

NotesOn: The Four Fundamental Life Cycles of IT

Introduction (V1.3):

This post addresses the four, not three, IT (Information Technology) Life Cycles that are on the “must know” list for those working in and managing an IT organization. One of the four is a new Life Cycle (LC) that while present and in use has never before been named. All four Life Cycles interact and interrelate continuously, whether we are fully aware of them or not. Executing well on all four will lead towards success in IT. Why? Because they are *fundamental* Life Cycles.

As I’ve noted elsewhere, and will continue to do so in the future, managing any part of an IT organization without knowing and applying the *Fundamentals of IT* is akin to, nearly identical to, pushing a boulder up a steep ice covered mountain (Sisyphus of mythology fame comes to mind). You may have experienced this level of frustration in your own IT career from time to time. Managing all or part of an IT group on “instinct” *can* be done but is generally a serious exercise in near futility, and knowing the rules of business (a la BA’s and MBA’s) is not enough. Because IT has its own fundamentals which are separate from (though allied with) generally known (but not always applied) best business practices. IT is a “bird of a different feather”.

This post provides an overview of the four basic Life Cycles of IT and how they apply. It is the first of a new series of articles exploring the Fundamentals of IT in increasing detail.

Introduction (V1.3):	1
What is a Life Cycle:	2
The IT Life Cycles Diagram:	2
1. Product Life Cycle:	3
2. Project (Management) Life Cycle	5
3. Systems (Software) Development Life Cycle	7
4. Risk Management Life Cycle:	9
Risk Unknown	10
Risk Identified	11
Risk Analyzed	12
Risk Managed	12
Risk Monitored	15
Risk Retired	16
Why Know and Use the IT Life Cycles?:	16
Summary:	17



What is a Life Cycle:

A Life Cycle (LC) is a sequence of, a connected series of events that have a beginning, a middle and an end. No reference is made to “duration” as this varies, sometimes wildly, by the type and application of the LC.

The IT Life Cycles Diagram:

Before beginning an executive summary dissection of each individual Life Cycle let me present a diagram (Figure 1) that shows their relationships. It includes a brand new Life Cycle that has never before been identified, though elements of it have been in use and discussed for a good many years.

IT's Life Cycles: Relationships between Product, Project, Risk Management and System/Software Development Life Cycles
© DP Harshman – www.fromtheranks.com

