

Systems Management; What Was It All About?

Paul Pencikowski
pencipa@yahoo.com

The School of Systems Management at the University of Southern California has provided the aerospace industry with (literally) thousands of technology-developers of the highest caliber; that the school has been “phased out” is an unfortunate sign of the times.

Throughout the years, there have evolved a series of Basic Tenets.... Specifically, the following “13 areas of interest” compiled by the author and designed to assist technology developers and high-technology “answer seekers” (and perhaps, in this day and age, for the aspiring “technology detective”).

Each of the 13 areas will be addressed in detail in serial order. No one tenet is really any more important than any other. They have been arranged in order of *approximate utility* by the author, based on his real-world experience as an Advanced Projects Manager for the Northrop Grumman Corporation.

Especially note that these tenets are oriented toward man-in-the-loop systems (either the end-product or the management system used to design/build the product).

At the end of each tenet is an abbreviated recommendation to the aspiring “systems person” as to how they might employ these tenets in real-world situations. The acronym RWR (“Real World Recommendations”) will preface the recommendation.

The 13 areas of interest are:

1) Invalid Assumptions (the #1 reason for system failure).

The days of Basic Research are over. Basic Research allows the investigation of phenomena-of-interest with *no* assumptions. As such, basic research also has little value; the work is too simplistic¹. But we have swung the pendulum-of-thought too far in the other direction; it is common for “analysts” to draw complex conclusions from shreds of data..... These “conclusions” then become the basis for an assumption-of-fact in another (perhaps unrelated) problem. Example: Software simulation is shown to correlate perfectly as a replacement for “test” in the design of a model (miniature) rocket or an earlier,

¹ An example of basic research would be to “mix chemicals A and B together and see what happens”; or perhaps, “Give the monkey apples instead of bananas and document its’ response”.

much simpler design. Algorithms get published. Based on this, software simulation is “assumed valid” and chosen to replace Test in the development of a complex rocket design. The software simulation is “validated” via application to a very small number of carefully-chosen subsystem simulations, which “succeed”. The Test program is downsized or eliminated, and the rocket development goes forward to an untimely end. (Example: Ariane V).

Workaround: Make as few assumptions as possible (especially when dealing with man-in-the-loop exercises); Support assumptions made by Senior People and based on first-hand experience; Be sure all assumptions pass the “common sense test”. Be wary of any assumptions that seem to invalidate tried-and-true practices.

RWR: Always provide a *list of assumptions* and a *traceability diagram* for your list of assumptions. This diagram need not be an absolute derivation/proof of your assumptions, just a “visual credibility enhancer”. As a corollary, always ask your data-providers for a list of assumptions and a traceability diagram. This prevents someone else’s invalid assumption from interfering with *your* design.

2) Overreliance on Tools (the #2 reason for system failure). Productivity-enhancement “tools” are all the rage. Unfortunately, after Microsoft Office (however simple it is...), there aren’t many validated ones. Remember Ada? (Originally sold as a “self-documenting programming language”. Where is it now?). Tools must be exercised with the lowest-quartile user, because almost any tool can be shown to work when your best technician is exercising it. Tools are best used for repetitive tasks. A handful of dubious tools can tie-up your best workers and, worse, these workers are not likely to appreciate being “replaced” by machines (which is an entirely new problem....). Tools are also known by the slang term “Silver Bullet” (i.e. “multi-dimensional problem-solvers”). Some examples of purported “do-all” tools: Object-Oriented Programming, and NeXT computers. On the human-operations side, “Integrated Product Teams” made no impact on aerospace industry productivity, especially as IPT’s are basically Matrix Management redux.

Regarding tools, their on-site exercise using *your* real-world problems, using *your* perfectly-average workers, using *your* pre-determined performance metric, is a mandatory step before purchase/implementation. Be

doubly suspicious when your customer mandates the use of specific tools.

RWR: Insist that the tool-purveyor teach your *bottom-rung* employees in the use of the tool-of-interest. If it works, you have made a productive employee out of a marginal producer.

3) Nothing is Random There is *no such thing as a Random Error* in aerospace or anywhere else. Anything that happened once can happen again. Far too many “random bugs” have surfaced in lab-testing, been dismissed as “random”, then re-surfaced to destroy a full-scale article. This applies to careers also; an action that caused career-pain, if repeated, will likely cause career-pain again. People as individuals are remarkably “unique”; however, people as “groups” (as in “management”) are amazingly homogenous.

