# When Systems Engineering Fails --- Toward Complex Systems Engineering

**Yaneer Bar-Yam**
New England Complex Systems Institute
Cambridge, MA, USA
yaneer@necsi.org

**Abstract -** *We review the lessons learned from problems with systems engineering over the past couple of decades and suggest that there are two effective strategies for overcoming them: (1) restricting the conventional systems engineering process to not-too-complex projects, and (2) adopting an evolutionary paradigm for complex systems engineering that involves rapid parallel exploration and a context designed to promote change through competition between design/implementation groups with field testing of multiple variants. The second approach is an extension of many of the increasingly popular variants of systems engineering today.*

**Keywords:** complex systems, complexity in engineering, systems engineering

## 1 Large Engineering Projects

The traditional approach to large engineering projects follows the paradigm established by the Manhattan project and the Space program. There are several assumptions inherent to this paradigm. First, that substantially new technology will be used. Second, the new technology to be used is based upon a clear understanding of the basic principles or equations that govern the system (i.e. the relationship between energy and mass, $E=mc^2$, for the Manhattan project, or Newton's laws of mechanics and gravitation $F=-GMm/r^2$ for the space program). Third, that the goal of the project and its more specific objectives and specifications are clearly understood. Fourth, that based upon these specifications, a design will be created essentially from scratch and this design will be implemented and, consequently the mission will be accomplished.

Large engineering projects today generally continue to follow this paradigm. Projects are driven by a need to replace old "obsolete" systems with new systems, and particularly to use new technology. The time line of the project involves a sequence of stages: a planning stage at the beginning giving way to a specification stage, a design stage, and an implementation stage. The various stages of the process all assume that managers know what needs to be done and that this information can be included in a specification. Managers are deemed successful or unsuccessful depending on whether this specification is achieved. On the technical side, modern large engineering projects generally involve the integration of systems to create larger systems, their goals include adding multiple functions that have not been possible before, and they are expected to satisfy additional constraints, especially constraints of reliability, safety and security.

The images of success in the Manhattan and Space Projects remain with us. What really happens with most large engineering projects is much less satisfactory. Many projects end up as failed and abandoned. This is true despite the tremendous investments that are made. A collection of such project failures is shown in Table 1 with costs ranging from around $50 million to $5 billion, and the final one, an automation project for dispatching of London Ambulances may have cost 20 lives before it was stopped after 48 hours. Each of these projects represents a substantial investment and would not have been abandoned without good reasons. The largest documented financial cost for a single project, the Federal Aviation Administration (FAA) Advanced Automation System was the government effort to improve air traffic control in the United States. Many of the major difficulties with air traffic delays and other limitations are blamed on the antiquated / obsolete air traffic control system. This system, originally built in the 1950s, used remarkably obsolete technology, including 1960s mainframe computers and equipment based upon vacuum tubes [14], with functional limitations that would compel any modern engineer into laughter. Still, an effort that cost 3-6 billion dollars between 1982 and 1994 was abandoned without improving the system.

When a large project like the redesign of the air traffic control system fails, participants and observers can often give reasons for the failure. Successful projects that are superficially similar (but do not involve the same level of complexity) seem to indicate that specific problems were responsible. In this case there are several good reasons for failure that appear unique. Specifically, the U.S. Government procurement process that involved both the FAA and Congress has been blamed. Other problems were that the specifications / requirements were not really known, that it was designed around a "Big Bang" change that would change the system from the old to the new over a very short time, that there was an emphasis on changing from manual to automated systems, and the “safety veto” exercised by air traffic controllers who could refuse the change because of their concerns about safety. The latter indeed appears to be a daunting challenge since the safety of airplanes full of people is a major concern that is not present in many other large engineering projects. While people have attributed the failure of the Advanced Automation System to these problems, the magnitude of failures of the large engineering projects in Table 1, and the

suggestion that each case involved its own unique reasons does not seem to strike at the core of the causes of failure.