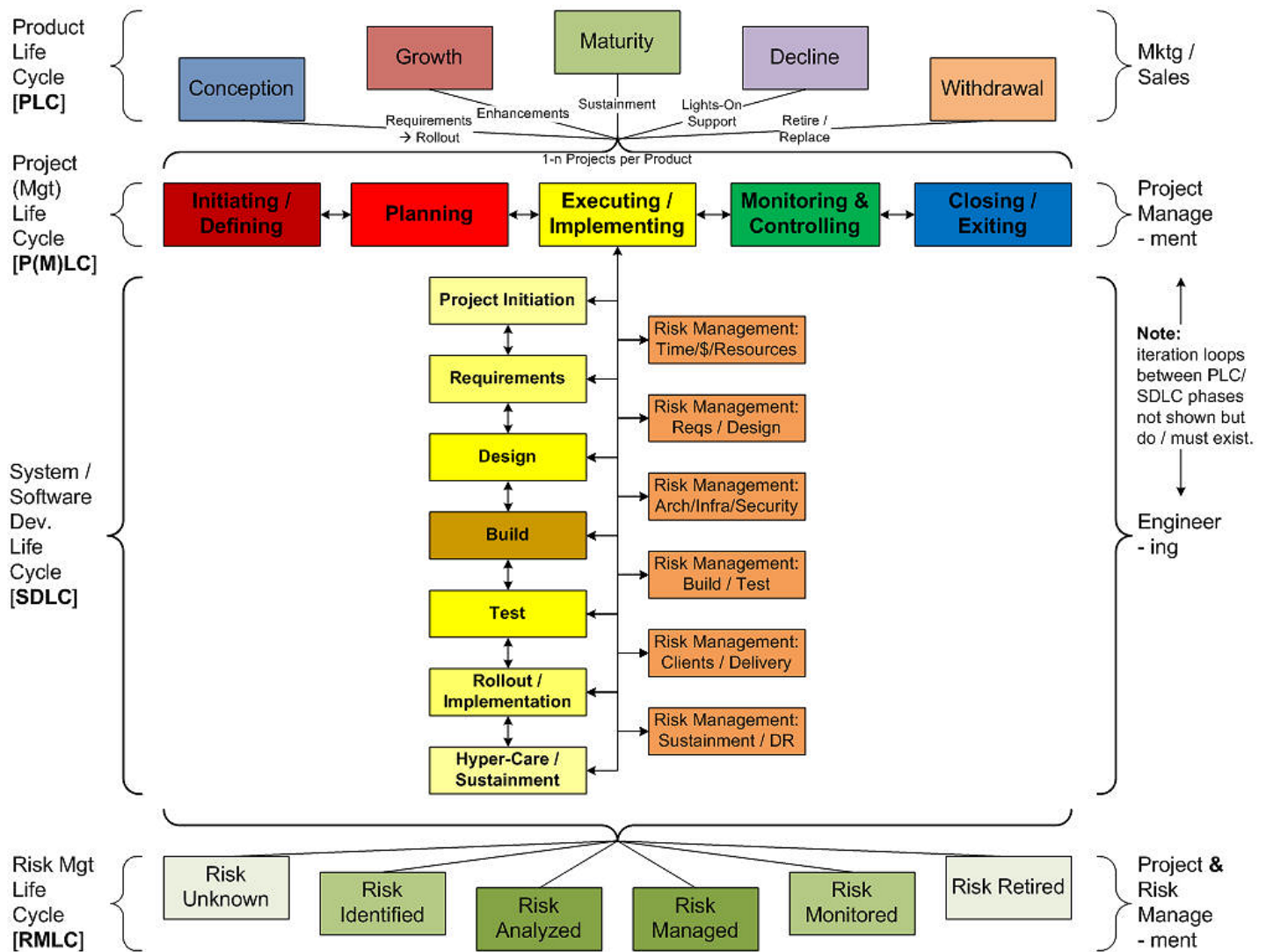


Figure 1 - The Four Fundamental Life Cycles of IT

Let's start at the top and work our way down.



1. Product Life Cycle:

Because the PLC acronym is used for *both* Product and Project Life Cycle, these two subject areas are often confused. They are, however, quite different. Interrelated but different. The Product Life Cycle phases are:

1. Conception
2. Growth
3. Maturity
4. Decline
5. Withdrawal

If you look into any product (or service) you will find this cycle was or is active. Nor does it matter the type of product or service. From buggy whips to the latest high tech electronic gizmo these phases are in play. The duration of the entire life cycle will vary – from minutes to decades – but there is no denying its existence.

Let me say again, for emphasis, the Product LC applies to all products and services ... very much including those generated by IT.

Any IT organization that does not manage their products and services (including marketing and selling of their products and services) from the beginning of the Life Cycle to the end of the Life Cycle is missing a beat and the boat.

There are three basic ways the PLC can be mismanaged: (a) by being ignored, (b) by being over-done, i.e. micromanaged to the point it is never done, or (c) by being short-circuited mid-stream by “fiat”. (Per the Encarta Dictionary “fiat” is defined as: “2. arbitrary order – an authoritative and often arbitrary command”.)

Choice “c”, by fiat, is particularly destructive. Too often I have seen products developed and interfered with in just such an arbitrary fashion, i.e. orders were sent down from “on high” and suddenly goals and directives are changed; priorities are abruptly altered; previously approved budgets are unpredictably slashed; scope and requirements are unexpectedly modified; existing viable products are altered or prematurely terminated; and so on; thereby creating chaos. Management by “whim” is the antithesis of management by Product Life Cycle.

This does not mean that product and services shouldn’t be responsive to changing conditions, on the contrary. That is, in part, what this Life Cycle is about. However. The change has to make sense. It has to be appropriate. The change has to address and satisfy needs, desires and risks and provide an ROI to all of the players at the table, each and every time. Change by fiat and change for the sake of change are deadly; they inject an organizational poison into the veins of the Product Life Cycle that is often fatal.

Almost nothing could be harder on an IT organization than failure to execute on the Product Life Cycle. Again, this cycle applies to services as well. They are, after all, products, too.

One negative aspect of failing to do so of failing to understand and execute on the Product Life Cycle is reflected by what I call: “distance between”.

In successful software houses (businesses whose sole products and services are software/application related) the connection between what is *done* and what is *earned* is very real and immediate. The *distance between* what the right and left hands are doing is nearly non-existent. Done and Earned as a management pair are almost literally “in your face” day in and day out, sometimes hour in and out. However:

As one scales up from small to medium- and on to mega-corporations an increasingly invisible distance in this relationship appears, an ever widening gap between what is done on a daily basis and the paychecks handed out at the end of the pay period can be observed.

In mega-corporations in particular (though hardly the only ones) there are usually so many trailing zeroes in the bank accounts and so many administrative, organizational and command-and-control layers between the products produced and the producers that the *distance between* Done and Earned appears to be nearly infinite, i.e. the connection between the two virtually evaporates ... despite the fact that every plan, every action item, every activity, every to-do list, every task, every delivery, every product and every satisfied (or dissatisfied) customer is directly related to each and every individual paycheck.