RWR: Always ask for a list of “prior failure items” in any data/subsystem that has the potential to influence your work. Be *absolutely certain* these deficiencies have been corrected. Corollary: Be absolutely certain that enough test-trials have conducted to satisfy reliability questions. Too many “Tests” have been early-terminated because “things were going so well” (correctly called a “false positive” result), with later—disastrous—results.

4) Deviations from Gaussian distributions Alarm bells! As far as possible, plot all “x-vs.-y” variables. The “bell-shaped curve” is what you’re looking for. If you find a distribution (beta distribution etc.) that does not fit the “common-sense test”, put the offending system under your microscope. Gauss was a genius!

RWR: Plot the management-makeup characteristics against the characteristics of the general workforce. If the distribution is not Gaussian, consider “relocating yourself” to a more favorable environment.

5) Linear vs. non-linear systems Beware of any graph that plots x-vs-y with the results as a straight line. Yes, these conditions do exist. But in the aerospace scheme of things, 99% of all action-reaction plots are non-linear. If a plotted straight-line is an *extrapolation*², avoid the associated project like the plague.

RWR: Analyze every projection made by every analyst/department that you can find. Search for “linear projections”. If more than 10% of these projections are linear, than the credibility of these analysts/departments is suspect. Given “excessive” linearities, you must adjust your decisions (relative to dealing with these individuals/departments) accordingly.

² “If 2 salesmen can sell 2 widgets in a week, 2000 salesmen can sell 2000 widgets in a week”. Not likely!

6) Probabilism vs. Determinism Other than death, taxes, and the speed of light³, not much is deterministic. Beware of anyone/anything predicting “certainty”. Or, in combination with #1 above (Invalid Assumptions), predicting “virtual certainty” (a combination of “pretty sure” with “it worked in the lab”).

RWR: Tag every variable in your work with a “probability function”. Use “unfavorable values” as levers to pry-loose more budget. *Make your competition do the same. Refuse-to-lose to a “counterproposal of promises”*.

7) Survey Data vs. Observed Data Survey data is suspect, *always*. The corollary is “don’t listen to what people say, *watch what they do*”. It is virtually impossible to build a survey without either preconceived notions (i.e. a “desired outcome”⁴) or the likelihood of the survey being “gamed” by the respondents. The worst offender in aerospace? Easy..... *marketing departments*.⁵

RWR: Build “man-in-the-loop” (MITL) tests, however simplistic they may seem, as substitutes for survey data. This makes you the expert, not the survey-respondent. Survey data is really simple to defeat, MITL studies live forever.⁶

8) Response in stressful situations People, in stressful situations, will always regress to habit. “When stressed, people will regress to what they know how to do, rather than what needs to be done”. This is why rote-memory and repetitive-training is a must for the combat soldier.⁷ Research (supported by the authors first-hand experience) says that the average combatant, in battle, performs at the level of an 8-to-12 year old child operating under “normal” conditions.

RWR: Many a career has been scuttled because a technology-developer got caught in a “verbal crossfire”

³ The author has been criticized by several physicists for naming the Speed Of Light a deterministic function; maybe so, but it was depressing to list “death and taxes” as the *only* “absolutes”.

⁴ The classic violator is the J. D. Powers surveys. If a client cannot succeed with conventional survey-topics, J. D. Powers will create a new one for you (the authors favorite being “Rated #1 in *perceived* initial quality”).

⁵ The “surveys” most-often quoted by marketing departments are “customer responses to verbal questions”. If there ever was a formula-for-disaster, this is it.

⁶ The author, somewhat sheepishly, remains a “recognized expert in Synthetic Aperture Radar use” based on a single MITL study done in 1979!

⁷ This is also why robots are such formidable adversaries (re: The Terminator).

during a critical presentation. Get into a Toastmasters⁸ club! The benefits will *prevent* your regression to blithering-idiot when overcome-by-events. Regarding technology-development, KISS⁹ and MITL-test with “non-experts”. Remember, the most likely “volunteer testee” for your new widget is a V.P. who “has a background with these things”.

9) Solution-paths for complex problems Complex problems require complex solutions, period! Absolutely beware of anyone proposing a “simple fix” to a problem that is anything other than simple.¹⁰ A classic example is the addressing of a “fix” in terms of software as “it’s *only* a software fix”. If there was ever a “red flag statement”, this is it.