Table 1: List of Large Engineering Project Failures*

| System Function – Responsible Organization | Years of Work (outcome) | Approx. Cost M=Million, B=Billion |
|---|---|---|
| Vehicle Registration, Drivers license – Calif. DMV [3,10,23,24,39,40] | 1987-1994 (scrapped) | $44M |
| Automated reservations, ticketing, flight scheduling, fuel delivery, kitchens and general administration – United Air Lines [27] | Late 1960s–Early 1970s (scrapped) | $50M |
| State wide Automated Child Support System (SACSS) – California [12,37] | 1991-1997 (scrapped) | $110M |
| Hotel reservations and flights – Hilton, Marriott, Budget, American Airlines [26] | 1988-1992 (scrapped) | $125M |
| Advanced Logistics System – Air Force [38] | 1968-1975 (scrapped) | $250M |
| Taurus Share trading system – British Stock Exchange [16] | 1990-1993 (scrapped) | $100–$600M |
| IRS Tax Systems Modernization projects [34] | 1989-1997 (scrapped) | $4B |
| **FAA Advanced Automation System [35]** | **1982-1994 (scrapped)** | **$3–$6B** |
| London Ambulance Service Computer Aided Dispatch System [30] | 1991-1992 (scrapped) | $2.5M, 20 lives |

*with thanks to J. Saltzer for providing some of the references.

A study of government Information Technology projects in 1994 [13] pointed to waste throughout the Government, including a number of DoD projects. This led to the Information Technology Management Reform Act (ITMRA) part of the Clinger-Cohen Act in 1996. The objective of this Act was to bring strategies that were in use in the private sector into government acquisition processes. It is useful, therefore, to ask whether indeed the private sector had greater success in large engineering projects.

A general survey of large software engineering projects was performed in 1995 by the Standish Group International [32]. This study classified projects according to whether they met stated goals of the project, the time table, and cost estimates. They found that under 20% of the projects were on-time, on-budget and on-function (projects at large companies had a lower rate of under 10% success), over 50% of the projects were "challenged" which meant they were over budget typically by a factor of two, they were over schedule by a factor of two, and did not meet about two thirds of the original functional specifications. The remaining 30% of the projects were called "impaired" which meant that they were abandoned. When considering the major investments these projects represent of time and money, the numbers are staggering, easily reaching $100 Billion each year in direct costs. The high percentage of failures and the remarkable percentage of challenged projects suggest that there is a systematic reason for the difficulty involved in large engineering projects beyond the specific reasons for failure that one might identify in any one case.

Indeed despite ITMRA and related improvements, successors of the Advanced Automation System that are being worked on today, are finding the going slow and progress limited [35]. From 1995 until today, major achievements include replacing mainframe computers, replacing communications switching system, and the en-route controller radar stations. The replacement of the Automated Radar Terminal System at Terminal Radar Facilities responsible for air traffic control near airports (the Standard Terminal Automation Replacement System (STARS) program), faced many of the problems that affected the Advanced Automation System: cost overruns, delays, and safety vetoes of implementation, and was implemented in 2002 by FAA emergency decree. Still, the new equipment continues to be used in a manner that follows original protocols used for the old equipment.

A fundamental reason for the difficulties with modern large engineering projects is their inherent complexity. Complexity is generally a characteristic of large engineering projects today. Complexity implies that different parts of the system are interdependent so that changes in one part may have effects on other parts of the system. Complexity may cause unanticipated effects that lead to failures of the system. These “indirect” effects can be discussed in terms of multiple feedback loops among portions of the system [33], and in terms of emergent collective behaviors of the system as a whole [6]. Such behaviors are generally difficult to anticipate and understand. Despite the superficial complexity of the Manhattan and Space Projects, the tasks that they were striving to achieve were relatively simple compared to the problem of air traffic control. To understand complexity of Air Traffic Control (ATC) it is necessary to consider the problem of 3-dimensional trajectory separation --- ensuring the paths of any two planes do not intersect at the same time; the many airplanes taking off and landing in a short period of time; and the remarkably low probability of failure that safety constraints impose. Failure in any one case may appear to have a specific cause, but the common

inability to implement high cost systems can be attributed to their intrinsic complexity.

While the complexity of engineering projects has been increasing, it is important to recognize that complexity is not new. Indeed, engineers and managers are generally aware of the complexity of these projects and have developed systematic techniques to address them. There are several strategies that are commonly used including modularity, abstraction, hierarchy and layering. These methods are useful, but at some degree of interdependence they become ineffective. Modularity is a well recognized way to separate a large system into parts that can be individually designed and modified. However, modularity incorrectly assumes that a complex system behavior can be reduced to the sum of its parts. As systems become more complex the design of interfaces between parts occupies increasing attention and eventually the process breaks down. Abstraction simplifies the description or specification of the system. However abstraction assumes that the details to be provided to one part of the system (module) can be designed independently of details in other parts. Modularity and abstraction are generalized by various forms of hierarchical and layered specification, whether through the structure of the system, or through the attributes of parts of a system (e.g. in object oriented programming). Again, these two approaches either incorrectly portray performance or behavioral relationships between the system parts or assume details can be provided at a later stage. Similarly, management has developed ways to coordinate teams of people working on the same project through various carefully specified coordination mechanisms.