When that relationship vanishes as a working, living, breathing concept “Close enough” and “Good enough” become the standard and norm. When that *starts* to happen your IT group’s “failure horizon” draws closer. If that continues as a modus operandi the failure horizon collapses on you and your IT group will cease to exist.

The anatomy of a “failure horizon” is quite simple. It consists of:

An ongoing accumulation of daily, weekly, and monthly *failures* to deliver products and services that are viable, desirable and sustainable; products and services that allow your customers to do something better and provide *them* an ROI on *their* investment too.

Failure horizons may appear to be further away in mega-corporations than in smaller sized companies but when fundamentals are not understood and practiced that is an optical illusion brought about by scale not necessarily by distance. How many mega-corporations have gone “belly up” in the last two years alone?

By the way, the anatomy of a “success horizon” is as simple. Grammatically replace “failure” with “success”.

I did mention this post was written at an executive summary level so we are going to leave the PLC behind for now (we’ll pick it up again in a future post). But. Before we move on here are two questions to contemplate:

1. When (if) Product Life Cycle management is done, is the entire plan, or any of it, communicated? For each product and service? In a way and with a clarity of purpose that “the troops”, all of them, can understand and get behind?
2. Are your Five Year Plans (though these days they should be Three), Annual Operating Plans (AOP’s), Annual Budgets and Annual Resource Plans done for the sole purpose of enlivening, supporting, extending or terminating each instance of the Product Life Cycle?

2. *Project (Management) Life Cycle*

Project Management has been with us far longer than IT has existed. I suppose one could attempt to go back and look for archeological evidence of the exact methodology used by the Egyptians in building the pyramids and so on but while that might be interesting doing so is hardly necessary. Without digging through dusty tombs what is known for certain is that engineering disciplines have been successfully completing projects for centuries, and infamously failing at some occasionally. From such endeavors (the good and bad) a set of fundamental steps evolved that has been, for many decades, called a PLC, or more precisely a PMLC.

About the most basic Project Management Life Cycle (PMLC) there is, is:

1. Feasibility Study (Why do it / Can it be done)
2. Plan (What can be done, How much will it cost, Who can do it – high level)
3. Design (How / When / Who / Where can it be done – detail level)
4. Build (Getting it done)
5. Complete/Turnover (Done)

We can, and eventually will, go into this Life Cycle in more detail in another post as well but, for now, suffice it to say that the above sequence has been around for a good long time and is, essentially, what I learned “way back when” in my engineering and architecture days (yup, I were one and could even spell it ... an inside joke).

Not at all surprisingly, the above steps / phases look pretty much like the Project Management Body Of Knowledge’s (PMBOK®) Process Groups of today:

1. Initiating
2. Planning
3. Executing
4. Monitoring & Controlling
5. Closing

Interestingly enough the early PMLC steps are quite similar to what we now identify as an SDLC as well. What has happened is that over the years PMBOK® separated the management skills from the technical/engineering skills, thus forming the “T” we see in Figure 1.

So important is the PMLC to IT’s continued success that I have spent years ranting and railing (politely) and hinting (nicely) that Project Managers need to be trained on, educated in, and gain extensive experience with the Project Management Life Cycle. One company spent two years flailing around and wasting a lot of time

and money on failed projects before the light bulb finally went on. When it did, that company endeavored to train every one of their would-be PM's (and Business Analysts and Tech Leads) on PMBOK® (and on SDLCs). And. Then. Their successful project completion statistics started to rise. Not 100% because not everyone absorbed the knowledge equally and a certain level of politics continued to interfere but, nevertheless, a significant improvement was obtained.

Lesson Learned: it is safe to say that one of the, if not *the*, primary root causes behind failed IT projects is: untrained IT project managers. This is more fundamental as a root cause than most of the "failed project surveys" out there. It isn't that those surveys are necessarily wrong, they just aren't right enough.

Because someone is given the title of PM does *not* mean they know what they are doing (believe it or not this happens all too often). They may well be expert in a different subject area but unless they are trained *and mentored* on and have hands-on experience with the technology of project management they will fail (usually if not always). Because ... Project Management is its own skill.

In a like manner, that an IT group contains trained and experienced PM's does not mean that a company's upper management and executive levels have a clue about the PMLC; their ignorance can inadvertently (or otherwise) interfere with proper, sensible and appropriate PM processes and procedures. Thus, it is best to train and mentor *everyone who is even remotely related to IT projects and products*, from top to bottom.

