THE UNWRITTEN LAWS OF SYSTEMS ENGINEERING

David F. McClinton
David F. McClinton & Associates
14181 Barlupi Circle., Sonora, CA 95370

Presented at the Fourth Annual International Symposium
of the National Council on Systems Engineering (NCOSE), 1994

Abstract. The paper states some of the unwritten but spoken laws that abound in any engineering project. The laws are really simple basic truths that arise in typical engineering discussions. We didn't hear them in engineering school but they came through loud and clear in that school of hard knocks, the engineering project. The initial three laws are (1) Everything interacts with everything else; (2) Everything goes somewhere; and (3) There is no such thing as a free lunch. Several other laws capture the lessons learned in various activities in the systems engineering process.

INTRODUCTION

Physical laws like Maxwell's Equations are brief and elegant. Newton's three laws are so simple and yet so profound. Beautiful. Technical disciplines, on the other hand, collect brief statements that capture the essence of that discipline. Sometimes in a cynical nature but many times right on the mark. Thermodynamics can be captured by three "laws." These are (1) You can't win; (2) The best you can do is tie; and (3) You can't get out of the game.

THE THREE LAWS OF SYSTEMS ENGINEERING

Many such laws relating to Systems Engineering come to mind for the various activities of the Systems Engineering process. These simple statements hold the meaning of an activity in a way that is easy to remember and forms a guiding principle. The three unwritten laws of systems engineering can be simply stated as:

- Everything interacts with everything else.
- Everything goes somewhere.
- There is no such thing as a free lunch.

Decomposition. The first law seems to be at odds with the principle of decomposition where we decompose a large complicated problem into a number of smaller complicated problems and so on until we have reduced the problem to a manageable size where a few people can get around the problem. The idea is to break things apart at simple and clean interfaces so that after solving the piece parts we can assemble things in a simple and easy manner into a final solution. Decomposing systems in the simplest way is an art but we should not forget that impacts ripple throughout the system and can never be ignored.
Interfaces. The second law deals with the multiple interfaces we have exposed in the decomposition. The interfaces have to be consistent and account for all things generated like control signals, communications, as well as thermal loads, EMI, power spikes, etc. We must account for everything at the interface and follow where it goes. If it leaves some place, then it must arrive someplace else.
Trade Studies. The third law deals with trade studies and the whole act of system synthesis. Never become so enamored with a design decision that you forget the down side of that decision. This third law is the law of decision analysis.

THE SIMPLE TRUTHS

The following laws treat the various lower level considerations that support the three laws above. Maybe they are not laws but only simple truths. These simple truths are what guide up along the systems engineering process.

Configuration Management. The requirements derivation process is the area where we seem to fail the most. Once the requirements are settled and under Configuration Management there will be many attempts to change the requirements. Requirements drift is caused by both the customer and contractor. Customers love to add features in an attempt to sell their program. Contractor design engineers love to improve the performance of the system. Either way, if left unchecked, they both can kill the program due to cost, complexity, and delay.
The fourth Law:

- Never confuse change with progress.

Functional Analysis. Functional analysis is conducted in order to understand the mission need and the things the system must perform. Different people performing a functional analysis will produce different functional decompositions and they all can be right. Most functional decompositions become convoluted and messy. As you work at lower levels, you start to understand the functions at the higher levels and wish that you had done it some other way.
The Fifth Law:

- Never be afraid to start over.

Failure-to-Commit. Once you have requirements defined and their ties to the functional decomposition, failure-to-commit sets in. Never try to be completely right (analysis paralysis) or you will never converge on anything. Failure-to-commit stops progress in its tracks. Don't try to make everything perfect.
The Sixth Law:

- Better is the enemy of good.

Engineering Documentation. Development of engineering documentation is essential in the system engineering process. Trade studies must be documented and referenced as the development process proceeds. Remember what Sam Goldwyn said: "a verbal agreement is not worth the paper it's written on." Force the documentation process. Schedule and assign engineering memos on analyses and trades that you know will be needed.
The Seventh Law:

- If it is not written down, it never happened.

Planning Documentation. The development of the system concept into a baseline and the development of plans and related documentation is a massive workload. Similar documentation from earlier programs can be the basis or form a template for the new documentation.
The Eighth Law:

- Never be above plagiarism.

Allocation of Resources. It is easy to overdo the documentation process or any other part of the system engineering process. Since we do not have unlimited resources, it is vital that resources are allocated to the critical tasks and not spent on tasks that become the playground of the analyst and designers.
The Ninth Law:

- A thing not worth doing is not worth doing well.

System Synthesis. We are finding more and more projects where cost is king. We are asked the build everything out of existing components. Of course increased performance usually means old technologies might not satisfy and finding something off-the-shelf just might not happen.
The Tenth Law:

- There is no shelf.

System Integration. The main premise of system engineering is that a large complex problem can be decomposed into a set of smaller, simpler problems exposing interfaces that tie the smaller blocks together. Interface configuration management is required to maintain agreed-to interfaces. This management activity is required throughout the lifetime of the project.
The Eleventh Law:

- Any interface left to itself will sour.

