LECTURE 28

SYSTEMS ENGINEERING

Parables are often more effective than is a straight statement, so let me begin with a parable. A man was examining the construction of a cathedral. He asked a stone mason what he was doing chipping the stones, and the mason replied, "I am making stones." He asked a stone carver what he was doing, "I am carving a gargoyle." And so it went, each person said in detail what they were doing. Finally he came to an old woman who was sweeping the ground. She said, "I am helping build a cathedral."

If, on the average campus, you asked a sample of professors what they were going to do the next class hour, you would hear that they were going to: "teach partial fractions", "show how to find the moments of a normal distribution", "explain Young's modulus and how to measure it", etc. I doubt that you would often hear a professor say, "I am going to educate the students and prepare them for their future careers."

You may claim that in both cases the larger aim was so well understood that there was no need to mention it, but I doubt that you really believe it. Most of the time each person is immersed in the details of one special part of the whole and does not think of how what they are doing relates to the larger picture. It is characteristic of most people that they keep a myopic view of their work and seldom, if ever, connect it with the larger aims that they will admit, when pressed hard, are the true goals of the system. This myopic view is the chief characteristic of a bureaucrat. To rise to the top you should have the larger view - at least when you get there.

Systems engineering is the attempt to keep <u>at all times</u> the larger goals in mind and to translate local actions into global results. <u>But there is no single larger picture</u>. For example, when I first had a computer under my complete control I thought that the goal was to get the maximum number of arithmetic operations done by the machine each day. It took only a little while before I grasped the idea that it was the amount of <u>important computing</u>, not the raw volume, that mattered. Later I realized that it was not the computing for the mathematics department, where I was located, but the computing for the research division that was important. Indeed, I soon realized that to get the most value out of the new machines it would be necessary to get the scientists themselves to use the machine directly so that they would come to understand the possibilities that computers offered for their work and thus produce less actual actual number crunching, but presumably more of the computing done would be valuable to Bell Telephone Laboratories. Still later I saw that I should pay attention to all the needs of the Laboratories, and not just the Research Department. Then there was AT&T, and outside AT&T

the Country, the scientific and engineering communities, and indeed the whole world to be considered. Thus I had obligations to myself, to the department, to the division, to the company, to the parent company, to the country, to the world of scientists and engineers, and to everyone. There was no sharp boundary that I could draw and simply ignore everything outside.

The obligations in each case were of: (1) immediate importance, (2) longer range importance, and (3) very long term importance. I also realized under (2) and (3) that one of my functions in the research department was not so much to solve the existing problems as to develop the methods for solving problems, to expand the range of what could be done, and to educate others in what I had found so that they could continue, extend, and improve my earlier efforts.

In systems engineering it is easy to say the right words, and many people have learned to say them when asked about systems engineering, but as in many sports such as tennis, golf, and swimming it is hard to do the necessary things as a whole. Hence systems engineers are to be judged not by what they say but by what they produce. There are many people who can talk a good game but are not able to play one.

The first rule of systems engineering is:

If you optimize the components you will probably ruin the system performance.

This is a very difficult point to get across. It seems so reasonable that if you make an isolated component better then the whole system will be better - but that is not true, rather the system performance will probably degrade! As a simple example, I was running a differential analyzer and was so successful in solving important problems that there was need for both a bigger one and second one. Therefore we ordered a second one that was to be connected with the first so that the two could be either operated separately or together. They built a second model and wanted to make improvements, which I agreed to only if it would not interfere with the operation of the whole machine. Came the day of acceptance on the shop floor before dismantling and moving it to our location. I started to test it with the aid of a reluctant friend who claimed that I was wasting time. The first test and it failed miserably! The test was the classic one of solve the differential equation

$$y'' + y = 0, \quad y(0) = 1, \; y'(0) = 0$$

whose solution is, of course, $y = \cos t$. You then plot $y(t)$ against $y'(t)$ and you should get a circle. How well it closes on itself, loop after loop, is a measure of the accuracy.

So we tried the test with other components, and the same result. My friend had to admit that there was something seriously wrong, so we called in the people who constructed it and pointed out the flaw - which was so simple to exhibit that

they had to admit there was something wrong. They tinkered and tinkered while we watched, and finally my friend and I went to lunch together. When we came back they had located the trouble. They had indeed improved the amplifiers a great deal, but now currents through the inadequate grounding was causing back circuit leakage! They had merely to put in a much heavier copper grounding and all was well. As I said, the improvement of a component in such a machine, even where each component is apparently self-standing, still ruined the system performance! It is a trivial example, but it illustrates the point of the rule. Usually the effect of the component improvement is less dramatic and clear cut, but equally detrimental to the performance of the whole system.