The question is why aren't these enough? An overly simple answer is that these mechanisms and techniques are hard to get right. A more useful answer addresses the basic issues in the behavior of complex systems, the effect of interdependence of parts and functional complexity of the parts and the whole system. Two theorems provide a basis for understanding the underlying problems of engineering complex systems. The first, the Law of Requisite Variety [4], relates the complexity of the engineered system to the complexity of its task. The second [6,8] states that for all practical purposes adequate functional testing of complex engineered systems is impossible.

Once we recognize these fundamental problems of designing complex systems, how can we solve them? A partial answer can be found in the process of incremental change [32]. Incremental engineering is commonly used in engineering design through the creation of improved versions of existing hardware or software. The key to this suggestion is that when a new project is started, existing systems or rapid prototypes will serve as the foundation for iterative incremental changes. These incremental changes enable experience based learning [25,36]. After many incremental changes the system can achieve substantial modification from its original form. This concept of incremental design is one step towards a more complex systems oriented approach. Recent extensions that adopt some complex systems ideas include spiral development and evolutionary acquisition [15] and adaptive programming [2]. A complex systems perspective provides a larger conceptual framework—evolution—from which to understand how incremental change can enable rapid innovation [7]. A discussion of various engineering approaches in relation to a conventional understanding of evolution is provided in Ref. [28]. This evolutionary process is most commonly associated with the formation of complex biological organisms.

# 2 Complex Systems & Innovation

The field of complex systems [6,8,9,11] provides *two* answers to failures of large engineering projects [7]. The *first* is to change objectives. Recognizing that complexity is a crucial property of engineering problems should lead planners to limit as much as possible the complexity of objectives. This is key to structuring of successful projects. The *second* is to use an evolutionary process. This becomes essential when simplification will no longer work because the function required is intrinsically complex and thus the limits of rationality and modeling imply high levels of uncertainty in function. In this case many alternative solutions can be tried in a systematic manner allowing construction of highly complex entities. This paper focuses on the second, evolutionary engineering process because most modern large engineering projects are intrinsically complex and this complexity cannot be eliminated and the desired function retained. A few remarks are made about the possibility of complexity limitation here for completeness.

## 2.1 Change objectives: Simplify when possible

The idea of limiting complexity seems obvious, but the real effort involved is to recognize what gives rise to complexity. The complexity of a task can be quantified as the number of possible wrong ways to perform it for every right way. The more likely a wrong choice, the more complex the task. In order for a system to perform a task it must be able to perform the right action. As a rule, this also means that the number of possible actions that the system can perform (and select between) must be at least this number. This is the “Law of requisite variety” [4] that relates the complexity of a task to the complexity of a system that can perform the task effectively.

## 2.2 Evolve highly complex solutions

Simplifying the function of an engineered system is not always possible because the necessary or desired core function is itself highly complex. When the inherent nature of a complex task is too large to deal with using conventional large engineering processes, a better solution is to use an evolutionary process [7] to create an environment in which continuous innovation can occur.

Evolutionary processes, commonly understood to be analogous to free market competition, are based on incremental iterative change. However, there are basic differences between evolution and the notion of incremental engineering. Among these is that evolution assumes that many different systems exist at the same time, and that changes occur to these systems in parallel.

The parallel testing of many different changes that can be combined later is distinctly different from conventional incremental engineering. The use of parallel initial exploration has been advocated in engineering [31]. However, this approach is also unlike evolution, because it leads to the selection of a single option rather than multiple parallel implementation. Multiple parallel implementation is more similar to the parallel and largely independent exploration of product improvements by different companies in a market economy, especially when there are many small companies. Another basic idea of evolution is that much testing is done "in the field"; the process of learning about effective solutions occurs through direct feedback from the environment. There are many more aspects of evolution that should be understood in order to make effective use of this process in complex large engineering projects. Even the conventional concepts of evolution as they are currently taught in basic biology courses are not sufficient to capture the richness of modern ideas about evolution [6 (ch. 6),9,18-22,29].

