The Art of Systems Engineering
A Chapter for the NASA Systems Engineering Handbook
By
John F. Muratore
Research Associate Professor
Aviation Systems and Flight Research
University of Tennessee Space Institute

Introduction

In modern systems engineering education, we mostly teach process. It is easy to understand why. There are many important systems engineering processes which have been developed over the years such as configuration management, requirements decomposition and hazard analysis. It is also easy to understand why engineering educators have focused on these processes. Engineers as a group like process. It is easy to teach process because it doesn't generally require innovative thinking and it is easy to tell when you are finished. Further, because many systems engineering processes are mandated on US government contracts for large systems development by NASA and the military, teaching of process is a lucrative industry for which there is continuing demand.

These systems engineering processes represent significant lessons learned and are an important part of successful projects. All good systems engineers should be knowledgeable and practiced in them. For examples, at the University of Tennessee Space Institute we teach the following processes to graduate students in Aviation Systems:

- Requirements Development Functional Decomposition and Allocation
- Requirements Traceability and Verification
- Design Review and RID processing
- Hazard Analysis
- Risk Management
- Configuration Management and Change Control
- Mass Properties Management
- Interface Control
- Trade Studies Management and Analysis of Alternatives
- Technical Performance Metrics and Key Performance Parameters
- Architecture definition and frameworks
- Technology Readiness Levels
- Natural and Induced Environments definition

However, good systems engineering consists of more than just successful execution of process. Good systems engineering has a definite component of artistry to it. This art of systems engineering is hard to define and the purpose of this chapter is to put some definition into the art of systems engineering. This chapter will use a lot of aviation examples, as aviation has been around for significantly longer than spaceflight and offers many examples of good and bad applications of the art of

systems engineering. But the subject of this chapter is applicable to all NASA systems, whether aircraft or spacecraft, ground systems or flight systems, science instruments or supporting systems.

The two natures of SE

In order to perform good systems engineering, you need to use both halves of your brain. We know from psychologists that the human brain segregates thinking activities into the brain's two halves. The left half is said to be more analytical, helping you with math, logic, and speech The right half helps you think about abstract things like music, colors, and shapes.

In systems engineering, there is a left half brain part that is about being compulsive about identifying requirements, decomposing them, tracking their verification etc… This is the part of the brain that helps you perform the process part of systems engineering. Then there is a right half brain part that is about intuitively inquiring about and understanding how all the parts of a complex system interact and engineering them to interact in desirable and predictable ways. This is the part of the brain that helps you do the art of systems engineering.

The importance of both process and art – the bathroom analogy

Good systems engineering is in many ways about using both halves of your brain to work on systems problems. It is not only satisfying to use all of the parts of your brain, it is also necessary to have successful projects.

To explain the importance of performing well in both the art of systems engineering and the process of systems engineering, I like to use an analogy to everyday life. We are taught early in life to wash our hands after going to the bathroom. It is good advice to prevent the spread of many easily prevented disease organisms. It is also a process.

But if you have a serious illness, not caused by the particular pathogens acquired in daily hygiene, things like cancer, diabetes or heart disease, you can wash your hands all day long and you are not going to stay in good health. In this case, you need more serious intervention and simple process is not going to keep you healthy.

Similarly in projects, if you have a good engineering approach in a project, keeping track of all those systems engineering processes will keep things healthy. But if you have a bad engineering approach, you can run systems engineering processes all day long and it isn't going to fix the fundamental problems. When you have fundamental engineering issues of a system nature, then you need the art of systems engineering to maker things right.

Joint Strike Fighter – Triumph of Art over Process

The Joint Strike Fighter (JSF) is an interesting example of how art can be more important than process in developing systems. The Department of Defense (DOD) ran a competition to develop the next generation strike fighter aircraft for multi-service and multi-nation use. Two major aerospace companies squared off in a multi-billion dollar fight. I was not personally involved in this process, but as best I can determine, both companies produced excellent well engineered aircraft that exactly met all of the design requirements. But they approached the problem with two different views.

The Boeing team, which produced the X-32 prototype, was optimized to meet all the requirements including the specified capabilities with the specified growth capabilities. The Boeing prototype represented total execution of a process to deliver the minimum cost and minimum risk vehicle to meet the requirements. The JSF had a requirement to perform Vertical Takeoff and Landing (VTOL) in addition to conventional runway takeoff. In order to meet this requirement, Boeing selected a propulsion technology that had been used on the previously selected Harrier aircraft. This is called direct-lift technology and involves rotating nozzles that direct engine thrust downward. This technology was low risk and met the design requirements but is relatively inefficient and did not provide much additional growth capability above the required capability and specified growth requirements.

The Lockheed Martin team, which produced the X-35 prototype, decided to pursue a more risky approach. They utilized a technique known as lift-fan which involved coupling the engine directly to a lift fan which generated the downward flow of air for VTOL. This technique is significantly more efficient however it involved a complex technology to couple the jet engine to the lift fan which had not really been used before. Thjus although this technique had growth potential above the specified capability and growth requirements, it involved a higher risk approach.
To some in the aerospace community, the X-35 also had a significantly more appealing outer moldline (partially due to the lift fan and partially due to other factors).

Boeing X-32 (left), Lockheed Martin X-35 (right)

In the end, the DOD selected the Lockheed Martin X-35. The additional growth capability provided by the lift fan as well as the outer moldline trumphed over the exact fulfillment of the DOD's specified requirements exhibited by Boeings X-32.

X-35 Powered Lift Fan

I would argue that this represented the triumph of art versus process. It represents a demonstration that the total vehicle is more than a summary of meeting requirements and that often unstated requirements are the most important. It also represents the continuing truth of the ancient and totally unscientific right half brain adage of the aviation industry "If it looks good, it will fly well".

How do we define the art of systems engineering ?