By the way, PMBOK® is not the sole-source end-all-be-all on the subject of Project Management. It is spot-on, there is a tremendous amount that can be learned from it, the education it provides is unparalleled, but what it describes is a distillation of project management "good practices" and as such is an outline or a framework not a cover-to-cover answer to all questions. The data is invaluable but PMBOK® is not a cook book that one can blindly follow from Page 1 step one to Page 297, beginning at the beginning and ending at the end, until the project is done. One has to, you have to know the PMBOK® data and you need to have experience applying the PMBOK® data before you can be routinely successful with the PMBOK® data ...

Hmmm. I might have to take that back. Just fractionally. *If* as a "newbie" you read, comprehend, follow and do each of the elements in each of the process groups you just *might* exit the other side, the Closing group, with a successful project under your belt. It would take a while as you and your team(s) would have to learn from your mistakes along the way ... it would be a rather painful way to do it ... but ... PMBOK® *could be* cook-booked. However:

The better and best way, the least painful way, the most time and cost effective way to come up the PMBOK® and PMLC learning curve is by having an already experienced mentor (or two) looking over your IT group's shoulder while everyone (executives, management and techies alike) is learning, and doing.

Getting back to my original point ... there is more to learn on the subject of Project Management than what is contained in PMBOK®, as pointed out in a post by Glen B. Alleman of Herding Cats (a top-flight IT project management knowledge site): http://herdingcats.typepad.com/my_weblog/2009/11/project-management-books-1.html.

And *Learn* is the operative word. The PMLC, and being a PM, is not a theoretical concept. It was conceived from learning and doing and requires learning and DOING on a first-hand basis to obtain proficiency.

Little about Project Management and the Project Management Life Cycle is intuitive, virtually none of it comes “naturally” (there is no such thing as a “natural” PM). PM fundamentals evolved and were recognized over many decades and must continue to not only be studied but “passed down” from one “generation” to the next.

The expertise to know what is appropriate in a given project circumstance comes solely from experience based on transitioned knowledge. And that dictum applies to IT Execs and Senior Management as well. You cannot, cannot, cannot manage what you do not at any level understand.

An upcoming “NotesOn” will delve more deeply into PMLC’s and SDLC’s so for now let me close off this section with the following summary:

- PMLC’s are what Executives, Managers and Project Manager’s *use* to obtain successful projects.
- The PMLC, as implemented appropriately for the project, is the over-arching guide for the entire project.
- The PMLC does not replace the “engineering” portion of the project, it is the umbrella under which the “engineering” portion of a project will get done.

And, to connect the dots fully,

- A PMLC is and should be used many times during the life of a Product (or service), hence in some ways it is subordinate to the PLC.

3. Systems (Software) Development Life Cycle

The third IT LC, which we have also had since nearly the beginning of time itself, is the System (a.k.a. Software) Development Life Cycle. As with the PMLC, the following outline did not spring up out of nowhere, it evolved. It was observed, recognized, discussed, tested, fixed, re-tested, over and over again. And yet. You would be amazed at how many managers, project managers, tech leads, and development team members cannot rattle off this sequence in their sleep.

First thing to know is that an SDLC is *not* a PMLC. Though there are similarities, though they evolved from the same roots, they are different life cycles. The PMLC is used to run, to manage, the entire project. An SDLC is geared towards the engineers, the techies, who are developing a specific system or part of a system. As you will note in the Four Life Cycles diagram above, the SDLC is a “sub-set” of the PLC, it is subordinate to it. One reference regarding this disparity and confusion can be found on Wikipedia:

“In [project management](#) a project can be defined both with a [project life cycle](#) (PLC) and a SDLC, during which slightly different activities occur. According to Taylor (2004) "the project life cycle encompasses all the activities of the [project](#), while the systems development life cycle focuses on realizing the product [requirements](#)".^[5] -- http://en.wikipedia.org/wiki/Systems_Development_Life_Cycle

To be very precise, the SDLC for IT embraces what successful tech leads and systems/software engineers use as a *template* (an outline of best practices) to successfully build IT's systems.

The fundamental phases of an SDLC are:

1. Project Initiation (similar to but different from the PMLC's Initiation phase)
2. Requirements
3. Design
4. Build
5. Test
6. Rollout / Implementation
7. Follow-up / Hyper-Care

These phases with slight modifications in terminology apply to many other classes of development such as found in engineering, architecture and construction, for that is where their roots are buried. What is often forgotten is that IT was built on and still rests on the shoulders of past engineers, architects and construction specialists who learned their "success lessons" the hard way.

Some of the "new kids on the (IT) block" label the above SDLC phases as antiquated, pigeon-holing them under the heading of "legacy" and thus unnecessary. One reason they do so is simply because these steps *are* based on *past* engineering, architecture and construction principles. Another is they feel that such a "linear" methodology can't possible work in their "new" "<fill in the blank> 2.0" software development environments.

