A Classic Look at Systems Engineering by Robert A. Frosch

From "Readings in Systems Engineering", edited by Francis T. Hoban and Wiliam M. Lawbaugh. NASA SP-6102.

Editors'Note

Before his term as NASA Administrator, Bob Frosch was an assistant secretary of the U.S. Navy in charge of research, development, test and evaluation of Navy programs. In that capacity, he delivered a controversial and well remembered speech to the IEEE Group on Aerospace and Electronic Systems during IEEE's international convention in New York on March 26, 1969. Edited portions of that famous speech follow in an effort to preserve what is now considered a classic formulation of systems engineering as an art rather than a science.

In this presentation, I really will be discussing the application of systems engineering to development, and in particular to military systems development (with which I am most familiar). However, from reading various journals and newspapers, I suspect my remarks are of more general applicability. I have said some of these things before, but some bear repeating and some I hope will spark new ideas.

I couple systems engineering, systems analysis and Management (with a capital "M"), because in practice they seem to be closely related terms, referring to the same constellation of systematic practices and attitudes.

o We badly lack: systems engineering of systems engineering; systems analysis of systems analysis.

o And, heaven knows, there is no: management of Management.

o Therefore, I will now preach against home, motherhood and apple pie.

To the charge that I am writing about bad systems engineering, I can only say that I am taking a pragmatic view: the thing is defined by what is done, not what is said; and if what I am describing is bad systems engineering, I can only say that I seldom see any other kind.

What I want to do is discuss briefly a series of antitheses (and perhaps an unbalanced question or two) that pit the systems world against what I believe are some aspects of the real world.

If I plot a graph versus time of what appears to be a recent rising tide of costs, cost overruns, unsatisfactory performance and unhappiness among engineers, I have reason to worry. (If this trend continues, we may have to debate whether the question "whither engineering?" is spelled with one "h" or two.) If I plot on the same graph versus time the rise in talk, directives, and use of 64 systems engineering," "systems analysis" and "Management," I see high correlation between the two graphs trouble versus time and the use of systems engineering versus time. This does not prove causation, but it suggests, at least, that the "new techniques" are proving to be a poor substitute for real science and engineering; they are, at the least, not doing what they are advertised as doing, if they are indeed actually not making things worse. It could be that things would be even worse without these new techniques, but I would like to ask some questions and suggest some reasons for believing that systems engineering, systems analysis and Management, as practiced, are likely to be part of the problem, and indeed causative agents.

I believe that the fundamental difficulty is that we have all become so entranced with technique that we think entirely in terms of procedures, systems, milestone charts, PERT diagrams, reliability systems, configuration management, maintainability groups and the other minor paper tools of the "systems engineer" and manager. We have forgotten that someone must be in control and must exercise personal management, knowledge and understanding to create a system. As a result, we have developments that follow all of the rules, but fail.

I can best describe the spirit of what I have in mind by thinking of a music student who writes a concerto by consulting a checklist of the characteristics of the concerto form, being careful to see that all of the canons of the form are observed, but having no flair for the subject, as opposed to someone who just knows roughly what a concerto is like, but has a real feeling for music. The results become obvious upon hearing them. The prescription of technique cannot be a substitute for talent and capability, but that is precisely how we have tried to use technique.

PAPER VS. PEOPLE

My first antithesis pits the systems world of paper and arrangements against the real world of people and hardware. When paper appears in the real world version of a system, it is generally only as an abstracted commentary. For example, in a very basic sense it really is of no consequence whether the documentation on a weapons system is good, bad or nonexistent; that is only a commentary on whether or why the people and the hardware actually work when called upon, and a tool to help them work. If the systems arrangements on paper and the documentation can help to make the stuff work, then they are of some use. If they are merely the formal satisfaction of a requirement, they are only an interference with engineering. Systems, even very large systems, are not developed by the tools of systems engineering, but only by the engineers using the tools. In looking back at my experiences in development, including watching a number of Navy developments over the past few years, it seems quite clear that in most cases where a system gets into trouble, a competent manager knows all about the problem and is well on the way to fixing it before any management systems ever indicate that it is about to happen. This happens if for no other reason than because the competent manager is watching what is going on in great detail and perceives it long before it flows through the paper system. That is to say, personal contact is faster than form filling and the U.S. mails. A project manager who spends much time in a Management Information Center instead of roving through the places where the work is being done is always headed for catastrophe. The MIC can assist the people who are not involved in the project toward learning of after the fact problems, but that is roughly all that it can do, and its value even for this purpose is frequently questionable.