If the art is important, how do we define art. After all, most engineers are not artistically oriented. I consulted a friend of my who is an artist and although I am assured that I don't understand the process of teaching real art, my left half brain identified the use of elements of style as a way of characterizing an art. Pursuing this theme, I identified seven elements of style defining the art of systems engineering. This is not a comprehensive list and a look forward to dialogue from readers on better identification of these elements. But for the purposes of an initial list, the elements of style are captured by systems that have the following characteristics:

- Robustness
- Elegance
- Balance
- Growth Capability
- Visibility
- Reasonableness
- Complexity

In this chapter we will examine each of these elements in order to better define them. These elements represent a solid framework to use to evaluate system designs and the adequacy of systems engineering within a project or program.

Robustness

Robustness is a characteristic that describes the sensitivity of a system to boundary conditions or off-nominal conditions and configurations. Robustness answers the question as to whether a system gracefully degrades or is there nonlinear behavior of the system at its boundaries. It also describes the behavior of the system as components fail either within the system or outside the system.

There are known techniques to evaluate robustness. Sensitivity analysis, performed either by computer modeling or test, provide visibility into the performance of systems at boundary conditions. Sensitivity analysis is a key technique that should be performed during system design and verification in almost any type of system. Flight test professionals speak of “expanding the envelope”. This is basically a process of determining robustness by gradually expanding the envelope of speed and altitude over which an aircraft has flown in order to determine its sensitivity. This is an inherently hazardous activity and in modern aviation is preceded by extensive simulation in computers and ground test facilities such as wind tunnels to understand the sensitivity to boundary conditions.

One of the dangers of computer simulation is that computer models are often built with linear or simplified models of system performance and they may not be accurate at boundary conditions.

Monte Carlo analysis is a technique which can be used with sensitivity analysis as multiple parameters can be varied within distributions to determine if combinations of parameters can cause unexpected system behavior.

One of the least expensive and most powerful techniques for determining system robustness is the application of imagination informed by knowledge of basic fundamental sciences and existing predictive data and modeling for a system. System configurations can be examined to determine where the linear assumptions break down and the sensitivity at those points examined. This type of analysis should be performed on systems and subsystems at major system design points.

Characteristics that contribute to robustness are fault tolerance and margin. Systems that incorporate features which increase system fault tolerance are usually more robust than those which do not. Systems with greater margin about design requirements also are usually more robust than those with less margin. Both of these are characteristics which should be examined as a part of system design.

Fault tolerance is normally defined as the ability to withstand failure and still perform system functions. Systems are characterized by the number of failures that they can tolerate while still performing their function. Zero fault tolerant refers to those systems where any single fault can invalidate their functioning. Single fault tolerance refers to systems that can withstand a fault in any single component and still perform their function. Dual and triple fault tolerant systems can support failures in any two or three components and still perform their system functioning.

One of the major techniques in fault tolerance is system redundancy. This involves placing additional components in a system so that failures in any component do not prevent the system from being able to function. This can involve placing additional individual components such as adding a primary or secondary battery to an electrical system or placing primary and secondary pumps in a fluid loop. It also can involve adding an entire system which replicates the functioning of the first.

Redundancy can be in placing additional copies of similar systems or components in a system design or in placing dissimilar components that can accomplish the same system function. Fault tolerance of support systems such as electrical power is often improved by adding additional systems For example, in the shuttle propulsion system there are two Orbital Maneuvering Engines (OME) that are used to change orbits including the deorbit orbital maneuver for re-entry. This is an example of similar redundancy. Within each engine there system are many redundant components to improve fault tolerance but there are many components which cannot be replaced such as the combustion chamber and nozzle of each engine. In order to further improve the fault tolerance of the system, the orbiter aft facing Reaction Control Jets (RCS) can perform the deorbit maneuver to return the crew home. This is a case of dissimilar redundancy. To make this work, fuel and oxidizer feed valves are added to the system so that the RCS jets can be fed from the propellant tanks that normally support the OME.

Shuttle OMS Pod with aft facing RCS jets

In many types of systems, it is impossible to completely replicate the system. This happens commonly in structures. For example it is usually not possible to add additional wings to a vehicle. In these cases a margin based approach is used. A design limit load is

established for a system which defines the highest possible load that a system will encounter in its performance envelope. Then a factor of safety is established and the system is designed, tested and analyzed to show that it has margin above the required design limit load times the factor of safety. This approach is typically used on structures and pressure vessels in spacecraft.

In system design it is assumed that when a system is designed with a design limit load and factor of safety that it is not susceptible for failure in its environment. Structures, pressure vessels, plumbing and electrical wiring are usually dealt with in this manner and are assumed to not to be failure proof when operated under their design limits. Note that just because components are considered failure proof, it does not mean that connections between components automatically inherit that characteristic. For example, in a propulsion system, tanks and plumbing are considered failure proof when operated within their design limits but the interface between a tank and a plumbing, a weld or a fitting, can fail or leak. In electrical systems, even though wiring is normally considered failure proof, it is possible for a short circuit to occur which overstresses or ignites the wiring.

The design of the Saturn V launch vehicle tells us a lot about the importance of engineering robustness into system design, even when the requirements, however carefully determined and allocated, do not call for it. The original Saturn V first and second stage designs met all known requirements with four engines. Von Braun's team at Marshall Space Flight Center added a fifth engine to first and second stage for margin above the stated requirements, concerned that weight growth in the Apollo spacecraft would invalidate their efforts if this robustness was not added. As it turns out, Apollo would not have been possible if that performance had not been available as mass in the command/service module and lunar module grew significantly. The additional performance also enabled more science content in the later Apollo J missions than in the first few lunar missions.