Something, anything, newer, faster (and less restrictive) is demanded by the "new kids" as they do not realize, and have never understood, that the above SDLC (and PMLC) phases are anything but, never have been, linear and are certainly not "legacy" (as in "old", "antique" or "worthless").

These steps have survived every "revolution in ..." every "latest fad" ... because they worked ... when used ... and continue to work ... when used.

If you have been around IT for any length of time I am sure you will recall how often UNIX was categorized as "legacy" (with the emphasis of a four-letter swear word) and pronounced as "dying" if not outright "dead". And yet UNIX (and its variants) are very much alive and well. Because its fundamentals are sound and it fills a fundamental need unmatched by any other operating system.

Yes, some folks have over-complicated the above SDLC template steps (in discourses amounting to multiple volumes containing hundreds and hundreds of pages of excruciatingly defined deliverables). Just as others have under-complicated them, severely.

But, as with “Goldilocks and the Three Bears” when you apply appropriately scaled “just right” SDLC phases to the project in front of you ... within the framework of the four fundamental Life Cycles of IT ... they work.

Fundamentals are fundamentals for a reason. When you violate fundamentals you fail. When you short-cut fundamentals you fail. When you consistently (as opposed to sporadically) apply fundamentals you succeed. Yes, you can go beyond fundamentals (the “outside the box” stuff) but *know* and *do* the fundamentals first.

No great artist became great without knowing and using their fundamentals. All notable painters knew, cold, how to mix paint, knew about color, knew about dimensions and perspective and so on. Once they no longer had to “think about” any of their “tools” they were able to build on, and reach beyond, their fundamentals; but they never forgot or ignored them. Great musicians knew, cold, how to read musical notes and about harmonics and about dissonance and about their instrument(s) and about rhythm and ... to the point they didn’t have to think about any of their fundamentals either. Then, they moved beyond average to greatness.

Do not leave IT fundamentals behind in your quest for “world class” greatness. It isn’t possible.

I will be diving into the subject of SDLC’s in much more detail in upcoming “NotesOn” posts, so allow me to conclude this segment by re-iterating, re-stating that, as with other engineering fundamentals, violating (one could say short-circuiting) the SD Life Cycle phases can (and usually will) have detrimental consequences. But one reason that this is true is that critical risk management steps are eliminated when their SDLC counterparts are done away with.

In the above diagram, next to the SDLC steps there are a series of “Risk Management:” functions. These need to occur as an ongoing part of any system/software development project. Please do not brush them off, either. They are necessary parts of an overall risk mitigation strategy that, along with an appropriate-to-the-project SDLC (under the umbrella of the PMLC), will help to ensure success in the fullest sense of the word.

[More on this subject will be presented in the upcoming “NotesOn: Agile Software Development – A Forensic Analysis” and “NotesOn: The Optimized Software Development Life Cycle (OSDLC)” series of posts.]

Speaking of Risk Management ...

4. Risk Management Life Cycle:

This is a brand new, newly identified, Life Cycle. Up until the publication of this post I did not and you could not find any reference on the web regarding: “Risk Management Life Cycle” (RMLC). Heretofore it was never recognized as such, i.e. that Risk Management had a beginning a middle and an end (as noted at the beginning of this post, the fundamental definition of a life cycle). [Note, please see “[NotesOn: Risk Management Life Cycle Addendum](#)” for a correction to this statement.]

What is not new is the subject of Risk Management. It has been with us for a while. Unfortunately, though, it was too often ill used or not used at all, in part I believe because it was often perceived as a never ending

continuum that could and would never be completed, i.e. it went on forever and ever and ever ... Forever and ever and ever can be rather disheartening. Most of us human beings do like to complete things.

It is my hope that by recognizing the RMLC for what it is, that the entire subject of Risk Management can be both contained and more efficiently managed ... from beginning to end. Here are the RMLC phases:

1. Risk Unknown
2. Risk Identified
3. Risk Analyzed
4. Risk Managed
5. Risk Monitored
6. Risk Retired

I included a section on Risk Management (and Risk Lists) in “NotesOn: An IT Executive’s Survey And Checklist – Part IV”, and the upcoming “NotesOn: Risk Management-Risk Analysis” is dedicated to detailing the RMLC so for now let’s review the above phases at an executive summary level.

Risk Unknown

Each and every detailed risk analysis must include the concept that there are unidentified, unknown or not fully known (which is nearly the same thing) risks out there. This is where thinking “outside the box” comes into play. The word “assume” must be stricken from each and every risk management process. You may come back later and decide a particular risk has a very low threat factor, but you won’t know that until and unless you first identify the unknowns.