Many of the more recent programming strategies, e.g. spiral devleopment, extreme programming, and the open source movement, embody features of evolutionary processes. Still, a better understanding is necessary in order to reaslize the promise of evolutionary methods. The objective revolves around mimicry of the processes that promote rapid innovation through competition. The creation of an effective « artificial ecology » or « artifical economy » requires design. In and of itself, a competitive system is not self-sustaining as it tends to become stuck through monopolization, or self-destructive behavior.

To introduce the concepts of evolution it is helpful to start from the conventional perspective then augment it with some of the modern modifications. Evolution is about the change in a population of organisms over time. This population changes not because the members of the population change directly, but because of a process of generational replacement by offspring that differ from their parents. The qualities of offspring are different from their parents, in part, because some parents have more offspring than others. The process by which the number of offspring are determined, termed selection, is considered a measure of organism effectiveness / fitness. Offspring tend to inherit traits of parents. Traits are modified by sexual reproduction and mutation that introduce novelty / variation. This novelty allows progressive changes over many generations. Thus, in the conventional perspective evolution is a process of replication with variation followed by selection based upon competition. In contrast with an engineering view where the process of innovation occurs through concept, design, specification, implementation and large scale manufacture, the evolutionary perspective would suggest that we consider the population of functioning products that are in use at a particular time as the changing population that will be replaced by new products over time. The change in this population occurs through the selection of which products increase their proportion in the population. This process of evolution involves the decisions of people as well as the changes that occur in the equipment itself.

It may be helpful to point out that this approach (the treatment of the population of engineered products as evolving) is quite different than the approach previously used to introduce evolution in an engineering context through genetic algorithms or evolutionary programming (GA/EA) [17,19]. The GA/EA approach has considered automating the process of design by transferring the entire problem into a computer. According to this strategy, we develop a representation of possible systems, specify the utility function, implement selection and replication and subsequently create the system design in the computer. While the GA/EA approach can help in specific cases, it is well known that evolution from scratch is slow. Thus it is helpful to take advantage of the capability of human beings to contribute to the design of systems. The objective of the use of evolutionary process described here is to avoid relying upon an individual human being to design systems that can perform highly complex tasks. A computer by itself cannot solve such problems either. Our objective here is to embed the process of design into that of many human beings (using computers) coordinated through an evolutionary process

A modern view of evolution recognizes that the process of evolution involves ecosystems of interdependent organisms. Such networks of dependency are generally characteristic of complex systems and are present at every level: inside the organism in genomic networks and neural networks, and outside of them in food webs and ecosystems [21]. The existence of networks reflects the importance of thinking about patterns of behavior in addition to the behavior of individual components. Still, for the purpose of simplicity we can start by using the concepts of evolution as a process of reproduction with variation and selection with competition to guide our understanding of key aspects of how processes inside and between organisms take place in such networks.

Since one of the basic concepts of evolution is competition, one question that has been of concern is the origins of cooperation. This is particularly relevant to understanding the nature of networks, which include various dependencies including cooperation as well as competition. Fundamentally, it should be recognized that cooperation and competition are not counter to each other if they exist at different levels of organization [9]. Indeed, they are essential complements; cooperation at one level of organization is necessary for competition at a higher level of organization, and vice versa. This becomes apparent when we consider team sports where cooperation between players is necessary for competition between teams, and the competition between teams gives rise to cooperation between players. This multilevel perspective is different than conventional perspectives and is an essential part of the modern understanding of the evolution and development of complex systems

Another important aspect of evolution arises from considering the continued existence of bacteria at the same time as human beings. Why should bacteria, that existed long before human beings, and therefore presumably are more primitive, continue to exist? Or if they exist, why should human beings exist as well? This question points to the remarkable diversity of life that exists as a

counterpoint to the centrality of selection in the evolutionary process. A simple interpretation of selection (survival of the fittest) would seem to suggest that there should be only one type of organism. The reason this is not the case ultimately resides in the existence of diverse resources. Diverse resources account for diverse organisms because a single organism type is not well suited to consume all of the different types of resources. Even though under some circumstances bacteria and human beings can compete for the same resources, there are many times when, due to the scale of the resources or their composition, there is no direct competition. Indeed, it is hard to determine whether it is more important to consider the cooperation or competition between human beings and bacteria in the context of the many different interactions between them (including symbiotic, parasitic and pathogenic). Thus the question of whether human beings or bacteria are more evolved is not really the central question, the key question has to do with which is better at consuming which kind of resources. Again, the diversity of entities and components must be considered in developing complex systems.

