

NotesOn: IT Fundamentals – Simple Defined More

Introduction (v1.0):

After posting “[NotesOn: IT Fundamentals – Simple Defined](#)”, a discussion and debate began on Herding Cats (the site referenced in my post). The result of that series of comments is quite educational and furthers the searching look into the definition of “Simple”.

While Glenn B. Alleman and I clearly come from different technical backgrounds, and our perception of “simple” when applied to IT varies, I believe you will discover that we are really not all that far apart. So, enjoy the debate.

Note: For brevity, I have shortened Glen’s comments a bit where I felt it would not interfere with his intent; the full text can be found in the comments section of his “[The Malformed Criticism of Power Point](#)” post. I have also edited my responses, here and there, to improve readability. Square brackets, [<some text>], and ellipses, ... , indicate the majority of these edits. Highlights were added after the fact for this document.

Introduction (V1.0):	1
The Discourse on Simple:	1
Summary:	8

The Discourse on Simple:

Glen’s article (referenced above) was posted on May 2, 2010. I responded on May 13 at 4:30pm with the following:

Greg,

Thank you for this post. I agree with it ... except for ... Mencken's Quote ... which you seem to refer to fairly often. My disagreement with the quote sparked an article on my site: "NotesOn: IT Fundamentals - Simple Defined" (It can be found on: www.fromtheranks.com).

*Best,
DP Harshman*

Glen (who works with the Department of Defense) replied, as follows:

Thanks for the comment and like. It is likely our definitions of "complex problem" are different and possibly the source of your irritation. My examples of complex systems - from hands on experience - include:

- 1. The rebuilding of the GIG 2.0 (Global Information Grid) ... we work this program through a NETOPS program in all theaters both CONUS and OCONUS (in and out of the Continental US).*





2. *The integration of mission capabilities for the Future Combat System.*
http://en.wikipedia.org/wiki/Future_Combat_Systems The program we work is Class I, a small UAV.

3. *The Missile Defense Agencies Ground-based Midcourse Defense ...*

As well, "smaller" complex problems we've worked include:

1. *Hubble Robotic Servicing Mission - canceled at PDR (preliminary design review)*

2. *The merger of two very large health insurance firms claims processing systems - now 150% over budget.*

As a graduate level and previous practicing Systems Engineer, the notion that decomposing complex system into smaller systems then reassembling them through analysis ignores the "killer" of all complex systems - the implicit and unrevealed interactions - unintended interactions - of these assembled components.

I say this from hands on management of "complex problems. Far from being cynical, it is an axiom of systems engineering that complex problems and their system should be avoided and simple - possibly standalone systems - should be sought. [I agree they should be avoided if possible. – DP Harshman]

Read "The Systems Bible: The Beginners Guide to Systems Large and Small," John Gall. One of Gall's quotes has served me well for decades in my work, starting in the nuclear power, on to real-time control for exothermic reactors, and best exemplified with the Deepwater Horizon, where interactions between the equipment, operations, and nature were poorly understood. Along with human error - intentional or accidental - have resulted in a complex system with no simple solution that has now come home to roost.

That quote is:

In a complex system, malfunction and even total nonfunction may not be detected for long periods, if ever. — John Gall

See: <http://www.niwotridge.com/PDFs/Fault%20Coverage.zip> for some "analysis" of how failure to recognize both Gall and Mencken [provide] insight into our naive assumptions of "simplifying" attempts lead to failure. ...

If one applied [your post's quote] "... superficial, uninspected, use of it [the Mencken quote] provides the "perfect excuse" for not doing "due diligence", i.e. not taking the time to, not making every effort to break a problem down into, what I call, "consumable components" (or elements) ..." that would simply be poor engineering.

There may be those who do that, but that does not diminish the effectiveness of the statement of Mencken - there are no simple solutions to complex problems. Failure [to] acknowledge this and act appropriately will result in continued failure.

Read Gall, he's on the shelf at most good book stores or Amazon, then we might have a better calibration of the term "complex."





Glen tagged on this comment a few minutes later:

In your post on your site, you've brought up an important topic, thanks. This brings out one more comment.

[From my post:] "... Effort is required to break a complex subject down into elemental statements and concepts. The upside is that when done right and when they are strung back together they will represent a coherent picture of, and provide answers and solutions to, the original problem. What Mencken's quote seems to ignore is that those elemental components are, and must be, simple. If not they should be worked over until they become so. ..."