Many years ago I ran across a business practice in a Japanese company that impressed me a great deal: the notion of the unrestrained asking of “what if”. During their design processes they did not limit the possibilities to what was known, to what was within their comfort zone, or to what they assumed to be true. They asked “what if this happens” “what if that happens” “what might occur that we haven’t thought of”. Reportedly all politics were set aside, and all ideas, if even remotely related, were fair game. In the process of running this process to completion they not only came up with better designs, and processes, they ended up limiting, or at least severely reducing their risk levels.

Almost as soon as I heard of it I modified the technique for IT and it became my: the “titles and positions don’t matter” the “no question is too stupid” the “any voice can be heard” the “what haven’t we thought about” approach to successful designs and builds. And. It worked. And works. Not to pat myself on the back but, using this approach, no “in production” system that I ever had a hand in designing and building from the ground up was ever hacked, ever lost a penny, or ever mislaid or altered information. This approach applies to all of IT, not just to the design and construction of systems.

The worst risks are the ones that you don't know about. And they *are* out there, especially if you or your organization haven't done a detailed Risk Analysis. The old "saw's", the old adages of "no news is good news" and "what you don't know won't hurt you" are so far off the workability radar as to not even be laughable.

The first step (really the zero step) in Risk Management is recognizing that we live in an imperfect world and in an imperfect world risks exist.

The second step is the recognition that there *are* risks out that you've never thought of, that you've never anticipated and that you don't know the consequences of were they to impact your organization.

Of course common sense must be used when doing Risk Management, you will never eliminate all risks and you couldn't afford to if you could. Nor is it the objective of Risk Management to create a state of paranoia in your organization. But, you can't not do it because you or someone might think that you are paranoid. After all "stuff" does happen. So. You can either put yourself in the position of proactively anticipating or leave yourself in the position of reactively responding.

Don't forget internal threats – to a certain degree with out-sourcing but most particularly with off-shoring the number of risks have escalated significantly and their threat levels are far higher than if your operations were home-based. Off-shoring opens up a huge panorama (possibly a Pandora's box) of risks, huge.

Risk Identified

The most rudimentary analysis process begins with the following action items:

- Identify Potential Vulnerabilities
- Identify Potential Threats (Threat Recognition)

The recognition of potential vulnerabilities (internal and external) is a somewhat boring process (to be honest) and it tends to stir up a fair amount of political "dust" but the process of doing the process is very educational for all involved and is well worth the "stepped on toes" and "mussed up sand-boxes". And don't forget to not forget the "small stuff".

You have a starter list of "vulnerabilities" documented already in SOX, PCI, PII, HIPAA and Safe Harbor audit control lists for IT. That's a start though, not an end all. And don't forget the potential vulnerabilities provided by other areas of the company where they have a direct or in-direct impact on IT.

Similarly, potential threats / threat sources must be searched out and recognized. The newspaper, TV and radio provide some information (a constant stream of it actually that requires filtering) but there are other sources. For instance, depending on the size and scope of your company and IT group, and whether it is local, national and/or international, there are both governmental and NGO (non-governmental organizations) that provide threat assessment information.

Two rules of Risk Identification:

1. Where you have a vulnerability and a matching threat(s) you have a genuine risk.
2. If you have a vulnerability that does not (yet) have a matching threat(s) you have a potential risk.

Risk Analyzed

Risk analysis is a skill best served up with experience but it is a skill that can also be learned from scratch and then polished as you go. It is a “better some than none” proposition. We’ll go into it in far more detail in the upcoming “NotesOn: Risk Management-Risk Analysis” post but, in brief the steps of risk analysis are:

- Identify Vulnerability-Threat pairs
- Identify potential Vulnerability-Threat pairs
- Determine Likelihood of Occurrence
- Estimate Potential Impact of Threats
- Present Risk Analysis and Mitigation Strategy

I promise, there is a simple and effective method and process for executing these steps but its description exceeds the length constraint of this post so I have no choice but to force you to wait for the next “NotesOn:”.

Risk Managed

This phase could also be called Risk Limited but the word “managed” really nails it and is what you *must* do with all high and medium level risks. This phase is where you do the planning, where you apply the Risk Management strategies to your identified risks.

As you have likely discovered if you’ve had any dealing with IT Audit controls there are four ways to “deal with” a risk:

- Mitigate it
- Avoid it
- Transfer it
- Accept it

These seem obvious enough as decision points, until you delve down into them and until you try to apply them in the real world. Let's take a look at how they are used to manage risks:

Mitigate it:

Mitigation can be defined as an action taken to lessen or alleviate a risk or potential risk to reduce its seriousness or harm. So one response to a risk is to come up with "an answer" that materially reduces the potential harm to the identified vulnerability. As an example, a common Vulnerability-Threat pair is "intranet-hacker". Mitigation steps against it should include isolation, firewalls, strong passwords, granular authentication and authorization, installed and regularly updated and patched security software, etc.