Saturn V First Stage

Robustness does not necessarily have to cost significant weight or even cost. If during system design, the designers evaluate the possible interaction between components in failure modes, it is often possible to improve fault tolerance without adding complete components or systems. For example, in the design of the X-38 flight control system we were able to establish single fault tolerance in a system that did not have any replicated components. During the X-38 design process we were examining the fly-by-wire control systems. Fly-by-wire means that the computer controls the vehicle by sending electrical signals that command aerosurfaces (flaps and rudders) to move. During this process we asked what would happen if the computer interface electronics failed off. It turned out that the off state resulted in a command to the surface actuators that forced the surfaces into a hard-over (full range) condition. A few quick simulations showed that this configuration was uncontrollable. We then asked the question if the vehicle could be controllable if the surfaces failed to any other position. We quickly found that if any one surface failed to a mid-range position that the other three surfaces could compensate and keep the vehicle stable. Addition of some pull-up resistors ensured that the actuators were scaled so that the fail off state of the computer interfaces resulted in mid-range on the surfaces. This established a highly robust configuration that was able to withstand loss of computer commands to any one surface and yet maintain stable flight. It is worth noting that thus change was only possible because of the concerted effort of multiple disciplines (avionics, flight controls, flight software) working together for a system goal (stable flight in the event of failure).

X-38 In Flight

Finally margin can be added by examining where a system becomes nonlinear and then adding devices to ensure control of the system as it enters the non-linear region to ensure predictable performance. In aviation, one of the most important non-linear behaviors is the performance of aircraft as the flow over the wing separates in a stall. Over the years a number of devices have been added to warn the operator of this condition (stick shakers, stall warning horns, angle of attack indicators) or to control the stall progression (wing twist, stall strips, vortex generators, canards) so that the aircraft stalls in a repeatable and easily controlled way. It is interesting to note that the first practical heavier than air craft of the Wright Brothers employed a canard (a pitch control surface forward of the center of gravity) in order to prevent an uncontrollable stall.

Elegance

The second element of style to be examined is elegance. We define elegance in system design as the degree to which a system design reflects simple unifying principles. An elegant design is one in which a single simple unified design solves all cases in which the system is intended to operate. An inelegant design is one which requires unique and different features to deal with cases that the system faces in normal operation.

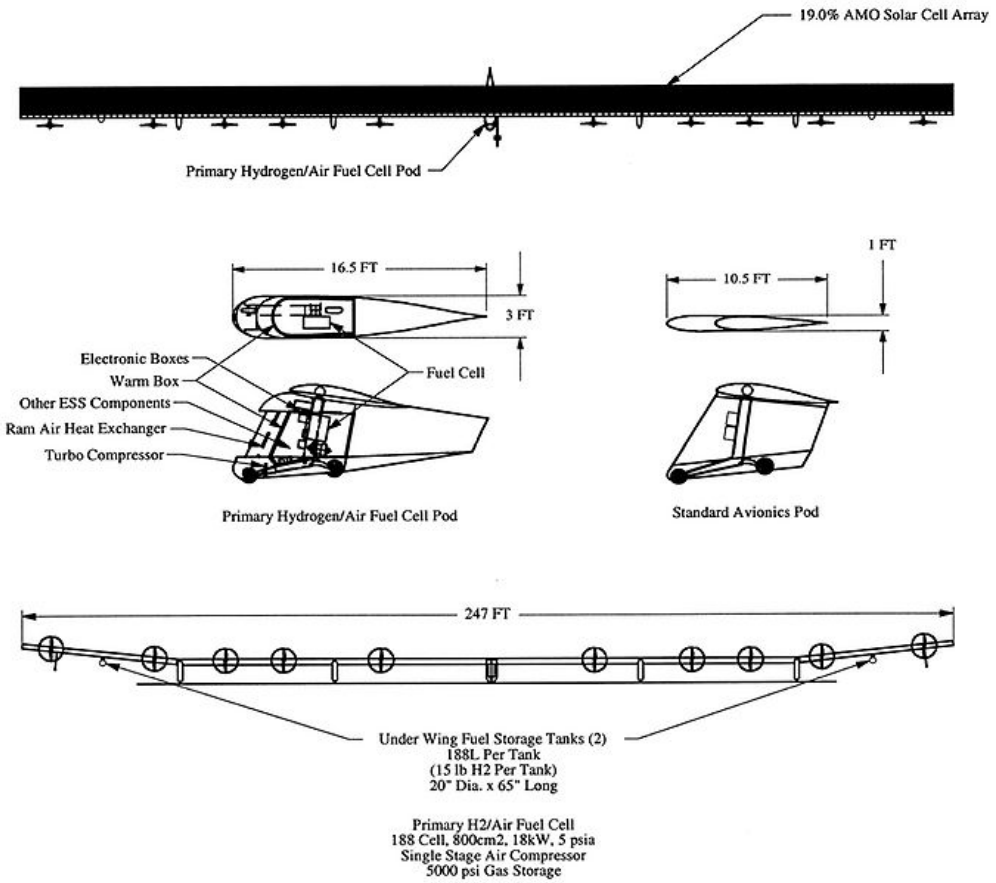
Design elegance can be determined by examination of the system design over its entire intended operating envelope. When this examination reveals that the system needs special and distinct design features to deal with different cases, the design is said to be less elegant. Requiring special design features does not necessarily mean that a design is bad or inappropriate. But as the features become more intrusive and different, the less elegant and desirable that the design becomes. Pejoratively, these special features are referred to as “kludges” (pronounced “kloodj”).

Balance

A balanced system is one where all of the disciplines of the design are considered and are engineered to work together to accomplish a common purpose. A balanced design can have certain design aspects that are paramount but all of the other aspects are engineered to account for that. For example the Voyager aircraft that flew around the world on a single fuel load was optimized for range. “No compromise for range” was Burt Rutan’s battle cry during its design and manufacture. However an examination of the design shows that propulsion, fuel system, aerodynamics, structure and flight controls were all harmonized into a balanced design that had the range to fly around the world un-re-fueled. Unbalanced designs are often proposed and many are built. But few unbalanced designs have sustained superior performance.