Blaming deficiencies in management systems for problems that exist in real unknowns, or in the deficiencies of people, is mere foolishness. In a poem called "Bagpipe Music," by Louis MacNeice, the final couplet is:

"The glass is falling hour by hour, the glass will fall forever But if you break the bloody glass, you won't hold up the weather.

LINEARITY VS. THE REAL WORLD

One of the key misassumptions in modern systems engineering and systems analysis is that the total problem can be, and frequently is, decomposed into subproblems; the subproblems can be solved more or less independently, and the total solution can be synthesized by combination of the subsolutions, treating the interactions of the parts as "interfaces." The real world is, however, highly nonlinear, and unless real attention is paid to this fact, the linear decomposition treatment will fail catastrophically, because the interaction terms may be as large as the subproblems and not reducible to simple interfaces. The result may well remain decomposed.

This criticism is frequently answered by the comment that problems are unmanageable unless sliced up and, therefore, the procedure is used even though we know it may be seriously in error. This is the case of the man who played in a poker game that he knew to be crooked, because it was the only game in town; or the drunk who looked for his ring under the street lamp even though he had lost it a block away in the dark the light was better under the street light. I have some difficulty seeing that a bad analysis is really better than an informed judgment, especially since faith in the analysis (and/or the decomposed solution to the problem) is frequently, nay, usually, used as a substitute for seeking or applying any judgment at all. I am often faced with a result that seems absurd, and can even produce a quick analysis that at least makes it obvious that the solution is absurd, but am then given the answer, "Well, that's what the analysis showed."

Such a situation usually indicates room for deep criticism, either of the way in which the problem was divided up, or of peculiarities of the assumptions that drive the problem in curious and unsuspected ways, particularly through the unsuspected (by the systems person) nonlinearities of the problem. It sometimes appears that the only rational subdivision of the problem is to fractionize the blame to the point where approval is sought by default.

I would argue that careful attention to the parts of the problem that do not seem to be easily decomposable into semi independent parts might be one very good guide to areas involving high risk, since these are likely not to be amenable to our usual rules, procedures and technologies, and hence probably will have to be approached empirically.

SERIAL VS. ITERATIVE MODELS

Systems engineering techniques themselves contribute to disaster because they are all paper techniques and there are only two instead of N dimensions available. What we end up displaying are linear sequential measures of system progress.

The PERT diagram and the milestone chart are excellent examples. These both essentially assume that the progress of development and design consists of doing step A, then step B, then step C, etc. Anyone who has ever carried out a development or a design (as

opposed to setting up a management system for doing so) is well aware of the fact that the real world proceeds by a kind of feedback iterative process that looks more like a helix than like a line. That is to say, you do A, then B, then C, then you look at C and go back and change part of A again, and that causes you to fiddle with B and perhaps bring in a B prime that you bounce against C, and then go back to A and then jump to D, so that there has to be continual adjustment, going back and forth so that the system is adjusted to itself and to its end objectives as it changes and as the design or development proceeds. Because it is difficult to predict this process or to diagram it, or to predict its costs precisely without using competent engineers, the systems engineering procedures simply ignore the iterative, feedback nature of the real world because the process has been degraded to clerical reporting. To a large extent, this tends to constrain project managers from doing work in the real way toward doing it in a way that fits with their management tools. This is clearly nonsense.

As a specific example, doctrine says that one is to consider the "ilities," that is, maintainability, reliability, operability, etc., from the very beginning of the process. This is a vast waste of time and effort. I do not mean that one should not think about these things at the beginning, but it is certainly ridiculous to have a complete plan for the logistics of the maintenance of an object that has not yet been designed. I have seen overruns in expenditure and unnecessary effort generated by the fact that the linear sequencing of milestones had forced development of a complete maintenance and reliability plan for what was no longer the design, and had not been the design for three months. The machinery forced everyone to grind on and on because, after all, the maintenance and reliability milestones could not be missed without disaster and fear of cancellation of the project, even though the plan being worked out had nothing whatever to do with the hardware being designed.

In fact, the point at which to start serious work on configuration control, maintainability and reliability cannot be very well preplanned; it can be roughly preplanned, but it must be adjusted to be at the point at which the design means something and is likely to stay still long enough so that the redesign for the "ilities" will really make some sense. Judgment, not tools, is what is required.

PREDICTION VS. PRODUCTION

This brings me to a related antithesis that I describe as prediction versus production. We have come to a time when meeting certain targets seems to have become more important than producing a satisfactory system. The question is not the development of a system that performs well and was produced at a reasonable cost and in a reasonable time, but rather replacement of this sensible desire by the question, "Does the system perform as predicted, did you produce it for the cost you predicted, and on the schedule you predicted, and did you do it in the way you predicted?" Consequently, looking at what is actually happening in the development has been replaced by measuring it against a simplistic set of predicted milestones. Fulfillment of prediction has been seriously proposed as the criterion for judging system managers. It is certainly a minor criterion. Fulfillment of a need when fielded continues to be our real objective.