Avoid it:

Avoid (altogether) could be the chosen response to some potential risks. If the risks are too high, if potential risks exceed any potential value then one answer is to just not do it (no matter how "popular"). One such example would be dealing with the risks (and they are many) associated with off-shoring (public relations issues, customer dissatisfaction, security leaks, data theft, data "borrowing", disaster recovery "issues", communication link and delay issues, language barriers, cultural barriers, etc.).

Transfer it:

A third option is to transfer the risk to someone else, i.e. dump it in someone else's lap. Insurers do this all the time. An insurance company doesn't necessarily carry the entire potential loss in a given category (such as fire, earthquake, flood, etc.) on its own. For a fee it "shares the risk" with other companies, and they may share theirs in return.

Out-sourcing is one way of transferring some or all of a potential risk but there are often down-sides, there are usually side-effects to so doing. For instance, a typical though hardly universal "transfer" is to out-source basic infrastructure and desktop support. For the most part this has made sense for companies as third party providers can generally buy and rack servers and buy and setup and support desktops at a volume based price point that most IT groups can't match. Though some notable exceptions come to mind, third party providers also have the seasoned IT teams to support your group's hardware and desktop installations. However:

There are always, repeat always, negative side-effects and unintended consequences with even such a "traditional" and "easy" transfer of power to an out-side agency.

Let me phrase that differently so that it really stands out:

When you transfer risk you also transfer power, over you and your organization, to the out-side agency(s).

That factor alone adds an element of risk to the "transfer it" strategy. And, adding to the difficulty factor of this solution class, out-sourcing infrastructure and desktop support is about the only "easy" decision there is when it comes to transferring IT risks.

Tip: I was discussing this with a good friend of mine, Alan Frawert (who also has extensive IT experience) and he suggested that one way to help mitigate the risk of out-sourcing your infrastructure functions is to take a gradual, graduated, approach (as opposed to “all or nothing”). Whether you are transitioning to an external dedicated Data Center solution or moving to an out-there-somewhere “cloud”, move your systems over one at a time. Starting with the least critical ones, test-drive the vendors infrastructure and support levels as the gap between sales promises and actual capabilities can be, and often is, vast. If your “proof of concept” reaches an adequate provision and support level, build that level into the SLA (Support Level Agreement) then hold your vendor’s feet to the fire. And if it doesn’t work? Don’t be afraid to “pull the plug”.

Off-shoring is another way of transferring risk. But. The actuality is that when you send some or all of your IT organization’s functions across one or more borders you are, not may be, but *are* adding more items to your risk analysis list than you are getting rid of.

The list of risks, and potential risks, related to off-shoring functions are nearly impossible to detail in a reasonable amount of space. Off-shoring consultants make it sound easy and dangle huge “savings” in front of everyone that has a say, but, and I speak from experience, the *true* cost of off-shoring is e-x-p-e-n-s-i-v-e ... by the time you identify, address and mitigate all of the “unknown” and “unmentioned” risks and unintended consequences the actual price tag is many times the consulting firms’ estimated costs ... by a factor of 2X-4X.

And that does not include the costs and additional risks incurred when you decide it is time to bring those functions back to home-base again. Nor the costs and actual risks associated with “upset customers” and “ruined local economies”.

The ugly hidden truth of off-shoring is that every single cost estimate has been low-balled to make the deal, to make the sale. Subsequently, later on, the true costs have to be buried in non-off-shoring related accounts to hide that fact. It almost goes without saying that the original “cost saving” estimates in being so far off the mark are never mentioned again, either.

This is why I have stated elsewhere that while *expansion* overseas can make sense (technically speaking this is not off-shoring), dumping costs and jobs and responsibilities overseas does not. As in not ever. Why? To put it very simply and succinctly:

While off-shoring nominally transfers risk into someone else’s lap, and may reduce some costs on a short term basis, it *also* transfers power and, ultimately, control and profits.

I probably should flag both of the above emphasized statements as rules of, maybe even laws of business.

Why does expansion overseas work and off-shoring not?

Heading overseas for expansion is a “new “ (... new facility, new personnel, new management, new regulations, new customers, new cost and price points locally, new opportunity, new ...). While expansion has its risks, new is a “positive”.

Heading overseas to transfer risk (and responsibility) is a “disentangling”, a surgical procedure with a not very sharp scalpel (... disentangling existing functions, disentangling existing entities, disentangling existing processes and procedures, disentangling existing relationships, disentangling local jobs from local economies, disentangling ...). Off-shoring to cut costs (etc.) is a “negative” from beginning to end.

These two are worlds apart. The first is done in daylight. The second is done under cover of darkness (under the radar). The first is primarily constructive. The second is primarily destructive. The first builds prosperity. The second destroys it. The first breeds good-will. The second sacrifices it.