Voyager

The Helios unmanned solar powered research aircraft provides an interesting case in point. Funded by NASA and constructed by Aerovironment, this The Helios Prototype was an ultra-lightweight flying wing aircraft with a wingspan of 247 feet (75.3 m), longer than the wingspans of the U.S. Air Force C-5 military transport (222 feet (67.7 m) or the Boeing 747 (195 feet (59.4 m) or 215 feet (65.5 m), depending on the model). The electrically powered Helios was constructed mostly of composite materials such as carbon fiber, graphite epoxy, Kevlar, styrofoam and a thin, transparent plastic skin. On August 14, 2001, the Helios Prototype piloted remotely by Greg Kendall reached an altitude of 96,863 feet (29,523.8 m), a world record for sustained horizontal flight by a winged aircraft. On June 26, 2003, the Helios Prototype broke up and fell into the Pacific Ocean about ten miles (16 km) west of the Hawaiian Island Kauai during a systems checkout flight in preparation for an endurance test scheduled for the following month. This failure was the result of not properly balancing the structural response to gusts, the aerodynamics and the flight controls of the vehicle. As the mishap report states “The aircraft represents a nonlinear stability and control problem involving complex interactions among the flexible structure, unsteady aerodynamics, flight control system, propulsion system, the environmental conditions, and vehicle flight dynamics. The analysis tools and solution techniques were constrained by conventional and segmented linear methodologies that did not provide the proper level of complexity to understand the technology interactions on the vehicle’s stability and control characteristics. “



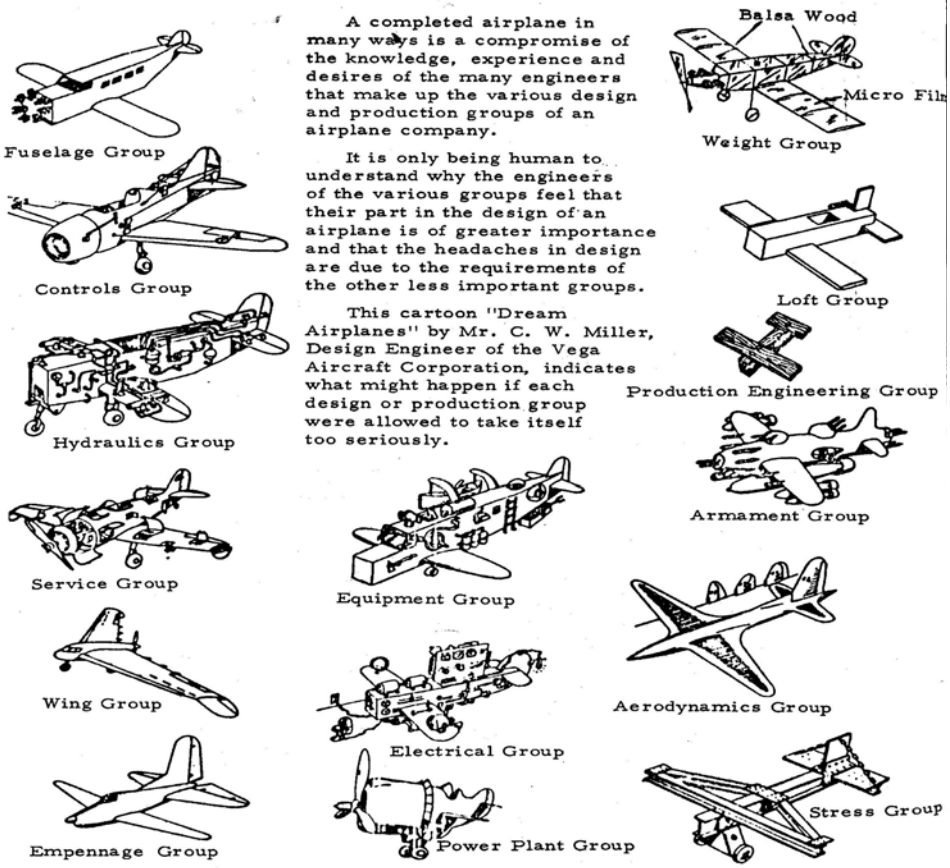
Helios configuration

NASA Dryden flight Research Center Photo Collection
http://www.dfrc.nasa.gov/gallery/photo/index.html
NASA Photo: ED03-0152-1 Date: June 7, 2003 Photo By: Carla Thomas

Equipped with an experimental fuel cell system to power the aircraft at night, the solar-electric Helios Prototype is shown during a checkout flight prior to its long-endurance flight demonstration in the summer of 2003.

Helios

Balance is hard to achieve because of the nature of discipline engineering required to build aerospace craft. As captured in the attached cartoon, most aerospace vehicles consist of products produced by different discipline specialties and they tend to see vehicles as being dominated by their contribution. Although this cartoon is amusing, my experiences on multiple air and space vehicle designs indicate that it is more often true than not. As a result, vehicles where one discipline dominates tend to occur on a regular basis. It is interesting to note that more realistic versions many of the vehicles pictured in the cartoon were actually built. The GEEBEE, a realistic version of the vehicle pictured in the Propulsion Group view, was a racing plane built in the 1930's and although optimized for propulsion, it did not have balance in its stability and control and was not a sustained world-beating aircraft. The Supermarine Spitfire, a realistic version of the aerodynamicist's dream aircraft, was successful throughout World War II although it was generally considered inferior to the North American P-51 Mustang, which although powered by the same engine, had a superior overall configuration.



A completed airplane in many ways is a compromise of the knowledge, experience and desires of the many engineers that make up the various design and production groups of an airplane company.

It is only being human to understand why the engineers of the various groups feel that their part in the design of an airplane is of greater importance and that the headaches in design are due to the requirements of the other less important groups.

This cartoon "Dream Airplanes" by Mr. C. W. Miller, Design Engineer of the Vega Aircraft Corporation, indicates what might happen if each design or production group were allowed to take itself too seriously.