[This quote ignores] the fundamental aspects of complex system - the unknown and possibly unknowable interactions between the parts of the simplified system.

This is an axiom of systems engineering. Showing this is NOT the case in some generalized and usable manner will get you a PhD and global recognition in the SE community.

I thought about this for a while and then posted the following reply on May 17th. (It is important to realize that this is not a "grudge match" on either of our parts. Its sole purpose was, and is, to expose and discuss a very fundamental issue in IT, and other systems engineering fields. I believe it a case of: we are both right.)

Glen,

First, thank you for taking the time to respond. As I noted in my post I am a great admirer of your site and your obvious experience in the PM world.

Though I cannot speak directly to DOD style projects (I've spent my time on the business side of IT rather than the defense industry), and I have not yet read the references you provided (though I will), I'm not sure that we're that far apart. A bit of background might help.

I have an architectural and civil engineering design background – commercial, institutional, residential, with bridgework, a small dam and some roads tossed in for good measure. I have been fascinated by Systems of any kind for years, in fact one could say it is my true passion. Both the smoothly working and the broken ones; there is much to be learned from what is broken and needs to be fixed. I also have 20+ years in IT (more on that is on my About page so I won't take up space here). In that time I too have had charge of some horrifically complex multi-million dollar projects.

One example that comes to mind was a system that required a dead-on accurate decision engine (surrounded by fairly flexible easy to maintain code), as there were literally hundreds of thousands of possible permutations of, potential combinations of, rules. The penalty for errors in the resultant calculations was rather steep (up to and including legal action and agonizingly noisy bad press for the company) so the margin for error was as close to zero as we could get it.

Through painstaking effort, over the course of a year my team and I stripped the business requirements down to bare earth, white boarded rules and permutations constantly, documented and flow charted every detail, and spent days, weekends, and late into the evenings with the users (over pizza, soda and gallons of coffee) going over every miniscule part of the process.





And then we started in on all of the exceptions and there proved to be many.

And then, and only then, did we truly begin the design process, during and after which we built all of the test cases – expected versus actual.

The core solution as noted was a decision engine that allowed us (and eventually the users) to not only store the rules but adapt the rules as they changed, and they did change, regularly. This was surrounded by a client-server piece (for the admin modules due to the high security nature of the data) and thin-client web components that could be securely accessed world-wide.

Each element of that problem was broken down to its “consumable component” level and then checked and cross-checked and brainstormed and “what if’d” nearly to death.

Was it an absolutely 100% perfect system? No. Of course not. There is no such critter. But we did our best to allow for changes and imperfections in our planning and cleaned up the remarkably few minor “gotcha’s” after the initial release.

So that’s the business side. Let’s move this discussion into a more “physical” arena, one perhaps closer to DOD projects. As it happens I’ve done a fair amount of studying of flight and flying (I love airplanes!) and I’ve flown [as a pilot] a bit too. I don’t pretend to be an expert on that subject but the process of analyzing situations can take one a fair ways. And so:

... since you brought it up, let’s take the small UAV [Unmanned Aerial Vehicle] you speak of. Here, as a first blush on the subject, is a brief overview of how I would dissect and attack the design process:

First I and my team(s) would define the high level requirement areas. For example (off the top of my head and in no particular order):

- 1. Mission Package Statement/Requirements*
- 2. Potential Range/Load Requirements*
- 3. Potential Operating Environment Requirements*
- 4. Potential Maneuverability/Survivability Requirements*
- 5. Potential Construction Requirements*
- 6. Potential Support / Field Support Requirements*
- 7. Potential Communications/Tracking Requirements*
- 8. Existing Technology (vs. Needed Technology)*

I would then break each down further (going down as many levels as necessary). I actually started doing so, sometimes waking up in the middle of the night to add something new to the list. But. To keep this reply reasonably short, I’ll cut it down to just one example:

- 1. Potential Operating Environment Requirements:
 - a. Natural
 - i. Temperature Limits – Min-Max*
 - ii. Weather Limits – Ice, Snow, Hail, Sand, Dust, Wind, Humidity, etc – Min-Max*
 - iii. Altitude Limits – Max*
 - iv. Duration constraints***





- b. *Human (Hostile)*
 - i. *Stealth Mode:*
 - 1. *Radar*
 - 2. *Visibility, Day*
 - 3. *Visibility, Night*
 - 4. *Noise Limits*
 - ii. *Hardening (Against):*
 - 1. *Radiation Limits*
 - 2. *EMF [and EMP] Limits*
 - 3. *Hostile Fire Limits – Ground Fire, Ground to Air, Air to Air*
 - 4. *Field Support Limits – Shipping, Handling, Storage, Fueling, Refueling, etc.*
 - iii. *Etc.*

Now, of course, this drill-down example is not complete. It would take more work to make it so. But for our purposes it should be close enough. The other topic areas are then processed in the same manner. Before the next drill-down phase begins.

