

NotesOn: A Quick-Fix IT Repair Plan

Introduction (VI.1):

To do a “quick fix” repair of an IT group there are only three steps that are necessary for you to execute. The key, the motto, and the guiding principle behind these steps is "don't over complicate them". Completing these action items (they are not thinking items) will not only make you and your IT group's life easier but it will have the added benefit of helping to recession proof your organization. (Rarely do groups who actually deliver what they promised find themselves out-sourced or off-shored or out-of-business.)

While polishing up the next in the NotesOn series of articles (NotesOn: An IT Executive Checklist” is up next), it occurred to me that some executives and managers may be in need of immediate solutions to very real existing problems. In other words the alligators may be swarming and help is needed, now! So I set those posts on IT Fundamentals aside for the moment to present this short-term fix plan as a band aid:

The steps are:

1. Stabilize
2. Simplify
3. Standardize

Stabilize

Before you can get anything else done you must first stabilize your organization. Chaos, of any magnitude, is seriously detrimental to the production and support of IT systems. To stabilize do the following:

- bring an end to endless reorganizations. Almost any organizational structure is better than constant change. Of course some are better than others. Particularly those that encompass these points.
- publish the organizational chart. It is not a “top secret” document!
- bring an end to the constant defining, redefining and talking about roles & responsibilities. You can't expect your teams to know what they need to do if you don't tell them. They will take responsibility, if you tell them what you expect them to do.
- publish the roles and responsibilities – even if they're in outline form to start with get them into the hands of your teams and then enforce them.

More chaos has been generated in the name of “we don't have time to get organized” than ten, thousand page, volumes could possibly describe. Being able to order anybody to do anything at anytime is not a virtue, it is a disaster waiting to happen ... continually.

- acknowledge what has happened in the past. Errors are errors. Screw ups are screw ups. Glib “PR spins” won't change that.



While you don't want to live in the past, you do need to learn from it. There is an old saying that goes something like: what you don't know about history you are doomed to repeat. Except it goes further than that: what you don't acknowledge about history will leave you trapped in it.

The primary value of history (outside of a source of interest for hobbyists) IS what can be learned from worked and what did not. Do not be denied the benefits of the "lessons learned" by those who chant "don't live in the past" or other similar clichés. Such "flack" is avoidance at best, and destructive at worst.

Besides, your IT teams are well aware of what has happened, and what has gone wrong. They know BS when they hear it. Be straight with them and they'll end up on your side. Lie to them and you lose them and any respect they had or will have for you. Truth works. Lies don't.

- talk to and listen to your Techies. If they have any "chops" at all, any experience in your IT group at all, they generally know what's going on. Ignore their input at your peril. They know who the good and bad managers and executives are. Listen to them. They know a technically un-sound solution when they see it. Listen to them. They will work their hearts out for you. If ... you listen to them.
- put a simple plan together to get the above done and then get it done

Simplify

- remove unnecessary and/or redundant organization layers
- change the focus of your organization from administration-centric to technical- / delivery- centric
- put support layers behind not in front of the technical teams; all non-technical IT groups must be in support of your product and service delivery teams. As a slightly cynical reminder: paperwork is not delivery, it is paperwork.
- fire any executive or manager who insists their administrative group is more important than delivery of products and services. It may be necessary work (Finance, Audit, HR, so called Business Relations groups, etc.) but it is not more important. Even architecture and security are secondary to your deliver teams.

Organizational Law: if it doesn't produce a product or service that is directly paid for by the customer then it is not a delivery group.

Design and development is delivery. Testing is delivery. Support is delivery. Infrastructure is delivery.

All other IT groups are not. Despite any efforts at "empire building" they are there solely and only to support delivery or to satisfy some government requirement.

- you want the technical teams face-to-face with the users not separated by ever-changing layers of administrative processes and endless procedures, exacerbated by the intervention of non-delivery personnel.

The more organizational layers between your technical delivery teams and your users the less delivery you will get. Period.



- stop experimenting with every new business model and “super organizational software package” that comes along. IT is not that difficult to manage, unless it is made difficult, unless unnecessary complexities are added. More often than not the only winners of the latest “new and improved” are the sales people and consultants, not IT. If it ain’t broke, don’t fix it. If it is broke, fix it simply.
- stop jumping in bed with every new “bleeding edge” technology that comes along. It *may* have value. But does it have more value than what your teams know already? Have the sales people (and your own Techies) prove to you, via proofs of concepts and test programs (pilots), etc., that “it” truly has more value than the existing technologies you and your teams know and use.

Change is expensive, very expensive. It takes “a lot” of in-place value to generate ROI. You can’t generate ROI while everyone is in yet another steep learning curve.

Standardize

- use a development methodology that has a proven track record of success -- teach it, implement it and enforce it.

(Note: I do not recommend Agile or Agile-esque methodologies, they violate too many fundamentals. If you have to use one, because you just have to be on the "bleeding edge", then go ahead ... but make sure you do all of the steps, and then plan on plugging the leaks these “accelerating” approaches leave behind.)

- develop, or implement if you have one, a simple straightforward in-take process that, again, doesn’t require four hundred administrative layers to get through. Think simple. Plan simple. Execute simple.
- develop, or implement if you have one, a simple monitoring process that keeps track of your projects and trouble tickets. More millions (perhaps even billions) have been wasted on “IT metrics” than you would probably believe. It’s simple folks:
 - you receive a project
 - the project is in the budget or not, if not who pays for it
 - you gather the requirements; the delivery team members must be involved, they must not receive them second or third hand
 - you plan the project against a time line; the delivery team members do the estimate, they have to live with it
 - you design and build that project, with liberal customer interaction and simplicity as a watchword
 - you monitor the project against the time-line
 - you brutally prevent scope creep; keep a “wish list” for Phase II
 - you make sure contingencies are for “unseen, unknown, who could have imagined” issues, not sloppy estimating or scope creep – no sand-bagging



- you prevent the users from constantly changing their minds – one of the pitfalls of Agile – they too have to commit ... which is why they must be intimately involved in requirements and design
- you build it, per requirements and design – no on-the-fly “cowboy” changes
- you test it and fix it
- you deliver it and support it – three months of “hyper care” then it goes onto regular maintenance

Notes:

Below are some of the notes I accumulated over time that went into this NotesOn:

- communications must be accurate and swift
- too many layers create unintended filters that invariably:
 - alter priorities & importances
 - lose key information and concepts
- processes & Procedures must be both simple and effective
 - avoid P&Ps for the sake of "something to do"; or empire building
- hoping doesn't improve anything, looking, asking and fixing do
- define success. It will vary for your IT organization but must include:
 - user's definition
 - technical's definition
 - management's definition
- decision points must be defined -- a balance must be struck between user "want" and IT budget and resources
 - must involve technical input and buy-in too
 - must include how to get eager user buy-in and participation
- all plans must result in "action items" that are: capable of being executed (as opposed to being "filed and ignored", i.e. a PR stunt), are assigned to someone(s) to execute and are then monitored for completion ... and success.
 - don't be afraid to fix a broken plan; your ego is not more important than your organization

Hope this helps.

DP Harshman