Work Planning. There is work, non-work and un-work. Work is applied against the planned activities. Non-work is work performed by the non producers and is inevitable in any large project. Non-work does not hurt any other task except for those that need the results of that effort. We can always pick up the ball and do that needed work. The killer that must be purged is the un-work or work that must be scrapped and redone. Any fool can design anything given

unlimited budget, time and talented workforce. The challenge of project management is to accomplish much with little.
The Twelfth Law:

- Plan your work and work your plan.

Contingency Planning. In our planning, we tend to be optimistic and set work spans as if we were going to do the task with our set way of doing things. When the job is assigned to someone else, it always takes longer than we planned. The assigned task is perceived as being more complicated than we imagine. Almost always, rework is required. Our schedules should recognize contingencies.
The Thirteenth Law:

- We don't have time to do it right but we have time to do it twice.

Assignment of tasks. Tasks will be complicated because of the difficulty in making understood the scope of the assignment. When we want something simple and accomplished in a few hours, it is often seen as a very difficult task.
The Fourteenth Law:

- Nothing is impossible to the man who doesn't have to do it.

The Problem of Make Work. Make work is wasted effort that is best directed elsewhere. Always make sure your tasks are directed toward an end goal that is a common vision of the workforce. It is easy to keep refining the design when the basic design itself is faulty. A task without value added to the project can not be tolerated.
The Fifteenth Law:

- Don't keep polishing the cannon ball but do get the caliber right.

Design Validation. Validation of the design against requirements may be based on analysis or test. The credibility of that result is always a problem. A test is always better if you can afford the expense. The Sixteenth Law:

- Any analysis will be believed by no one but the analyst who conducted it.

Any test will be believed by every one but the person who conducted it.

Performance Verification. We set the scope and range of tests to gain confidence in the ability of a system to meet requirements set by the mission need. We always have arguments that we are over testing and wasting time and money.
The Seventeenth Law:

- One test is worth a thousand expert opinions.

Test Planning. The subject of test philosophy reveals another law. Never perform a test just because it seems like a good idea. It has to be needed to validate a requirement.
The Eighteenth Law:

- Never conduct a test if you can't live with all possible results.

Time Management. Attending meetings seem to be a major undertaking. Push your position at the time. Later study and analysis may prove you right but events have moved on and passed you by. Poor time management can dilute the best laid plans.
The Nineteenth Law:

- After all is said and done, a lot is said and very little is done.

Briefing Chart Complexity. Briefing charts seem to be one of our most cherished products. Follow the KISS principle (Keep it simple stupid). Don't dump a complex chart on your audience when a couple of simple charts will do.
The Twentieth Law:

- Never have more than ten blocks in your block diagram.

Briefing Technique. Word charts are boring and most of your audience can read anyway. You can say anything you want with a picture chart.
The Twenty First Law:

- Never use a word chart when a picture chart will do.

Systems Management Risk. The systems engineering management of a large project is very difficult in the start up phase. Requirements are late. Of course they are late but engineering should never use that as an excuse and fail to conduct parametric analysis around expected requirement limits. Staffing is under the planned level. Design engineering is yelling for firm requirements as if the system engineer knows what the requirements are but just won't tell. The result is that the Chief Systems Engineer is under fire and may be replaced by a new person who is given a new longer schedule and increased budget and perhaps reduced expectation of results. This typical half life of one year makes some people cautious.
The Twenty Second Law:

- Never go in with the first wave.

More Risk. It is not clear that you can succeed if you are part of the second wave. Some projects are so difficult that an extension of the law is as follows:
The Twenty Third Law:

- Never go in with the second wave either.

Management Style. You need a management style with a firm manner and that gives the appearance that you know which way to lead. Hang tough.
The Twenty Fourth Law:

- Have the heart of a child but keep it in a jar on your desk.

Legal Conflict. Finally, you should remember that as a system engineer you can be liable for damages if your system should fail. Don't spend time on lawyer bashing. Just remember the guidelines of any legal conflict.
The Twenty Fifth Law:

- Deny everything, admit nothing, demand proof, and reject the proof.

CONCLUSION

The application of these twenty-five laws surfaces almost every day in the normal process of systems engineering. They are triggered in the day-to-day events and are a memory response to their basic truths. They may make you stop and think.
Have we cut the oral tradition of passing these truths down to our junior members? I don't think so. I hear a different one stated everytime a new project forms and people start the systems engineering process once again.

ABOUT THE AUTHOR

David F. McClinton has over thirty years direct systems engineering and project management experience. He has provided consulting services to Telecom Australia, Marconi U.K. and the Australian Commonwealth Department of Defence on OTH Radar and other Commonwealth Programs. He has consulted in Saudi Arabia on Air Traffic Control. He has conducted System Engineering Process seminars to Telecom Australia; Melbourne Royal Institute of Technology; New South Wales State Rail; University of Adelaide; the Australian Department of Defence; Caltech; and other Universities.
Mr. McClinton was for many years a Chief Systems Engineer at Lockheed Missiles & Space Co. He holds a B.S. degree from UCLA.

- Link to D. F. McClinton
- Go to EE - Home Page