A third aspect of evolution is recognizing that in complex organisms like the human being, the process of adaptation through learning can itself be considered a kind of evolutionary process. This kind of evolution is often called "mimetic" evolution. The internal process that occurs in trial and error learning involves multiple possible concepts and processes. Through this process the more effective ones are selected within the specific context or environment in which people exist. The learning that occurs through communication between people corresponds to replication of patterns of thought. The rapid pace of human social evolution can be compared with the rapid pace of bacterial biological evolution. This comparison suggests that even when large complex structures exist, the evolutionary process of change continues to be rapid through the ongoing change of internal parts.

While the development of system-wide evolutionary process is not the standard use of evolution in engineering, it can be considered an extension of how innovation actually takes place in a market place. The larger process in this case is one in which many different companies are competing and independently innovating with tests of the effectiveness of their products being seen through their adoption by people who choose which products to buy and use.

### 2.2.1 Enlightened evolutionary engineering

The basic concept of designing an evolutionary process is to create an environment in which a process of innovation and creative change takes place. To do this we develop the perspective that tasks to be performed are analogous to resources in biology. Individual parts of the system, whether they are hardware, software or people involved in executing the tasks are analogous to various organisms that are involved in an evolutionary process. Changes in the individual parts take place through intro-ducing alternate components (equipment, software, train-ing or by moving people to different tasks). All of these changes are part of the dynamics of the system. Within this environment it is possible for conventional engineer-ing of equipment or software components to occur. The focus of such engineering efforts is on change to small parts of the system rather than on change to the system as a whole. This concept of incremental replacement of components (equipment, software, training, tasks) involves changes in one part of the system, not in every part of the system. Even when the same component exists in many parts of the system, changes are not imposed on all of these parts at the same time. Multiple small teams are involved in design and implementation of these changes. It is important to note that this is the opposite of standardization—the explicit imposition of variety. The development environment should be constructed so that exploration of possibilities can be accomplished in a rapid (efficient) manner. Wider adoption of a particular change, corresponding to reproduction in biology, occurs when experience with a component indicates improved performance. Wider adoption occurs through informed selection by individuals involved. This process of "selection" explicitly entails feedback about aggregate system performance in the context of real world tasks.

Thus the process of innovation involves multiple variants of equipment, software, training or human roles that perform similar tasks in parallel. The appearance of redundancy and parallelism is counter to the conventional engineering approach which assumes specific function assignments rather than parallel ones. This is the primary difference between evolutionary processes and incremental approaches to engineering. The process of overall change consisting of an innovation that, for example, replaces one version of a particular type of equipment with another, occurs in several stages. In the first stage a new variant of the equipment (or other component) is introduced. Locally, this variant may perform better or worse than others. However, overall, the first introduction of the equipment does not significantly affect the performance of the entire system because other equipment is operating in parallel. The second stage occurs if the new variant is more effective: others may adopt it in other parts of the system. As adoption occurs there is a load transfer from older versions to the new version in the context of competition, both in the local context and in the larger context of the entire system. The third stage involves keeping older systems around for longer than they are needed, using them for a smaller and smaller part of the load until eventually they are discarded 'naturally'. Following a single process of innovation, is, however, not really the point of the evolutionary engineering process. Instead, the key is recognizing the variety of possibilities and subsystems that exist at any one time and how they act together in the process of innovation.

The conventional development process currently used in large engineering projects is not entirely abandoned in the evolutionary context. Instead, it is placed within a larger context of an evolutionary process. This means that individuals or teams that are developing parts of the system can still use well known and tested strategies for planning, specification, design, implementation and testing. The important caveat to be made here is that these

tools are limited to parts of the system whose complexity is appropriate to the tool in use. Also, the time scale of the conventional development process is matched to the time scale of the larger evolutionary process so that field testing can provide direct feedback on effectiveness. This is similar to various proposals suggested for incremental iterative engineering. What is different, is the importance of parallel execution of components in a context designed for redundancy and robustness so that the implementation of alternatives can be done in parallel and effective improvements can be combined. At the same time, the ongoing variety provides robustness to changes in the function of the system. Specifically, if the function of the system is changed because of external changes, the system can adapt rapidly because there are various possible variants of subsystems that can be employed.