Fig. 1.1 Resulting Aircraft Design if One Group Is Dominant

GeeBee

Supermarine Spitfire

The need for balanced design is why it is vitally important for systems engineers to know what is important in a given design. Not all elements of the design get the same attention or need the same amount of rigor. In a world of limited resources it is important to "sharpen your pencil" only on the important areas of the design. However all elements must be considered to ensure that they are working together instead of against each other.

P-51 Mustang

Balance must occur not only at the system level but within subsystems as well. Glenn Bugos in his book “Engineering the F-4 Phantom II Parts into Systems” talks about he need in subsystem design for continuing cycles of

- Aggregation – finding the parts (often off the shelf) to make a system function
- Disaggregation – talking them apart to identify the pieces you need
- Re-aggregation – putting them back together in a way that is optimized for a given application

There is so much good off the shelf aerospace hardware available today, and the desire to reduce development cost is so important, that we have trained a generation of subsystem engineers to aggregate as much off the shelf equipment as they can. We have not emphasized that for high performance applications you may need to disaggregate and then re-aggregate components into subsystems.

As an example, the X-38 was a prototype for the Crew Return Vehicle for the International Space Station intended to serve as an ambulance and a lifeboat for the station. It operated as a lifting body during entry and flew under a parafoil during final descent and landing. During the initial X-38 test flights we used a separate Guidance, Navigation and Control system for two phases of flight – lifting body phase and parafoil phase of flight. The parafoil GN&C was off the shelf and it allowed us to partition our

efforts. As the program progressed it was clear that the parafoil GN&C was very limited and that the weight of the separate system was not acceptable for the space test vehicle. We took apart the functions of the parafoil GN&C and integrated them with the lifting body GN&C resulting in a lighter weight system, with simpler crew interfaces and greater functionality

As another example, the telecommunications front-end of the Johnson Space Center Mission Control Center in the mid 90's consisted of close to 100 racks of electronics These systems had accumulated over time and as new functionality was required, the easiest solution to add onto the system was taken. Each of the racks required spare parts, logistics, operations and maintenance personnel. During the MCC redesign, we found that the same functions were being reproduced at many places in the architecture We repackaged the functionality into less than half of the original number of racks with common commercial off the shelf parts. This resulted in significantly reduced operations costs and greater automation and functionality.

Balance also involve mutual support between systems. In many cases, deficiencies in one subsystem can be compensated for by cooperative design in other subsystems, Two examples of this occurred during the X-38 design.

During the design of the X-38 flight control system we had initially a zero fault tolerant air data system for sensing angle of attack. The flight mechanics community realized that based on the pitch attitude and pitch rate in response to a commanded surface position that they could estimate angle of attack sufficiently to maintain control. These parameters were available from the inertial measurement system, a separate system from the air data system. We built in a system using available inertial sensors to back up the air data system

In another example. ElectroMechanical Actuators (EMAs) were used in the X-38 flight control system to move the flight control surfaces. EMAs required power to hold loads but actually back generated current under certain conditions in the same way that power is re-generated in modern hybrid cars. Initially we used current shunts to deal with the generated power, but then we learned to put the re-generated power back into the batteries. This significantly reduced battery requirements for the vehicle.

Growth Capability - Scalability , and Extensibility

Another important element of style is growth capability. It is a simple fact of life that most systems have a longer and more diverse operational life than stated in the original design requirements. When the Space Shuttle was designed, each vehicle was designed for 100 missions with a 10 year operational life. The designers would have been shocked to find out that a total of 126 missions were flown over a 30 year life. Many of the current problems of the Space Shuttle are due to the fact that the system was designed for a relatively short life and was not designed for maintenance or upgrades. It has to be remembered that when the Shuttle was designed, the United States had just built 4

generations of spacecraft (Mercury, Gemini, Apollo and Skylab) in just over a decade. The idea of a single spacecraft spanning three decades was simply out of their experience or imagination.

In fact, the entire aerospace business has gone to much longer lived systems. The basic B-52 was originally designed in the late 50's and will still be in service in 2010. All of our front-line combat aircraft are in similar extensions.

Not only is life extended, but missions change as time goes on. The B-52 for example was a long range strategic bomber. With the advent of GPS guided munitions and high performance satellite communications links with ground troops, it has evolved into an unexpected close support aircraft. Similarly the shuttle was designed for carrying deployable satellites and attached payloads. Now the shuttle is almost exclusively performing International Space Station (ISS) servicing missions. Even robotic spacecraft now are performing secondary or even tertiary missions after their primary missions are completed. Sometimes these missions were not even imagined before the spacecraft was launched.

So an essential element of style in engineering systems is ensuring the proper growth capability is incorporated into the basic system design so that it can be adapted or modified as its life is extended and its mission changes. There are three terms we use when describing the element of growth. These are excess capability, scalability and extensibility.

Systems with excess capability are the easiest to understand. If a launch vehicle has a design goal of lofting 35,000 pounds to low earth orbit and it actually can lift 38,000 pounds, it has 3,000 pounds of excess capability for growth. It is important to realize that most systems lose performance as they age. In re-usable systems, some of this is due to accumulated weight and wear and tear on components. Accumulated weight can be from minor items like dirt, paint, and debris, which nonetheless can add significant weight. In addition, as systems are upgraded or maintained, sometimes wire or equipment that is no longer functional remains in the system as it may be too difficult to remove. This is very typical in wire harnesses as sometimes it is difficult to extract wire that is no longer required from existing harnesses without causing collateral damage to other wires in a harness during the removal process. Because of this weight growth, additional performance is always required at end of life.

Scalability is the ability of a system to grow to provide additional capacity. Scalability usually involves the ability of a system to easily add components to add capability. For example a computer system that contains 2 processors nominally but that can then add additional processors to handle more workload is said to be scale-able. Scalability is very difficult to achieve in most aerospace systems as they usually do not scale well because of physics based limitations.

Extensibility is the ability to adapt a systems design to incorporate additional functions. Extensibility must be designed in from the start to be effective. Extensibility is a property

that can be achieved by aerospace systems. For example the Russian Mir space station and the International Space Stations were/are both extensible systems.