You probably still do not believe the statement so let me apply this rule to you. Most of you try to pass your individual courses by cramming at the end of the term, which is to a great extent counter-productive, as you well know, to the total education you need. You look at your problem as passing the courses one at a time, or a term at a time, but you know in your hearts that what matters is what you emerge with at the end, and what happens at each stage is not as important. During my last two undergraduate college years when I was the University of Chicago, the rule was that at the end you had to pass a single exam based on 9 courses in your major field, and another exam based on 6 in your minor field, and that was mainly what mattered, not what grades you got along the way. I, for the first time, came to understand what the system approach to education means. While taking any one course, it was not a matter of passing it, pleasing the professor, or anything like that, it was learning it so that at a later date, maybe two years later, I would still know the things that should be in the course.

Cramming is clearly a waste of time. You really know that it is, but the behavior of most of you is a flat denial of this truth. So, as I said above, words mean little in judging a systems engineering job, it is what is produced that matters. The professors believe, as do those who are paying the bill for your education, and probably some of you also, that what is being taught will probably be very useful in your later careers, but you continue to optimize the components of the system to the detriment of the whole! Systems engineering is a hard trade to follow; it is so easy to get lost in the details! Easy to say; hard to do. This example should show you the reality of my remark that many people know the words but few can actually put them into practice when the time comes for action in the real world. Most of you can't!

As another example of the effects of optimizing the components of a system, consider the teaching of the lower level mathematics courses in college. Over the years we have optimized both the calculus course and linear algebra, and we have stripped out anything that was not immediately relevant to each course. As a result the teaching of mathematics, viewed as a whole, has large gaps. We barely mention: (1) the important method of mathematical induction, (2) after a brief mention in algebra in con-

nection with quadratic equations we ignore, almost in holy dread, any mention of complex numbers until the fatal day, late in the linear algebra course, when complex eigenvalues and eigenfunctions arise and the poor student is faced with two new, difficult concepts at once and is naturally baffled, (3) the important, useful method of undetermined coefficients is briefly mentioned, (4) impossibility proofs are almost totally ignored, (5) discrete mathematics is ignored, (6) little or no effort goes into trying to convert what to many of the students are just "chicken tracks on paper" into meaningful concepts that are applicable to the real world; and so it goes, large parts of any reasonable mathematical education are omitted in the urge to optimize the individual courses. Usually the inner structure of the calculus and the central role of the limit is glossed over as not essential.

All the proposed reformations of the standard calculus course that I have examined, and there are many, never begin by asking, "What is the total mathematical education and what therefore should be in the calculus course?" They merely try to include computers, or some such idea, without examining the system of total mathematical education that the course should be a part of. The systems approach to education is not flourishing, rather the enthusiasts of various aspects try to mold things to fit their local enthusiasms. The question, as in so many situations, "What is the total problem in which this part is to fit?" is simply regarded as too big, and hence the sub-optimization of the courses goes on. Few people who set out to reform any system try first to find out the total system problem, but rather attack the first symptom they see. And, of course, what emerges is what ever it is, and is not what is needed.

I recently tried to think about the history of systems engineering - and just because a system is built it does not follow that the builder had the system rather than the components in mind. The earliest system I recall reading about in its details is the Venetian arsenal in its heyday around 1200-1400. They had a production line and as a new ship came down the line, the ropes, masts, sails, and finally the trained crew, were right there when needed and the ship sailed away! At regular intervals another ship came out of the arsenal. It was an early "just in time" production line that included the people properly trained as well the equipment built.

The early railroads were surely systems, but it is not clear to me that the first builders did not try to get each part optimized and really did not think, until after the whole was going, that there was a system to consider - how the parts would intermesh to attain a decent operating system.

I suspect that it was the telephone company that first had to really face the problems of systems engineering. If decent service was to be supplied then all the parts had to interconnect, and work at a very high reliability per part. From the first the company provided a service, not just equipment. That is a big difference. If you merely construct something and leave

it to others to keep it running that is one thing; if you are also going to operate it as a service then it is another thing entirely! Others had clearly faced small systems as a whole, but the telephone system was larger and more complex than anything up to that point. They also found, perhaps for the first time, that in expanding there is not an economy of scale but a dis-economy; each new customer must be connected with <u>all</u> the previous customers, and each new one is therefore a larger expense, <u>hence</u> the system must be very shrewdly designed.

