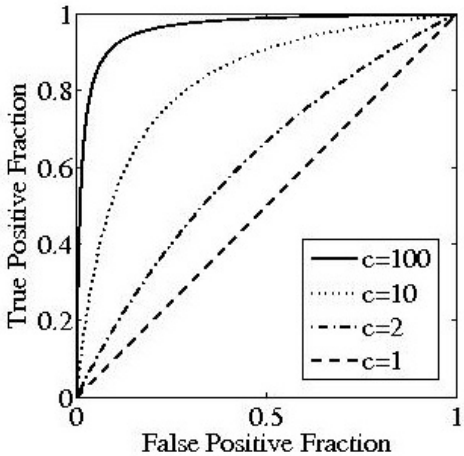


**Figure 4.15: Family of Receiver Operating Characteristic (ROC).**

For a fixed parameter *c* in Figure 4.15, the operating point on the ROC determines both the probability of correct target classification, and the probability of a false positive. The parameter is set by the flight condition and the operating point is determined by the threshold setting in the sensor.

#### 4.4.6.14 Operator

Fundamental to the field of cooperative control is the level of realized team autonomy. A continuum of team autonomy levels is possible, from completely autonomous action from release, to human operator management by exception, to human operator management by permission – the latter being the least autonomous. These different autonomy levels can exist as a function of engagement phase or current task, and dynamically change as a function of system state. Furthermore, an autonomous team is a network of distributed functionality, where for example an operator could provide the critical target recognition functionality (difficult to accomplish "by machine"), interface to outside systems, and/or provide supervisor functions. Autonomous target recognition is a long range goal and it therefore makes sense to assign the object classification task to a human operator.

In the context of stochastic cooperative control, the operator can introduce significant uncertainty. Chief among these are operator errors, e.g., object classification errors, and delays. If the team consists of a number of vehicles all sending sensor streams to an operator, the operator's workload can be high, which would increase the operator's delay and error rate. For time critical UAV operations a high degree of automation of team decision and control is required. To this end, a model of the human operator's performance is needed [Pew, 1998], e.g., the probability distribution function of the operator delay or the operator's error rate – as in a Poisson process model [Parzen, 1960]

Close coordination of a UAV team, including the operator, can be maintained in spite of significant communication delays and processing delays associated with operator cognition phenomenology, operator workload, and operator errors. These operator characteristics must be captured in a stochastic model and incorporated into a controller obtained by solving a stochastic program. This enables robust team coordination to be maintained despite a significant level of uncertainty brought about by operator misclassification of objects of interest, as well as delays.

#### 4.4.6.15 Adversary Action

Much of the research done to date on cooperative teams has been with passive targets, not allowing for intelligent adversary action.

Although threats were incorporated into simulation models, the targets did not react to actions taken by the UAV team in the work described herein. While in some simulation studies threats do react to UAV actions, behavioral/reactive modelling is routinely used, namely, the threats' reactions are pre-programmed and the strategy is known ahead of time to the UAV team. For example, should a UAV penetrate a missile engagement zone, it will be fired upon. For the known missile emplacements, UAV trajectories are planned around the zone ahead of time. If there is an approximate distribution of threats, then UAV trajectories are planned based on the threat expected locations. Unplanned for threats encountered by the UAV trigger a fixed set of UAV responses, such as evasion, or a deviation around the new threat is replanned. Addressing adversary action on the fly is fundamentally different, in that the intelligent adversary observes the state of the engagement, is cognizant of the attacker's capabilities, and solves for his optimal strategy. Here a UAV or team of UAVs must derive a strategy based on the possible actions the adversary may take for every action that the UAV team may take. This two sided optimization problem explodes exponentially for increasing sequences of action-reaction "moves". This can be cast as a dynamic programming problem if all the possible action/reaction pairs are known. Such two sided optimization problems are solvable offline for a modest number of stages, a.k.a., a few moves deep, so that the computed optimal control strategy can be implemented online.

Making the situation even more complex is the possibility of adversary deception such as the employment of decoys, diversions, traps, and false information. For example, the employment of decoys complicates the operator's classification task and thus causes an increase in the rate of false positives. This, in turn, depletes the team's resources, adversely affecting the team's performance.