RWR: In proposing any “fix” to a complex problem, *quote this law first!* Then graphically show that no “simple fix” has the bandwidth to deal with the problem. Prioritize the value of the elements of your plan, and show a gradual phase-in connected to probabilities of success; after all, you *might* get lucky and solve the problem earlier than expected.

10) Surface Validity An offshoot of #9 is the hypothesis that has only “surface validity”; that is, if one were to probe the basis of the research, the entire hypothesis would fall apart quickly.¹¹ The industry pressure is to reduce costs, which means man-hours, which (in the worst case) means reducing the investigative stages of a problem to only that required by a vivid imagination and the bare minimum of facts. Sadly, this is becoming the norm.

RWR: Don’t bash anyone’s project in real-time; Propose your “fix” as a required add-on to the offending system. In the case where you do want to defeat a proposal, break it down into as many discreet elements as possible, and if possible, attack only those that fail the “common sense test” or that “clearly violate the laws of physics”. Highlight any of the “technology miracles” that must happen for the offending proposal to succeed; point out the time/cost losses that are likely to occur, and graphically show the projected point where the “project is

⁸ Toastmasters is a nationally-recognized public-speaking-improvement social organization.

⁹ “Keep It Simple, Stupid”.

¹⁰ Interestingly, this situation most often pops up in the form of a management directive, as in: “Just put in a fix” when a re-design is in order.

¹¹ The classic example of the hypothesis/surface-validity argument is law enforcement. While it is fashionable to blame “slick lawyers” as the reason for the relatively low conviction rates, the real reason (per UConn School of Criminal Justice) is that the cases brought against the accused have only surface-validity, and will not stand up to rigorous investigation.

irrecoverable”. Corollary: Regarding your program, the inclusion of a multitude of formalized “exit points” will go a long way to easing management fears of “excessive risk”.

11) Suboptimization vs. System Completion The two are incompatible. Given infinite assets to allocate toward the optimization of a subsystem (“suboptimization”), the design can *never* be completed. As engineers and analysts both support the maxim “*When stressed, people tend to do what they know how to do, rather than what needs to be done*”, they tend to keep chopping a problem into ever-smaller pieces, performing a computational function, and announcing “progress”. The #1 job of the Systems Engineer is to *define interfaces*, the #2 job is to *prevent suboptimization*.¹²

RWR: Get your Systems Engineering (SE) people *on-site* with the system developers. This means “in the trenches” and permanently located (desks etc.) on-site. Operate in *real time*. Do not attempt system engineering via “remote control” (“work reviews” conducted after-the-fact at another location; management will seldom allow “fixes” to seemingly-completed subsystems). The best SE groups operate making “minimal waves”.¹³

12) Support Groups In short, support groups “don’t”. Extensive research¹⁴ has shown the following “support groups” to be *counterproductive*..... Human Resources, Project Engineering, and Employee Relations. Systems Engineering (SE), due to near-universal management neglect, has become relegated to a support-role and therefore somewhat ineffective. There is nothing done by these types of organizations that cannot be done within a department, by the members of that department, within the constraints of a 40-hour week.

RWR: For SE members/groups, visibility and authority-for-approval are mandatory. Getting “on-site with the system designers” will go a long way toward SE-effectiveness. Getting the SE group into the interface-definition mode *before* the subsystem designers get ramped-up is even better. As SE professionals, *stick to the SE tasks*. SE-empowerment does NOT make you the “subsystem designer”.

13) Visualization of Complex Problems If you cannot visualize a problem, *you cannot solve it*. Visualization in

¹² The worst “offender” is a classified program operating under a cost-plus contract. (Limited oversight/maximum budget)

¹³ In fact, the authors observation is that the most-effective SE groups are comprised of workers who mimic Mother Theresa, and an SE manager who mimics Attila The Hun.

¹⁴ Done by the consulting firms of Data Express, and MCD Consulting; over 1000 cases were analyzed.

this case means via graph or pictorial representation in which the critical variables (and their drivers) can be seen interacting. A visual “cause and effect” diagram. A block diagram, and a simple software simulation are powerful visualization tools.

RWR: Never, *ever*, go into any meeting without a fistful of charts defining each step from start-to-finish. *Use only the minimum number of charts to tell your story.* Large numbers of “just in case” charts to cover unforeseen questions will give your project a credibility-boost. Guideline: “A good chart is worth a thousand words; do NOT use a thousand words to describe your chart”.

CONCLUSION

It is the authors observation that following each of these tenets is no guarantee of success; NOT following them most often yields failure.