Extensibility is provided by "hooks" and "scars". Hooks are provisions in software to allow growth. For example software can be developed to be able to perform thermal control by monitoring a single temperature transducer and then sending a command to a heater. When the software is designed, it can be designed to perform control based on multiple transducers or to perform the same type of control action on multiple pairs of temperature sensors and heaters. When this software is installed in a system that initially only has one sensor and one heater but retains the capability to handle additional hardware it is said to have "hooks" to handle additional strings of sensors and heaters. Scars are provisions in hardware for adding additional functions. An extra power outlet in a power system is a simple type of scar.

Techniques exist to help maintain scalability, extensibility and growth capability in systems. One technique is to build a established standards, particularly on interfaces. Building on standards often facilitates future growth and expansion. Monitoring and managing technical budgets that contain a growth reserve during development is another way to provide future growth capability. Engineering hooks and scars to extend capability during initial design are additional techniques. Growth capability is usually limited by physics based limitations, so modeling to understand these limitations can sometimes determine how much effort should be expended into maintaining growth options.

To understand the importance of design with growth in mind, the reader should consult Glenn Bugos' excellent book "Engineering the F-4 Phantom II, Parts into Systems" published by the Naval Institute Press in 1996. This book describes the life history of the F-4 Phantom II, one of the longest lived and most successful combat aircraft of the 20th century. In his first chapter, he describes how previous successes and failures had taught McDonnell (the designer and manufacturer of the Phantom II) the importance of a multi-mission aircraft, even though the original Navy specification called only for a fleet defense fighter/interceptor. By the end of its life, the Air Force had bought three times as many aircraft as the Navy. The Air Force used these aircraft as fighter bombers, anti-surface to air missile and reconnaissance aircraft and many nations bought and built variants for a variety of tasks. The decisions to use two engines and provide for two crew members provided versatility in the basic airframe that allowed it a long life and multiple roles.

Visibility

John Aaron, the legendary flight controller of the Apollo 12 and 13 missions and a major leader in the development of the Space Shuttle flight software once said to me that "software is inherently invisible" and that to understand its function one had to carefully design in features, environments and tests to make its functions visible. I would generalize John Aaron's assertion to "All systems are inherently invisible, especially

software intensive systems". An essential element of style in systems engineering is to recognize this nature of systems and then to design visibility into systems so that their functioning can be understood.

There are many "paintbrushes" in the system artist's toolkit to help ensure systems visibility. Instrumentation is a key tool. The design of instrumentation for ground and flight test during design verification is critical to well engineered systems. It is impossible to verify that systems meet requirements without instrumentation. Given the large amount of attention given to master verification plans and requirements verification matrices in traditional process based systems engineering, it is remarkable that little focus is provided in traditional process based systems engineering into properly instrumenting systems so that they can be verified. Instrumentation is usually left to the domain of very specialized engineers whose connectivity to the test verification objectives is often tenuous. Instrumenting operational systems is even more critical. Operational instrumentation must provide visibility to operators to ensure proper system functioning while also providing sufficient diagnostic and trend monitoring data to help the maintenance organization and the sustaining engineering organization to perform their functions. Familiarity with instrumentation techniques, sensor types, their advantages and limitations is critical to the art of systems engineering. Instrumentation may consist of traditional systems such as sensors for temperature, pressure, voltage etc… or may be items such as imagery to verify system performance.

Monitoring the performance of system to compare it to predictions is a key tenet of modern systems verification and especially of flight testing. In fact, the mantra of the United States Air Force Test Pilot School at Edwards Air Force Base, the original inspiration for the term "The Right Stuff" has now become "Predict, Test and Validate".

A commitment to instrumentation also requires a commitment to record and process the data from that instrumentation in order to make it useful in decision making. The process of establishing a database of recorded instrumentation so that data can be reviewed for emerging trends is a key enabling technology for proper systems engineering of projects. The term "data mining" is used to describe the process of examining systems performance records in a postflight or non real time environment order to establish trends or determine changes in systems behavior. In addition to real time monitoring, it is vitally important to postflight compare the performance of the system to predictions and to the initial design requirements of the system. It is possible to have excellent real time monitoring of a system without a commitment to this type of closed loop trend monitoring. Lack of commitment to this process was one of the contributing factors cited by the Columbia Accident Investigation Board in the loss of the Space Shuttle Columbia. The lack of this type of long term monitoring of trends in navigation data was also established as one of the causes of the loss of the Mars Climate Orbiter spacecraft.

Alerts and warnings as well as displays and controls are additional important "paintbrushes" in the system artist's toolkit. These tools can be valuable during systems verification as well as during operations and maintenance to provide visibility into systems functioning. To properly implement these systems requires both knowledge of

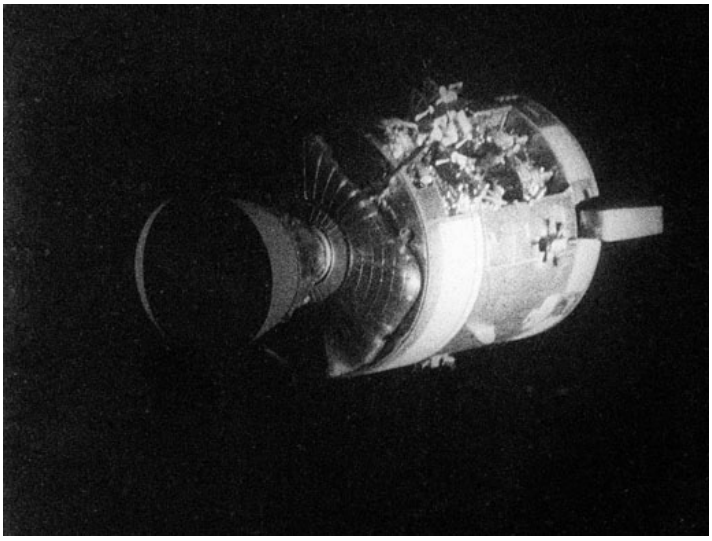
system monitoring techniques as well as human factors. Given that almost all systems ultimately interact with humans, it is also remarkable that human factors is not considered a major part of most systems engineering training and practice and is relegated to a relatively specialized discipline of human factors experts. Human factors professionals are a key resource for any successful systems artist, however the successful system artist must have sufficient knowledge in this area to recognize that interface to humans is the most pervasive interface in most systems (whether in assembly, test, operations or maintenance) and that attention to proper system visibility is critical in almost every aspect of the design. This is especially critical in the design of alerts, warnings, displays and controls which provide real time visibility into system performance.