#### 4.4.6.16 Algorithms for Cooperative Control

Following is a partial list of algorithms that could be used for cooperative control of a team of UAVs. We refer to the relative benefit optimization method [Rasmussen, 2002], network flow [Nygard, 2001], iterative network flow [Rasmussen, 2003; Goldberg, 1990], iterative auction [Bertsekas, 1992; Bertsekas, 1998], decomposition (assign, then time) [Rasmussen, 2003], parallel auction [Wein, 1990], combinatorial or bundle auction [Guo, 2001, Mixed Integer Linear Programming (MILP) [Schumacher, 2007; Richards, 2002; Schumacher, 2004; Richards, 2002], Partially Observable Markov Decision Processes (POMDP) [Lovejoy, 1991], Bayesian belief networks [Jensen, 1996], Genetic Algorithm or generalized search [Rasmussen, 2003; Passino, 2005], centralized or distributed constraint satisfaction or optimization [Yokoo, 2000], stochastic dynamic programming [Puterman, 2005; Ross, 1983], job shop scheduling [Sycara, 1996], vehicle routing [Toth, 2002; Reinelt, 1994], parallel computing [Bertsekas, 1989], voting or consensus [Olfati-Saber, 2003; Lamport, 1998], contract nets [Smith, 1981], games [Vorobev, 1977; Luce, 1989], receding horizon to periodically reevaluate the strategy [Cassandras, 2002; Mayne, 1990], and multiple agent systems [Wellman, 1998].

While not exhaustive, this is a good cross section of the available options for cooperative control algorithms synthesis. For strong task coupling, the centralized algorithms such as MILP and dynamic programming can give optimal solutions for problems that are computationally tractable. For scenarios with weak task coupling, network flow and auction protocols are reasonable approaches, while for scenarios with an intermediate level of task coupling one can call on iterative methods, including relative benefit. Many forms of uncertainty can be accounted for using stochastic dynamic programming, POMDP, or Bayesian belief networks. Some of these can also account for various levels of coupling, but they are not readily decomposable. More strongly decentralized approaches such as distributed constraint satisfaction and parallel auction cannot easily capture coupling among different tasks. Asynchronous parallel computing may address some of the communication issues previously discussed.

In summary, there is no one approach that addresses all the manifold facets of cooperative control. The development of heuristic methods and iterative schemes geared to addressing the dominant traits and the specifics of a particular operational scenario is required.

#### 4.4.6.17 Coupling of Tasks vs. Decentralization vs. Communication

Each of the distributed decision and control approaches or algorithms discussed, as applied to the cooperative team problem, has certain advantages which can be brought to bear on the team problem at hand. A notional trade space is shown in Figure 4.16, where strength of task coupling, communications volume/bandwidth, and the degree of decentralization feature. The corners of the cube, labeled 1 – 8, show those approaches that appear to be more suitable to address the problem characteristics represented by that node. The origin, or node 3, is typified by: weak task coupling or few constraints; full information, that is, a centralized or nested information pattern; and very low communication costs – high bandwidth, near zero delays, and complete connectivity. The single assignment optimization problem with binary decision variables and simple constraints can be posed as a network flow problem. This class of integer programs can be solved using slow relaxation techniques, or fast specialized binary tree search algorithms. The control can be partially distributed through implicit team coordination, which is where the centralized solution is computed, redundantly, by each of the UAVs.

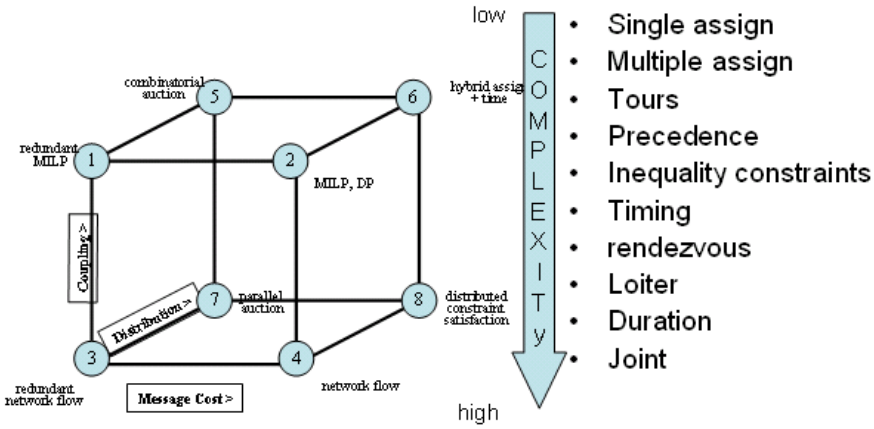


**Figure 4.16: Coupling–Decentralization–Communication Trade Space.**

As the cost of sending messages increase, the need for redundancy could motivate us to use the centralized network flow approach shown at node 4. Concerning node 3: as the tasks become more strongly coupled and, particularly, when floating timing constraints are included, the network flow solution is not suitable. MILP can rigorously include both integer and continuous constraints. While a full information (centralized) scheme, the same approach can be used to have the solution computed redundantly by each of the vehicles, as at node 1. As the cost of sending messages increases, the need for a degree of redundancy leads one to use the centralized MILP algorithm, which could also be formulated as a dynamic program at node 2. Receding horizon optimization can reduce the computational burden, enhancing scalability, and also reduce the need for information about the environment, but at the expense of optimality.