I do not pretend to understand how I, with a classical pure mathematics education, was converted to being a systems engineer, but I was. I suppose it started quietly with my college education, but it really got started at Los Alamos where it was obvious to all of us that we were constructing a design for which every component had to be properly coordinated if the whole was to do what it had to do - including fit into the bomb bay of the current airplane. And to do the job rapidly before the enemy, who was known to be working on it too, reached success.

The Nike guided missile systems, the computer systems that I ran, and many other aspects of the work at Bell Telephone Laboratories all taught me the facts of systems engineering - not abstractly, but in hard lessons daily illustrated by idiots who did not understand the whole as a whole, but only the components. I have already observed that I did not immediately grasp the systems approach as I was running the computers, but at least I gradually realized that the computers were but a part of a research-development organization, vital to be sure, but it was their value to the system that mattered in the long run, how well the computers helped reach the organization's goals, as well as society's goals, and not how comfortable it was for the staff operating the computers.

That brings up another point, that is now well recognized in software for computers but it applies to hardware too. Things change so fast that part of the system design problem is the fact that the system will be constantly upgraded in ways that you do not now know in any detail! Flexibility must be part of modern design of things and processes. Flexibility built into the design means not only that you will be better able to handle the changes that will come after installation, but it also contributes to your own work as the small changes that inevitably arise both in the later stages of design and in the field installation of the system. I had not realized how numerous these field changes were until the early Nike field test at Kwajalain Island. We were installing it and still there was a constant stream of field changes going out to them!

Thus rule 2:

Part of systems engineering design
is to prepare for changes
so that they can be gracefully made
and still not degrade the other parts.

Returning to your education, our real problem is not to prepare you for our past, or even the present, but to prepare you for your future. It is for this reason that I have stressed the importance of what currently is believed to be the fundamentals of various fields, and have deliberately neglected the current details which will probably have a short lifetime. I cited earlier the half-life time of engineering details as being 15 years - half of of the details that you learn now will probably be useless to you in 15 years.

Rule 3:

The closer you meet specifications
the worse the performance will be when overloaded.

The truth of this is obvious when building a bridge to carry a certain load; the slicker the design to meet the prescribed load the sooner the collapse of the bridge when the load is exceeded. One sees this also in a telephone central office; when you design the system to carry the maximum load then with a slight overload of traffic performance degrades immediately. Hence good design generally includes the graceful decay of performance when the specifications are exceeded.

In preparation for writing this Lecture I reread once more an unpublished set of essays on: <u>One Man's Systems Engineering</u>, by H. R. Westerman (1975), then of Bell Telephone Laboratories. They are the only deeply philosophical discussion I know of the "what, how, and why" of systems engineering. While I will make small differences at various points from what he says I am in fundamental agreement with him. I can only summarize, all too briefly, what he says in 10 essays whose titles are:

1. One Man's Systems Engineering
2. What is Systems Engineering?
3. On the Objective
4. What Does a Systems Engineer Do?
5. The Framework of Systems Engineering
6. Organization and Systems Engineering
7. Objectives and Policy Makers
8. On the Methodology of Systems Engineering
9. Evaluation and (Un)common Sense
10. Envoy

The list shows clearly his breadth of vision, which arose from many years on both military projects and telephone systems problems.

He believes more in the group that attacks systems engineering problems than in the individual problems attacked, whereas I, from my limited experience in computing where I had no one near by to talk to about the proper use of computers, had to do it single handed. Of course his problems were far more difficult than mine.

He believes that specialists brought together to make a team

are the basis of systems engineering, and that between jobs they must go back to their specialties to maintain their expertise. Using the group too often to fight fires is detrimental in the long run since then the individuals do not keep their skills honed up in their areas.

We both agree that a systems engineering job is never done. One reason is that the presence of the solution changes the environment and produces new problems to be met. For example, while running the computing center in the early days I came to the belief that small problems were relatively more important than large ones, that regular, dependable service was a desirable thing. So I instituted a 1 hour period in each morning and each afternoon during which only 3 minute (or less) problems were to be run, (mainly program testing), and if you ran over 5 minutes you got off the machines no matter how much you had claimed you were practically finished. Well, people with 10 minute problems broke them up into three small pieces with different people for each piece and ran them under the rules - thus increasing the load in the input/output facilities. My solution's very presence alters the system's response. The optimal strategy for the individual was clearly opposed to the optimal strategy for the whole of the laboratories, and it is one of the functions of the systems engineer to block most of the local optimization of the individuals of the system and reach for the global optimization for the system.