The process itself requires extensive white-boarding/what-if sessions with all of the stakeholders. Initially no “hard and fast” order of importance is assigned to the potential requirement topics (they aren’t truly known yet), nor is it required that one requirements topic be fully exhausted before another is picked up. If something pops up that can’t be assigned to a topic area at the moment it is tossed onto the “parking lot” board for later discussion and inclusion.

The idea is to work the topic areas back and forth, side to side, up and down, corner to corner and push the boundaries of all possible requirements, with no question being improper (during these sessions there is no such thing as “rank”). The process must also allow for the “brain storm” in the middle of the night, and the “oh my gosh, I just realized ...” etc., with the goal being the uncovering of as many potential needs and risks as humanly possible, however they may be uncovered, until the topic as a whole is exhausted.

It is important to note that no constraint is placed on the subject until it has been suitably sliced and diced.

Then:

Reality and Risk Management are laid on over top, priorities are agreed upon, and the compromise process begins – i.e. the “triple” constraints [scope, time, cost, quality, risk, customer satisfaction] begin to make themselves felt. A key element of the process, is that conscious decisions are made on each of the potential requirements/risks on the lists, knowingly trading off one against another (where necessary), knowingly eliminating some with a conscious understanding of the effects of doing so (if needed).

It is [also] important to note that the entire process itself is part of the Risk Management strategy for the project, for out of it comes recognition of the high, medium and low risk elements.



High and medium risk elements must be flagged and tracked as such, for they require further research, prototyping, proofs of concept, etc., especially when the project envelope is being forced into new “unknown” areas.

Note: I was unable to add the following graphic to the original comment in order to illustrate the assignment of risks to each process at any given level. I’m including it here:

<p>Red Risk Analysis Required Risk Management Area High Risk Element</p>	<p>Orange Requirements Not Started Research Required No Review</p>	<p>Yellow Requirements In Progress Research Incomplete Review Incomplete May Have Issues</p>	<p>Green Requirements Done Well Researched Reviewed & Approved No Known Issues (does not imply zero risk)</p>
---	---	---	--

Process Analysis Color Code Designations

Some of the highest risk areas are bound to be in the “social engineering” aspects of a project. The mind map chart you included in your original post, the one created by General McChrystal and his team, is a perfect example.

From what I know of Afghanistan (and I have listened to and spoken with a number of folks who have spent quite a bit of time over there) he and his team have done a remarkable job of diagramming all of the primary issues. I would be willing to bet a month’s worth of meeting donuts that each one of those points drills down into several increasingly detailed layers of charts, reports and documents, with associated risk levels attached.

I would be very surprised, too, if each risk area didn’t have a number of options (mitigate it, avoid it, transfer it, accept it) that allow both the decision makers and the folks on the ground to flex their solutions as required. It is true that, when dealing with the human factor, there are no absolute 100%-guaranteed-to-work solutions, that is a given; so ... on the human side ... one does the best one can in providing solutions, and then tinkers and tweaks from there.

The social engineering side notwithstanding, from this process (if done well and thoroughly and honestly) comes a known set of requirements and conditions that make up a “pretty darned close” roadmap for the design and build phases. Are the requirements 100% dead on? Of course not. Do they have to be tracked, with proper risk assessments and resolutions occurring on an ongoing basis? Of course. Will new “fine point” requirements pop-up from time to time, that necessitate that the other requirements be revisited? Without a doubt. Do high risk elements, especially, have to be prototyped and POC’d [Proof Of Concept’d] early on? Absolutely. Does it take time? Certainly. But such an investment should result in nothing major being missed or someone has, inadvertently or otherwise, choked the process down or flat out subverted it.



Often, in the past, I have seen this process “sabotaged”, if you will, by: too many predefined “hard coded” requirements, or by “my way or the highway” attitudes that will derail the entire process, or by “commitment” levels that are subject to whimsical changes, or by internal and/or external politics (empire building, etc.), or by someone in the Sales Group deciding that to get the contract a million dollars must be slashed (they don’t care from where), or by budget slashing, etc.