The lack of attention to increasing visibility into system function by instrumentation and human factors in every aspect of system design can have dire consequences. At least two Airbus crashes have been blamed on what is called “mode confusion” This occurs when the operator of the system thinks that the system is in one mode of operation when it is actually in another. What is particularly interesting about these crashes is their relationship to each other. In one crash, the aircraft was approaching the runway and the pilot called for a go-around. The pilot thought that the aircraft was in Takeoff Go Around (TOGA) Mode which provides maximum thrust/maximum lift to climb away when the aircraft was actually in an approach mode which provides a low rate descent. In this case, the aircraft continued to descend until it crashed. In the other crash, the pilot was in a descent for landing and the aircraft switched to TOGA. The pilot fought with the aircraft trying to get it on the runway, while the aircraft was trying to climb away from the runway. This resulted in a crash on the runway. Confusion about the TOGA mode versus the descent/approach mode was the cause in both crashes although the situations were reversed. Clearly greater visibility into the software mode was required.

The nuclear accident at Three Mile Island is another case where visibility of the system was required but the actual state of the system as invisible to the operators. In this case, a Pressure Relief Valve failed open. The open pressure relief valve allowed the liquid level in the relief system to climb. The operators were trained to believe that this indication was an indication of excess fluid in the core of the reactor, They were desperately concerned about the liquid level becoming so high that there was no gas in the top of the system to absorb pressure fluctuations. This “hard manifold” case would’ve resulted in a rapid system rupture and the escape of radioactivity from the core of the reactor. As a result of this concern, they rapidly dumped cooling water from the system causing a drop in the cooling liquid level. The liquid level got so low that the reactor dangerously overheated. Alternate temperature instrumentation that would have told them that the reactor coolant water temperature was dangerously high was defeated because the system was at the upper end of its temperature range but did not warn the operators that the indicator was at the top of its possible range (offscale). Unaware that the indicator could not read higher, the operators assumed that the reactor water was still cool. This case is explained in great detail in James Chiles book “Inviting Disaster, Lessons From the Edge of Technology” and represents a solid case where lack of visibility into a system function nearly resulted in disaster.

Three Mile Island

It is interesting to note that Chiles observes in another chapter of this book that the chain of events that led to the Apollo 13 inflight explosion was set in place by overheating of heater circuits in an oxygen tank during an off-nominal prelaunch procedure. During this procedure, the temperature of the circuit was monitored however the operators were again unaware that the temperature monitoring circuit was at the limits of range and could not display the dangerously high temperature occurring in the tank that caused the damage that ultimately led to an inflight explosion.



Apollo 13 Service Module showing effects of explosion

It is interesting to note that at the Mission Control Center in Houston and in the onboard software of the Space Shuttle it was felt that this lesson of Apollo 13 was so important that a systematic solution was implemented in order to notify flight controllers and astronauts whenever instruments were at their maximum or minimum range. In both the Mission Control software and the shuttle onboard software, whenever a parameter is displayed as numeric text (e.g. 123.4), its value is compared to the maximum or minimum value of the range of the parameter. If the software determines that the parameter is at its maximum possible reading (i.e. offscale high), the letter "H" is displayed next to the parameter (e.g. 123.4H). If the software determines that the parameter is at the lowest possible reading (i.e. offscale low) then the parameter is displayed with an L next to it (e.g. 123.4L). In this way, the flight controller or astronaut is immediately informed if the parameter is at the end of its instrument range.

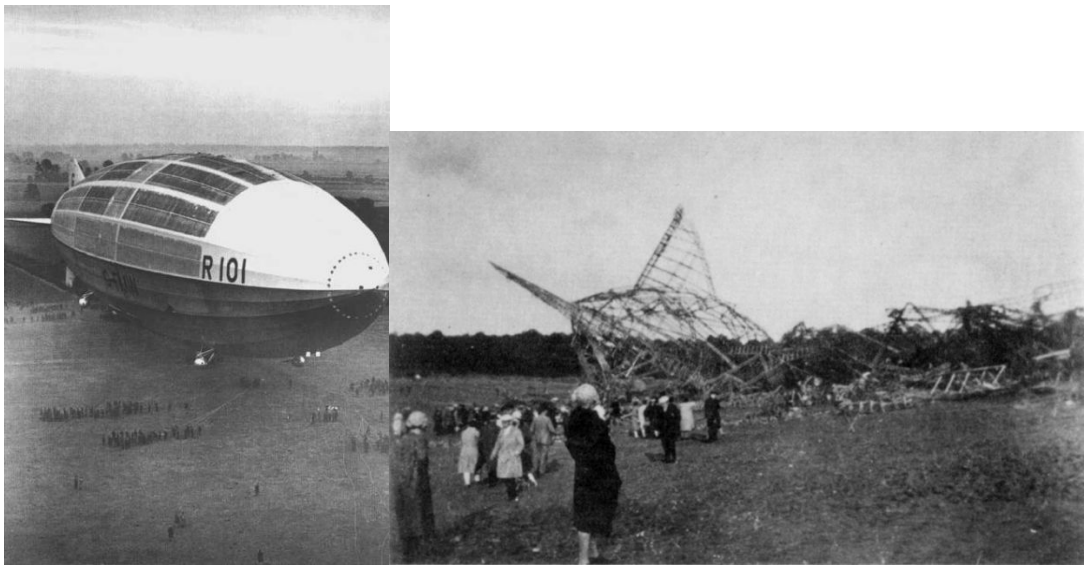
Learning how to make the system visible and building it so that its behavior is natural and instinctive for humans is a key element of style in the art of systems engineering.