One objective of cooperative teams is to distribute the control so that the UAVs can operate more autonomously, but cooperate in performing team tasks. Decentralization increases in moving from node 3 to node 7. The Jacobi auction [Kempa, 1991] iterative algorithm, while intuitively appealing, is, in its basic form, mathematically equivalent to the network flow optimization problem. This iterative optimization scheme does not require full information locally, but it does require a centralized auctioneer with an attendant increase in the message load. Continuing to node 7, the centralized auctioneer is eliminated in the asynchronous parallel auction scheme. Again, this is feasible because only the simple constraints (coupling) of network flow are active, and the very low message cost does not incur a penalty on iteration (bidding). The full benefits of parallel computing are not achievable, since targets cannot be processors.

As the task coupling strength increases at node 7, the simple constraints easily accommodated using parallel auction based algorithms, are no longer appropriate. Combinatorial auction at node 5, where bundles of goods are traded, is a mechanism by which more extensive task constraints can be included: coupled tasks are bundled together during the UAVs bidding phase. In the worst case, every feasible permutation could be a separate object to bid on. Preparing such bids may require full information. Timing constraints cannot be readily incorporated, and combinatorial auctions cannot be directly layered on top of a parallel auction. From node 7, as communication costs increase, parallel auction – based protocols are no longer appropriate, since the latter are contingent on near instant messages. Distributed constraint satisfaction type resource allocation algorithms at node 8 may be preferable because less messages are needed, and they are strongly decentralized. Only simple constraints can be accommodated here, however.

Finally, at node 6, we have the most challenging case of strong coupling, minimal communications, and strong decentralization. No one approach or solution could address these requirements, and, in some cases, no suitable approach currently *exists* which addresses all the requirements, since, in some sense, they are contradictory or mutually exclusive. One practical strategy is to apply a network flow, auction, or parallel auction algorithm iteratively for a fixed (receding) horizon. While not optimal, this allows more complex constraints to be incorporated, including timing constraints. This is a two stage strategy, which partially decouples task assignment from task planning. The minimum deviation task times allow the vehicles to refine their trajectories (or bids) to meet the team task constraints.

## 4.5 ARTIFICIAL INTELLIGENCE

Historically artificial intelligence (AI) research focuses on the simulation of human-intelligence; today this is called strong AI. Modern AI though is concerned with producing useful machines to automate human tasks requiring intelligent behavior. One area of modern AI is AI robotics which concentrates on bringing the software intelligence to the hardware realization. In practice the machines are developed to work autonomously utilizing the least human interaction possible. In order to help AI robotics, several competitions are held challenging different areas of the research, such as legged robots, wheeled robots, flying robots, all in different sizes and also human like robots to name a few.

The research focuses on developing an autonomous flying robot and since this area is a very new field of research it is necessary to look at what the other fields have achieved.

The closest to the UAV application are the ground robots which are discussed in the literature exhaustively.

### 4.5.1 Architecture

The first subject to address is the robot architecture which is well described in [Coste-Manière]. Basically in the process of designing an architecture for robotic systems several requirements have to be set, from computational power restrictions through hardware arrangements to modularity for future enhancements.

### 4.5.2 Uncertainty

The greatest challenge in AI robotics is posed by the operation in unknown environment as described in [Saffiotti]. Industrial robots are capable of highly complex actions but they lack the reaction to changes in the environment; in other words they operate in an environment that is assumed to be completely known to them.

### 4.5.3 Deliberative Architectures

The early robotics scientist tried to solve the challenge of handling uncertainty by developing systems that would gather information about (sense) the environment and react to changes in it by planning their sequence of actions accordingly, hence the name deliberative architectures. These systems showed their disadvantage immediately because of the very limited computational power at that time. The moment an event occurred (a deviation from current states and 'planned current' state) the replanning had to be initiated and thus it took a relatively long time to carry out a mission. A sophisticated deliberative architecture is presented in [Saffiotti].

### 4.5.4 Reactive Architectures

A new wave of research started with the introduction of reactive robotic architectures which lacked the world model and therefore the exhaustive planning and acted 'instinctively'. These robots were impressively fast in carrying out missions but having no model of the environment prevented them from being able to achieve complex tasks.

### 4.5.5 Three-Layer Architectures

A natural evolution of architectures was to create hybrid systems capable of both reactive and deliberative behaviors. These are the three-layer architectures described in detail in [Gata]. Today most of the architectures are based on the three-layer concept but differ in accordance to what they are designed for.

An excellent overview of existing architectures in respect to an aerial vehicle implementation is presented in [Burridge].

### 4.5.6 Cognitive Architectures

A different approach originates from the cognitive science. Researchers developed cognitive architectures that emulate human behavior. These are based on observing human behavior and possess a structure accordingly. These systems evolved into highly complex structures and are widely used on different fields of AI research; the most popular being Soar described in [Lehman]. The use of cognitive architectures for aerial combat simulation which is closely related to the problem of coordination of unmanned vehicles is presented in [Jones]. The serious disadvantage of these systems is their need for massive computational power.