I know of a number of cases where the pressure on prediction has been so great that the project managers were forced to destroy the possibility of having a good system because they were not allowed to adjust what they were doing to the real world; otherwise, they would have been so far off prediction in one or another dimension that the project would have been canceled. We fell between two stools. We had a system that was only approximately what we wanted and the system failed to meet the prediction. Similarly, we have not had the sense to cancel something that met the predictions, but was no damn good.

A QUESTION OF PREDICTABILITY

It is curious that those of us, sophisticated as systems engineers, and having read history (in which no one ever seems to anticipate what really happens), knowing that the prediction time for random noise seen through a bandpass filter is only about one over the bandwidth, should yet seek predictability for the processes with a wide bandwidth of unknown information. No one can predict politics or economics; few of us predict what happens in our own lives. Why then do we assume the predictability of development of the unknown?

Should we expect development milestones to be met? Presumably, the prior probability of meeting the perfectly chosen milestone on time is distributed randomly and symmetrically about the predicted time. If the accomplishment is relatively simple, the distribution is narrow and this is called "low risk;" if the accomplishment is difficult, the distribution is wide and this is called "high risk." However, all development schedules assume success of each process. If we put trouble contingency time allowances into every task, the total contingency allowance would be unacceptably large and the development unacceptably long. This tends to bias the true risk distribution in such a way as to move the peak to the late side. Thus, there is a tendency for the "risk distribution" to peak after the milestone. The contingency allowance should be

provided in an unpopular program element, "allowance for stupidity and the unforeseen." Even so, it probably would be eliminated by the efficient review process.

All I am saying is that we only assess the risk of the predictable problems and that there is always a family of unpredictable problems that make things take longer; there are few ("oh, happy few!") cases of luck that make things take less time. We should not expect milestones to be reached, and they never (or hardly ever) are, although milestones are needed to assure adequate program pressure.

This question and my trial answer suggest a signal to noise ratio approach to risk and error assessment in development models. I have not tried to carry this further; it is left as an exercise for the developer.

SYSTEMS IN SPACE VS. SYSTEMS IN SPACETIME

My next antithesis I would label "systems in space" versus "systems in spacetime." We talk about system design and system choice in terms of ten year lifecycle costs, but the assumption we tend to make is that the system we are costing is a static object once it is designed and produced. In a way, this is forced upon us by the accountant's formalism of dividing costs into investment and recurring costs. Any system managers who say that they are designing their system in spacetime, and that they propose to design it so as to facilitate their ability to change it during the course of the ten year life cycle, will promptly have their project removed from under them because the doctrine says, "This is terribly uneconomical;" furthermore, it says that it is bad system design. I would simply like to note here that real world history tells us that all systems are changed frequently during their lifetime, if for no other reason than that the real requirements and environments and technologies for them change, often in ways that make it stupid to leave them alone. In fact, it is almost true that no military system is ever used for the precise purpose for which it is designed. Consequently, it makes sense to think about the system as something that will have a history in time and that is likely to require change, and to include some thought of this in the design. Change, strangely, is the only truly predictable attribute of the system. Perhaps I am merely going to be enshrined in the next generation of systems engineering doctrine with a special group in every project organization called 44 changeability management." I hope not. The question is not whether there will be changes or not, but whether the change process will be under conscious control. Do the developers know "what" and "why" when they allow or make a change? Pretending that no changes are allowable or desirable is merely a way of losing control of the change process.

An example of the consequences of what I mean follows. It is systems engineering doctrine that the system should be matched throughout; that is to say, it is regarded as poor practice to have, for example, high reliability components matched with low reliability components since system reliability will really be set by the low reliability components whereas system cost is likely to be set by the high reliability components.

This ignores the fact that since the system will have to change in time it may be very sensible to build in high reliability components in some parts of the system, even though the technology does not provide them for other parts of the system. During the course of the lifetime of the system, there may be a high probability of bringing the low reliability parts up to an equivalent reliability with the high reliability parts for a reasonable cost. Thus the system could be designed for great improvement in reliability from the very beginning, whereas if everything is matched to the lower reliability, the cost of improvement becomes gigantic, because the changes are extensive. In fact, the rule of thumb may not be good engineering at all if the system is designed considering change with time. We should design for growth and a process of technological leapfrogging in the system.