Reasonableness

Technology moves ahead both in gradual evolution and rapid revolution. Evolution involves design principles and technology with good heritage. Revolution involves new design principles and technologies. When attempting both evolutionary and revolutionary progress, it is vitally important to apply reasonableness tests.

For evolutionary change, reasonableness tests can involve asking questions about the design principles involved and heritage of technology being utilized. For revolutionary change, where there is little experience from previous large scale application, it is important to ask about experience in smaller scale prototypes and the theoretical-model based analysis and predictions.

The history of technological progress is littered with ideas whose promise was so appealing that the analysis which showed that the idea was impractical was ignored. In particular model based analysis and small scale experience is critical when attempting to do something either never done before or never accomplished on as large a scale as being attempted. Chiles compares the challenges associated with building the large scale dirigible R101 with those in developing the Space Shuttle in his book Inviting Disaster and shows that in both programs there was significant evidence that the programs were not proceeding along a safe path, In both cases, the allure of routine operational travel in a new regime was so seductive that people ignored evidence of problems. In NASA we use the terminology that "the hardware is talking to us" to indicate a situation where evidence exists that a system is not behaving the way that the designers intended. In both the R101 case and in the loss of the Challenger, the hardware provided ample evidence that the systems were not going to behave as anticipated however the benefit of the technology blinded its proponents from deeper problems.



R101 Before and After

In many cases, the system never really makes it off the drawing board even though extensive efforts are made. In both the case of the Nuclear Airplane in the 1950's and the X-33 Single Stage To Orbiter Demonstrator of the 90's, the allure of the benefits of a new technological concept blinded its proponents to analytical evidence that showed that the concept was not achievable with existing materials and performance. In both cases there was insufficient experience with small scale applications of the technology to justify commitment to large scale production.

The only protection against this type of problem is to continually ask reasonableness questions and to use evidence, whether based on system performance or model based predictions to evaluate progress against goals.

Complexity

Managing complexity is one of the key aspects of the art of systems engineering. Understanding and avoiding overly complex solutions is critical. Establishing clean interfaces which minimize interaction between components is a critical technique. For example, in Apollo Saturn there were simple clean interfaces between the Apollo spacecraft and the Saturn Launch Vehicle. The Saturn launch vehicle had its own complete redundant guidance and control system which was able to deliver the spacecraft to orbital conditions without interaction from the spacecraft. The mechanical interfaces were relatively simple and in nominal flight separation between the spacecraft and the launch vehicle occurred on orbit or after trans-lunar injection. The aerodynamic and aerothermal interactions between the spacecraft and the launch vehicle were minimal. The crew escape system did not require co-operation of elements other than the spacecraft and the abort system.

This can be contrasted with the Space Shuttle design which has complex mechanical interfaces with significant aerodynamic and aerothermal interactions between the components. Separation of the Solid Rocket Boosters occurs during a complex flight regime with significant aerodynamic loads. Essentially all of the avionics are located in the orbiter and there are complex control and monitoring interfaces which cross element boundaries. Crew Escape requires significant co-operation between the orbiter and launch vehicle elements for success.

Clearly, the Space Shuttle system has been successful for over 125 flights. However this success was based on extensive investments in making this complex configuration work and verifying the adequacy of all of the design interfaces. Over 15,000 hours of wind tunnel testing were required to establish this complex configuration. Complexity does not necessarily mean that a design is unachievable, but it does indicate that a greater investment of resources will be required in order to make a more complex configuration reliable. In general, given the difficulty in designing, certifying and operating aerospace vehicles, a simpler solution is almost always preferable to a more complex solution. Complex solutions always contain a hidden cost in verification, maintenance and sustaining engineering activities.

Establishing layers in defining a system is one of our best techniques for managing complexity. Layers should only be established when they encapsulate a complete function. Layers should have complete interfaces which provide a clear set of services to layers above and rely on primitive actions provided by the layers below. Layers should also provide isolation that minimize and control fault propagation between layers.

Conclusion

Having established seven elements of style in the art of systems engineering, it is important to apply them in the design and development of systems. Designs, whether in their preliminary or detailed phase, should be evaluated to determine if they meet the elements of style. Given the process nature of most of systems engineering, it is natural to start to identify processes to evaluate a design against these seven elements. After all anything that can be measured can be evaluated and compared to a standard in a process. Many of these elements, such as margin in robustness, or even complexity can be analyzed quantitatively. In software engineering, complexity metrics are a fairly standard technique to establish risk level for a new piece of software or a software change. Some readers will undoubtedly attempt to start to build a process using a metric around these seven elements:

- Robustness
- Elegance
- Balance
- Growth Capability
- Visibility
- Reasonableness
- Complexity

But attempting to build these elements into an analytical process where each item could be individually be evaluated against a standard would be a mistake, just as if we attempted to evaluate the beauty of a painting by a numerical analysis of brushstroke density or angle. These items are not a pass/fail checklist, but rather a point of departure for discussion. One design is not better than another or rendered adequate or inadequate if it is more robust or more elegant than another. These elements of style need to be viewed in a holistic way. For example, designs that are incredibly robust may require less visibility. For example most automobile engines are very complex but are monitored with a temperature gauge and perhaps a few alarm limits implemented in warning lights. Because the engine is robust, it is possible to operate with limited visibility. Only by evaluating all of these elements in context can we determine the value of a design and whether or not additional effort is required to implement a well engineered system. Just as a piece of art can only be evaluated by looking at the total work, the artistry of a good system can only be evaluated by a total system view of all of the elements of style.