A second reason that the systems engineers design is never completed is that the solution offered to the original problem usually produces both deeper insight and dissatisfactions in the engineers themselves. Furthermore, while the design phase continually goes from proposed solution to evaluation and back again and again, there comes a time when this process of redefinement must stop and the real problem coped with - thus giving what they realize is, in the long run, a suboptimal solution.

Westerman believes, as I do, that while the client has some knowledge of his symptoms, he may not understand the real causes of them, and it is foolish to try to cure the symptoms only. Thus while the systems engineers must listen to the client they should also try to extract from the client a deeper understanding of the phenomena. Therefore, part of the job of a systems engineer is to define, in a deeper sense, what the problem is and to pass from the symptoms to the causes.

Just as there is no definite system within which the solution is to be found, and the boundaries of the problem are elastic and tend to expand with each round of solution, so too there is often no final solution, yet each cycle of input and solution is worth the effort. A solution that does not prepare for the next round with some increased insight is hardly a solution at all.

I suppose that the heart of systems engineering is the acceptance that here is neither a definite fixed problem nor a final solution, rather that evolution is the natural state of af-

fairs. This is, of course, not what you learn in school where you are given definite problems which have definite solutions.

How, then, can the schools adapt to this situation and teach systems engineering, which because of the elaboration of our society, becomes ever more important? The idea of a laboratory approach to systems engineering is attractive until you examine the consequences. The systems engineering described above depends heavily on the standard school teaching of definite techniques for solving definite problems. The new element is the formulation of a definite problem from the background of indefiniteness that is the basis of our society. We cannot elide the traditional training, and the schools have not the time, nor the resources, except in unusual cases, to take on the new topic, systems engineering. I suppose the best that can be done is regular references to how the class room solutions we teach differ from the reality of systems engineering.

Westerman believes, apparently, that the art of systems engineering must be learned in a team composed of some old hands and some new ones. He recognizes that the old hands have to be gradually removed and new people brought into the team. I have no answer for how to teach my "lone wolf" experiences except what I have done so far, by stories of what happened to me in given situations. Usually the actual circumstances are so complex that it takes a long, long time to get across the outside policies, organization habits, characteristics of personnel that will run the final system, operating conditions in the field, tradition, etc. that surround, and to a great extent circumscribe, the solution that is to be offered to the systems problem. The solution is usually a great compromise between conflicting goals, and the student seldom appreciates the importance of the intangible parts of the boundary which shape the form of the answer. Thus real systems engineering problems are almost impossible to exhibit in proper realistic detail; instead toy situations and stories must be used that, while eliminating much detail, do not distort things too much.

If you will look back on these Lectures you will find a great deal of just this - the stories were often about systems engineering situations that were greatly simplified. I suppose that I am a dedicated systems engineer and that it is inevitable that I will always lean in that direction. But let me say again, systems engineering must be built on a solid ground of classical simplification to definite problems with definite solutions. I doubt that it can be taught ab initio.

Let me close with the observation that I have seen many, many solutions offered that solved the wrong problem correctly. In a sense systems engineering is trying to solve the right problem, perhaps a little wrongly, but with the realization that the solution is only temporary and that later on during the next round of design these accepted faults can be caught provided insight has been obtained. I said it before, but let me say it again, a solution that does not provide greater insight than you had when you began is a poor solution indeed, but it may be all

that you can do given the time constraints of the situation. The deeper, long term understanding of the nature of the problem must be the goal of the system engineer, whereas the client always wants prompt relief from the symptoms of his current problem. Again, a conflict leading to a meta systems engineering approach!

As an example of the deepening of our understanding of a system and its problems, consider the Nike guided missile project. At first it was to build a missile that would shoot down a single target. This accomplished, we began to think of a battery of Nike missiles and how to coordinate the individual missiles when under attack by a fleet of enemy airplanes. Then came the day when we began to think about what targets to defend, which cities to defend and which not to. We began to realize that the answer is that all targets should be equally expensive to the enemy - there should be no under-defended or over-defended target, each should be defended in proportion to the damage that could be done by the enemy. Thus we began to see that the Nike missile is merely a device to make the enemy pay a price for the damage he can inflict, with no "cheap" targets available. How different this view is from the one with which we began! It illustrates the point that each solution should bring further understanding of the problem; the the first symptoms they tell you will not last long once you begin to succeed; the goal will be constantly changing as your and the customer's understanding deepen.

Systems engineering is indeed a fascinating profession, but one that it hard to practice. There is a great need for real systems engineers, as well as perhaps a greater need to get rid of those who merely talk a good story but cannot play the game effectively.