OPTIMIZATION VS. UNCERTAINTY

One of the fundamental tenets of systems engineering is that the system should be optimized to its purpose. This is dandy if the purpose is very specifically definable and if it is very independent of scenario and enemy behavior. If these requirements are not true, and they almost never are for any military system of any great sophistication, then optimization may merely be the definition of which catastrophe you want to undergo. My analogy is the matching of a narrowband filter to a specific signal. This is an elegant engineering procedure, provided you can depend on the signal to stay put. If the enemy, for example, has a slight adjustment in their frequency, then optimization in the normal sense rapidly becomes nonsense. There is no sense in optimizing the system beyond the accuracy of the definition of requirements, and I never, or almost never, see a definition of requirements with estimated error limits.

This particular kind of catastrophe is most often generated by the portion of systems engineering that the economists like to call systems analysis. That is to say, having chosen some scenario or problem defined in a very specific way, the system prescription follows optimization of this problem to the bitter and ridiculous end. There is a vast reluctance to look at the difficulties and the risks involved in assuming that the chosen problem is the correct problem. I will feel much better

about the use of scenarios and prediction of warfare ten years ahead for system choice and optimization if ever I meet a person who can really predict a chess game, or what will happen in the stock market tomorrow. This is not to say the game should be ruled out just because the results cannot be predicted, but rather to reinforce the fact that it is a game and cannot be taken literally.

There is a procedure called sensitivity analysis, but I have rarely seen it applied to the right parameters and variations. It is usually too difficult to do so. One rarely ever considers an error analysis, even when something is known about the error distributions of the input parameters.

A problem related to this is posed by the analysis of multipurpose objects. A tremendous difficulty is generated by the fact that the costs and characteristics must be allocated to the appearance of the system in several different scenarios. Consequently, these systems must be single solutions to several systems engineering requirements. Our usual way of dealing with this problem is to bow three times in its direction and then ignore it, because it is just too hard to solve. Solving it requires solving the systems problems for all the situations in which the multipurpose system appears, then doing all the (nonlinear) interaction cases.

In addition, the cost allocation to the various uses must be attacked. There is simply no methodology available for really trying this and hence the problem is generally ignored. This makes many of the analyses useless, but that is generally ignored too. There is no sense in pretending to solve problems by refusing to address them realistically because they are too difficult, but we go on playing that game.

OBJECTS VS. OBJECTIVES

Finally, we do not distinguish sufficiently between objects and objectives. The working tools and most of the life of systems engineering are spent trying to reach an objective, the objective finally becoming an object. It is important to keep this distinction in mind. The trouble in procurement of a development is that procurement procedures are designed to buy objects, whereas in development there is no object until the end, only an objective, and the two are not the same thing.

For example, what is a specification? A specification is an abstract set intended to describe what is to be produced, but of course it is only a portion of a total description. It is a subset of points selected from a continuous portion of an infinite multidimensional space. The object itself and its total future history is the only complete specification. Consequently, the idea of a "complete" specification is an absurdity; we can only produce a partial subset. In fact, it is possible (and we have all seen it happen) for an object that meets the subset of specification points to badly miss being a sensible solution to the problem, because it departs from the required reality between the specification subset points. I hasten to add that sometimes even the object itself, without regard to its future history, is not a sufficient specification, because it does not contain the details of the techniques used to produce it. Let the specifier beware!

Having complained about all of this throughout this article, what do I propose? The only thing I know that works is to obtain a competent person and assistants, and make sure they understand the problem not the specifications of the problem, not the particular written scenario, but what is really in the minds of those who have a requirement to be solved. Then give them funds, a good choice of managerial and systems engineering tools, and let them work at the problem after reasonably frequent conferences with those who have the requirement.

In this way, the end object may become the best that both parts of the system can produce and not merely the solution to a paper problem, said solution having the best paper properties to match the previous set of paper. (Some paper is water soluble.) It might do well to bear in mind the following closing thoughts:

o As we are now behaving, we are using up our best people in filling out documentation for their superiors to read, and most of the time no one is running the store.

o We have lost sight of the fact that engineering is an art, not a technique; a technique is a tool. From time to time I am briefed on the results of a systems analysis or systems engineering job in a way that prompts me to ask the questions: "That's fine, but is it a good system? Do you like it? Is it harmonious? Is it an elegant solution to a real problem?" For an answer I usually get a blank stare and a facial expression that suggests I have just said something really obscene.

We must bring the sense of art and excitement back into engineering. Talent, competence, and enthusiasm are qualities of people who can use tools; the lack of these characteristics usually results in people who cannot even be helped by techniques and tools. We can all do better.