All of these non-process variables interrupt the time and resources required to truly work the process, from top to bottom (high level down to the logical consumable component level), and so forth as noted earlier. Our current “instant gratification”, “rubber on the road” culture makes it difficult (though not impossible) to properly drive the entire analysis process. So:

It seems to me that the question that should be asked here is:

“Is it the fault of the analysis process itself or the environment [around the process] when a system fails?”

What is too rarely understood these days is that the time invested in the early requirements process, of any project, is more than paid for by the increased accuracy and acceleration of the design, build and acceptance phases. And that lack of understanding is, in my humble opinion, a primary, if not the trigger for quotes such as the one in your reply:

“In a complex system, malfunction and even total nonfunction may not be detected for long periods, if ever.” – John Gall

It is true, can be true. But. It doesn’t speak to the methods for reduction of malfunctions and total non-functions.

The quote reminds me of several “undetected” conditions that were bound to cause a major malfunction and total non-function. Ones that I have little doubt were raised by the engineers, ahead of time. For instance. Remember Three Mile Island in 1979? And Chernobyl in 1986? On the other hand (with a couple of notable exceptions where the engineers were also ignored), the U.S. Space program has been remarkably error free ... because (from what I have read) ... the NASA teams (government and private industry), inch by inch, step by step researched, tested and risk-managed their way into the unknown, learning from each success (and each failure) before moving on to the next phase.

On that note, before this reply gets too long, let me close with an experience based quote of my own:

“One mind, one point of view, cannot think of everything. However. With practice, and when permitted to do so, multiple minds of sufficient expertise, when coalesced into a team, can come awfully close, when permitted.” – DP Harshman

I believe such failures cannot be charged off to the process (as loosely described above) but most typically the environments around the process.

Summary:

You should now have a better idea of the “complexities” surrounding the subject of “simple” in the field of engineering (very much including IT). My contention is that, assuming qualified and experienced team members, done well it is not the requirements/analysis process truly at issue but the “stuff” that can surround the execution of the process. So true is it, for me at least, that I finally realized there is a **Seventh Constraint**, in addition to the six generally accepted “Triple Constraints” (as briefly described above). [Note: the post, “NotesOn: IT Fundamentals – The Seventh Triple Constraint”, will soon follow the release of this one.]

Certainly in social engineering situations the attainment of a perfect analysis just “ain’t going to happen”. No process, no computer system will ever be invented that will accurately predict human nature; for one individual let alone hundreds or hundreds of thousands or millions. For that condition, the social side of systems, the tool to use, as noted in my original post, is Risk Analysis - Risk Management (see link below).

The hard(ware) side of Life is another matter. In my humble (experienced based) opinion, Success is more a function of persistence in detailed drill-down requirements gathering, followed by a rational application of the “Triple Constraints”, than a heavy dependence on mathematical formulae predicting probabilities of possible outcomes (not that they aren’t important, they are just not senior). The recently expired Mars Lander (a few days ago it ceased returning signals, two months beyond the estimated expiration date) is but one example.

Imagine what it took to safely send a computerized robotic device to Mars, a hostile planet roughly 36 million miles (58 million kilometers) away, at its closest, from Earth. Consider how many thousands of processes, tens of thousands of operations, and hundreds of thousands if not millions of consumable elements had to be identified and then “what if’d” and tested by bleary eyed analysts and engineers just to get to the design phase? All of which had to be mixed in with “lessons learned”, “best practices” and a fair amount of “outside the box” thinking. And. Yet. NASA and JPL and a host of other companies did so with stellar success.

Gall’s quote *may* be true, especially if the Project Management Life Cycle is subverted in some way. I do not have an issue with this quote. On the contrary. A key focal point of Risk Management, and its processes, must be the identification of unknown, unrecognized risks. I mentioned this in several ways in “[NotesOn: Risk Management-Risk Analysis](#)”.

I understand there are risks involved in any complex system (intuitively, the more complex the system the more potential risks are present), and I understand that the threats underlying those risks *can* be left unidentified, but I do not believe that – given the opportunity and the proper application of requirements gathering and analysis processes by a dedicated team – that we have to accept Mencken’s quote as “reality”.

Hope this helps,

DP Harshman

PDF Link