The process of generational variation in biology includes sexual reproduction. This is analogous to the formation of composite structures or systems when a modular architecture is used [6]. In this context, "composite" refers to making new combinations of system modules as a method of introducing new variants. Indeed, the use of modular composite patterns is a basis for creativity in any context [6 (ch. 2)]. The importance and attention that should be devoted to establishing module boundaries reflects the non-universal nature of the functional performance of different modular architectures and their adaptiveness. Modular boundaries and encapsulation methods should be used so that interdependence between modules is simpler than dependence within modules.

The conventional division between human beings and machines should be modified in the context of thinking about evolutionary engineering processes. Human beings and the technology (computers, communication devices, electronic networks, etc.) should all be understood to be part of the system. Moreover, the process of creating system components (training, design, engineering, construction) also becomes part of the system itself. In particular, human beings are interactive agents in the process of creation (design, development) and the process of implementation, as well as in the process of system function. Similarly, computers are also interactive agents involved in the processes of design, development and function.

Evolution is a process of cyclical feedback and the role of the dynamics of this feedback often leads to a need to balance different performance aspects that are mutually contradictory. Understanding the balance needed is a current area of research and simple guidelines are not yet known. The best that can be done is to alert the manager of the evolutionary engineering process to the symptoms of effective evolutionary change so that they can be recognized and modifications "on the fly" can be made in the evolutionary environment with the objective of improving the balance. Since the evolutionary engineering process will be designed in such a way that iterative refinement of the process itself is possible, this is not a critical limitation. Indeed, this is consistent with the idea that comprehensive advance planning (as currently understood) is often not possible and that the system is designed to be effective in an adaptive process.

The central contradiction here is that the process of selection and competition after some time generally gives rise to a single dominant type that inhibits innovation. This is known as the "founder effect" in biology and sociology and as monopolization in economics. To avoid internal inhibition of change, the process must be designed to promote change and destabilize uniform solutions to problems, when it is appropriate (i.e., dictated by system performance in the context of interaction and feedback with the external environment). Such promotions of change might on the surface appear counter to the process of selection itself, since over the short term, promoting alternatives to established solutions appears to be counter to selection of the most effective system known at that time.

Another balance that must be reached is between promoting the propagation and adoption of improved systems and inhibiting propagation in order to allow sufficient time for testing. If adoption is too rapid, a solution that appears effective over the short term may come to dominate before it is tested in circumstances that are rare but important, leading to major failures when these circumstances arise. [29] If adoption is too slow, the system cannot effectively evolve, giving rise to an inhibition of change as previously noted.

### 2.2.2 Application to air traffic control

How can we apply evolutionary processes to implement change in a context where risk of large scale catastrophe is high? Our primary example will be the air traffic control system. Similar problems exist in other contexts including the nuclear power industry, and in various military contexts such as with nuclear weapons.

The problem with innovation in the air traffic control system does not appear to have been solved because we still have the "safety veto": How can we introduce changes in what an air traffic controller is doing without introducing grave risks to people in airplanes? This was the problem that eventually derailed the Advanced Automation System. Still today, the process of innovation in the air traffic control system is very slow because of a need to extensively test any proposed change. The key to solving this problem is recognizing that there already exists a process of innovation in the air traffic control system — the training of new air traffic controllers. Air traffic controllers undergo extensive, multi-stage on the job training [1]. A key one for our purposes is the stage in which the air traffic controller in training is acting as Controller, but a second Controller (supervisor) is present with override capability over the trainee. Thus, when a person is being trained, he or she performs the task under supervision with override to prevent accidents from happening. This same mechanism can be used for air traffic control innovation in hardware and software as well as in other processes. The key is to have two different stations that can perform the same functions, where one of them has an innovation in hardware or software, and the other with the more conventional system has override

capability over the first. In this case both of the air traffic controllers would be experienced controllers, not trainees. This dual system can be used to test new options for air traffic control stations while providing the same standard of safety. (Note that this dual system is not the same as the current dual system of Radar Controller and Radar Associate Controller, but is either in addition to, or possibly as a substantial modification of, this system).