Hint: the best answer to “overseas” is to identify, educate and reinforce fundamentals. No well managed group that knew its fundamentals cold and, as a rule, delivered viable quality products and services which provided its clients with a true ROI on their investment ever suffered off-shoring.

Understand that “transfer it” is, by its nature, a shifting of responsibility. It is not a “one way ticket to success”. You *are* always giving up something, minimally reduction of power and control.

By the way, one of the most critical, and more difficult, aspects of “transfer it”, is developing an honest, unbiased, analysis of *all of* the pro’s, con’s and recommendations. Finding a consulting team who does not have a vested interest in “the answer” is tough! This is why “transfer it” risk analysis is a very high risk endeavor.

Accept it:

The fourth option, accept it, should be reserved for the lowest level (lowest tier) Vulnerability-Threat pairs. “Accept it” by definition, at the purest level, is equivalent to saying: “We’ll ignore it and take the risk results come what may”. This “no action” solution is roughly equivalent to “turning your back on the problem” thus it needs to be re-evaluated regularly (and the wise risk manager has a mitigation plan written and in place, if not activated.)

Over the years I have only seen a few select conditions where companies have successfully (so far) dealt with high-risk threat pairs on an “accept it” basis. “Self insuring” is an example of this approach but the truth is it is a huge gamble that appears to pay off until it doesn’t.

“Accept it” as a strategy is the least desirable of all risk management approaches. In IT I can’t think of any risk above the lowest of Vulnerability-Threat pairs in the lowest tier level that shouldn’t have some strategy in place to manage and monitor it. Speaking of which:

Risk Monitored

Needless to say, but I had better anyway, any Vulnerability-Threat pair, however dealt with, needs to be monitored. I know this is a horrifically obvious statement but it is so obvious that if I don’t bring it up it could go un-noticed and thus un-acted upon.

- Regular reviews of risks and likelihoods and re-evaluations of vulnerabilities need to take place.
- Regular reviews of previous risk analyses need to be done.
- Regular reviews of Mitigation, Avoidance, Transfer and Accept plans need to occur (for instance, were the risk management plans ever started let alone completed?).

Risk Management is not a “one shot” deal. Not if you wish to stay in business. That is no more, or less, true today for having the Internet and terrorism than it was fifty or even one hundred years ago. Threats and some vulnerabilities may have been different but they still existed.

Risk Retired

Believe it or not, there does come a time where a Vulnerability-Threat pair can be retired. Could be a significant change in technology. Could be a considerable change in economic climate. Could be that a new process and procedure has been hammered home so thoroughly there is no chance the pair will raise its ugly head again. Could be that a threat source ceases to exist altogether.

Though it makes sense to be cautious when retiring a vulnerability-threat pair (or putting it on a back-burner), don't be afraid to do so when it is appropriate. There is enough to worry about without dragging old “expired” Vulnerability-Threat pairs around. Failure to do so is one of the causes behind the phenomena of “never ending” risk management (and was, in part, a trigger behind my realizing the RMLC existed).

Why Know and Use the IT Life Cycles?:

Even if you are a “one man band” who is “doing it all” by yourself, the above four life cycles need to be an integral, understood, part of your daily IT life ... if you wish to succeed.

The raw unfiltered truth is that no matter the size of your IT organization these four life cycles are in play.

The diamond hard fact is that even if you are a dedicated user of “agile” type methodologies that eschew, that shun, anything that doesn't appear to be immediately and absolutely necessary to what you are doing in that instant on that project ... these four life cycles are in play.

You may not be cognizant of it, you may not realize that these Life Cycles affect what you do and how you do it but they are so fundamental to IT that your success is defined by and dependent on them.

That you may have succeeded in the past without knowledge of one or more of these four LC's should not lead you to believe, should not escort you down the path towards the incorrect conclusion that they were not and are not active catalysts. Whether you realized it or not where you succeeded your success was painted with at least the broad strokes of the Four Fundamental LC's of IT. Of course accidental accomplishment is not a sure fired, reproducible path to continued success.



Summary:

When you put all four Life Cycles together and treat them holistically (sorry, I know it's an abused cliché word, much like "synergy", but it fits), if you have recognized these four LC's as intertwined interacting fundamentals then what you have likely also recognized is they form a framework around which you will be able to build a successful IT organization and with which you can routinely deliver:

"...products and services that are viable, desirable and sustainable; products and services that allow your customers to do something better and provide *them* an ROI on *their* investment too."

Of course you not only have to read about IT's fundamentals you have to understand them, teach them, apply them and reinforce them ... with your passion for IT as the glue that binds.

Hope this helps.

DP Harshman

PDF Link