There are many possible innovations that could be tested. For example, the traditional air traffic control stations consist of monochrome screens with visual sweeps of the air space. Any change in this system could introduce problems. For example, the sweeping of the screen appears obsolete compared to modern screen technology and only a residue of the limited technology that existed in the 1950s. However, a process of sweeping may be useful to keep a person alert in the context of continuous monitoring. In this case, an unchanging screen may lead to failures rather than improvements. How can this be tested safely? By introducing a version of new screens that involves continuous presentation, color displays or other changes in a trainer context. Allowing sufficient time for an air traffic controller to become used to the new system, the override capability can be retained for an extended period of time to test the system under many contexts: day, night, low and high traffic, extreme weather, etc. Such redundant execution of tasks is needed as well as maintaining older solutions that are more extensively tested. Indeed, we can expect that many variations on displays would be distracting or ineffective at bringing the key information to the attention of the air traffic controllers. Without such extensive field testing mistakes would surely be made.

The idea of using a double "trainer" has a biological justification through analogy with the double set of chromosomes that exist in humans and animals generally. The double set of chromosomes acts at least in part as a security system to buffer the effects of changes in the genome. In this case either of the chromosomes may be changed so that there are two different parallel systems that are both undergoing change. The probability of failure would be high, except that they both exist and failure of one does not generally lead to failure of function of the organism.

The overall picture of the use of such trainers is that most if not all air traffic controllers would work in pairs, where one has override capability. It is also possible to set up a double override capability to allow mutual oversight. It may be argued that the cost of having double the number of air traffic controllers is prohibitive. However, the alternative has already been demonstrated to be ineffective at the level of $3-6B in direct wasted expenses for modernization [35], while the ongoing losses to the industry on an annual basis were billions of dollars per year due to canceled and delayed flights caused by ineffectiveness of the air traffic control system.

# 3 Conclusions

The complexity of large engineering projects has led to the abandonment of many expensive projects and led to highly impaired implementations in other cases. The cause of such failures is the complexity of the projects themselves. A systematic approach to complex systems development requires an evolutionary strategy where the individuals and the technology (hardware and software) are all part of the evolutionary process. This evolutionary process must itself be designed to enable rapid changes while ensuring the robustness and safety. The systematic application of evolutionary process in this context is an essential aspect of innovation when complex systems with complex functions and tasks are to be created.

This paper has proposed that complex engineering projects should be managed as evolutionary processes that undergo continuous rapid improvement through iterative incremental changes performed in parallel and thus is linked to diverse small subsystems of various sizes and relationships. Constraints and dependencies increase complexity and should be imposed only when necessary. This context must establish necessary security for task performance and for the system that is performing the tasks. In the evolutionary context, people and technology are agents that are involved in design, implementation and function. Management's basic oversight (meta) tasks are to create a context and design the process of innovation, and to shorten the natural feedback loops through extended measures of performance.

# References


[1] *3120.4 FAA Handbook.*

[2] See e.g. Agile Software Development, *CrossTalk,* Vol. 15, No. 10, Oct. 2002.

[3] C. Appleby and C. Wilder, “Moving violation: state audit sheds light on California's runaway DMV network project,” *InformationWeek,* No. 491, p. 17, Sep. 5, 1994.

[4] W. R. Ashby, *An Introduction to Cybernetics,* Chapman and Hall, London, 1957.

[5] R. Axelrod and M. D. Cohen, *Harnessing Complexity: Organizational Implications of a Scientific Frontier,* Basic Books, New York, 2000.

[6] Y. Bar-Yam, *Dynamics of Complex Systems, Perseus,* Reading, MA, 1997.

[7] Y. Bar-Yam, Enlightened Evolutionary Engineering / Implementation of Innovation in FORCEnet, Report to Chief of Naval Operations Strategic Studies Group, 2002 (Brief, 2000).

[8] Y. Bar-Yam, “Unifying Principles in Complex Systems,” in *Converging Technology (NBIC) for Improving Human Performance,* M. C. Roco and W. S. Bainbridge, Eds., in press.

[9] Y. Bar-Yam, “General Features of Complex Systems,” in *UNESCO Encyclopaedia of Life Support Systems,* in press.

[10] J. S. Bozman, "DMV disaster: California kills failed $44M project," *Computerworld,* Vol. 28, No. 19, pp. 1 and 16, May 9, 1994.

[11] D. Braha and O. Maimon, *A Mathematical Theory of Design: Foundations, Algorithms and Applications,* Kluwer, Boston, 1998.

[12] "California State Auditor/Bureau of State Audits, Health and Welfare Agency: Lockheed Martin Information Management Systems Failed To Deliver and the State Poorly Managed the Statewide Automated Child Support System," Summary of Report Number 97116, Mar. 1998.

[13] W. S. Cohen, "Computer Chaos: Billions Wasted Buying Federal Computer Systems," Investigative Report, U.S. Senate, Washington, D.C., 1994.

[14] Committee on Transportation and Infrastructure Computer Outages at the Federal Aviation Administration's Air Traffic Control Center in Aurora, Illinois [Field Hearing in Aurora, Illinois] hpw104-32.000 HEARING DATE: 09/26/1995.

[15] DoD Directive 5000.1, "The Defense Acquisition System," May 12, 2003.

[16] H. Drummond, *Escalation in Decision-Making,* Oxford University Press, Oxford, 1996.

[17] L. J. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution,* Wiley, New York, 1966.

[18] B. Goodwin, *How the Leopard Changed its Spots: The Evolution of Complexity,* Charles Scribner's Sons, New York, 1994.

[19] J. H. Holland, *Adaptation in Natural and Artificial Systems,* 2d ed. MIT Press, Cambridge, 1992.

[20] J. H. Holland, *Hidden Order: How Adaptation Builds Complexity,* Addison-Wesley, Reading, MA, 1995.

[21] S. A. Kauffman, "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets," *J. theor. Bio.,* Vol. 22, pp. 437-467, 1969.

[22] S. A. Kauffman, *The Origins of Order: Self Organization and Selection in Evolution,* Oxford University Press, New York, 1993.

[23] R. T. King, Jr., "California DMV's computer overhaul ends up as costly ride to junk heap," *Wall Street Journal,* East Coast Edition, p. B5, Apr. 27, 1994.

[24] M. Langberg, "Obsolete computers stall DMV's future," *San Jose Mercury News,* p. 1D, May 2, 1994.

[25] G. S. Lynn, J. G. Morone and A. S. Paulson, "Marketing and discontinuous innovation: The probe-and-learn process," *California Management Rev.,* Vol. 38, No. 3, pp. 8–36, 1996.

[26] E. Oz, "When professional standards are lax: the CONFIRM failure and its lessons," *Communications of the ACM,* Vol. 37, No. 10, pp. 29-36, Oct. 1994.

[27] A. Pantages, "Snatching Defeat from the Jaws of Victory," News Scene (monthly column), *Datamation,* Mar. 1970.

[28] M. T. Pich, C. H. Loch and A. De Meyer, "On Uncertainty, Ambiguity and Complexity in Project Management" Management Science 48, 1008-1023, 2002.

[29] E. Rauch, H. Sayama and Y. Bar-Yam, "The role of time scale in fitness,"*Phys. Rev. Lett.* 88, 228101-4 2002.

[30] "Report of the Inquiry Into The London Ambulance Service," The Communications Directorate, South West Thames Regional Health Authority, Feb. 1993.

[31] D. K. Sobek, A. C. Ward and J. K. Liker, "Toyota's principles of set-based concurrent engineering," *Sloan Management Rev.,* Vol. 40, pp. 67–83, 1999.

[32] Standish Group International, *The CHAOS Report,* 1994.

[33] J. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World,* McGraw-Hill, New York, 2000.

[34] R. Strengel, "An Overtaxed IRS," *Time,* Apr. 7, 1997.

[35] U.S. House Committee on Transportation and Infrastructure, FAA Criticized For Continued Delays In Modernization Of Air Traffic Control System, Mar. 14, 2001.

[36] R. W. Veryzer, "Discontinuous innovation and the new product development process," *J. Product Innovation Management,* Vol. 15, pp. 304–321, 1998.

[37] T. Walsh, "California, Lockheed Martin part ways over disputed SACSS deal," Government Computer News State and Local, Feb. 1988.

[38] P. Ward, "Congress may force end to Air force inventory project," *Computerworld,* Vol. IX, No. 49, p.3, Dec. 3, 1975.

[39] G. Webb, "DMV-Tandem flap escalates," *San Jose Mercury News,* p. 1A, May 18, 1994.

[40] G. Webb, "DMV's $44 million fiasco: how agency's massive modernization project was bungled," *San Jose Mercury News,* p. 1A, Jul. 3, 1994.