Lots of IT Stuff – Where's the (Software) Engineering?

This month's newsletter covers software engineering with some new developments and some common knowledge that is not common practice. Make sure you read the "Go with your Gut" article and try to get January's issue of the Harvard Business Review. Also, we have added a new section called "Tips of the Month" and this month's tips are for the Road Warriors and occasional business travelers.

Annual January Run at Getting Fit – Discipline Lasts About 4 Weeks!

Jim Collins said it best in "Good to Great"
Start with disciplined people.

I typically make three visits per week to the gym for resistance training interspersed between 5-mile runs on alternating days. Although most people focus on either endurance or resistance training, almost all research indicates that the combination of the two provides the best balance of fitness and health. Over the last 20 years of gym work, I am amazed at how many people show up at the gym and are doing most exercises substantially wrong. At best, they will produce little results and in the worst case could do serious damage to their body. Compounding this problem is the fact that it is difficult to find trainers that know any better. With no formal training most of these people see little results and quit within 8 weeks.

Unfortunately, the software engineering business has the same problem. Few come into enterprises with **software engineering backgrounds** and even fewer are doing any ongoing training. Unlike the new exercisers, the newly challenged software people don't quit, they stay on the payroll for years not cracking a book or completing a course. So what makes up software engineering anyways and does anyone know what it is?

Does Anyone Really Know What Software Engineering (SE) is Anyways?

The challenge for most companies is not the technology. Rather it's putting in place the skilled people with talent and processes that coordinate all the disjointed disciplines that typically are involved in getting applications into production. Unfortunately, computer science knowledge typically constitutes an important but narrow subset of SE, typically focused in the design and construction areas. When it comes to getting the applications into production, many people who are trained in computer sciences do not want to get involved in other knowledge areas outside of the traditional computer sciences. The accepted knowledge areas as defined in the software engineering body of knowledge (a.k.a., SWEOK) proposed by IEEE are:

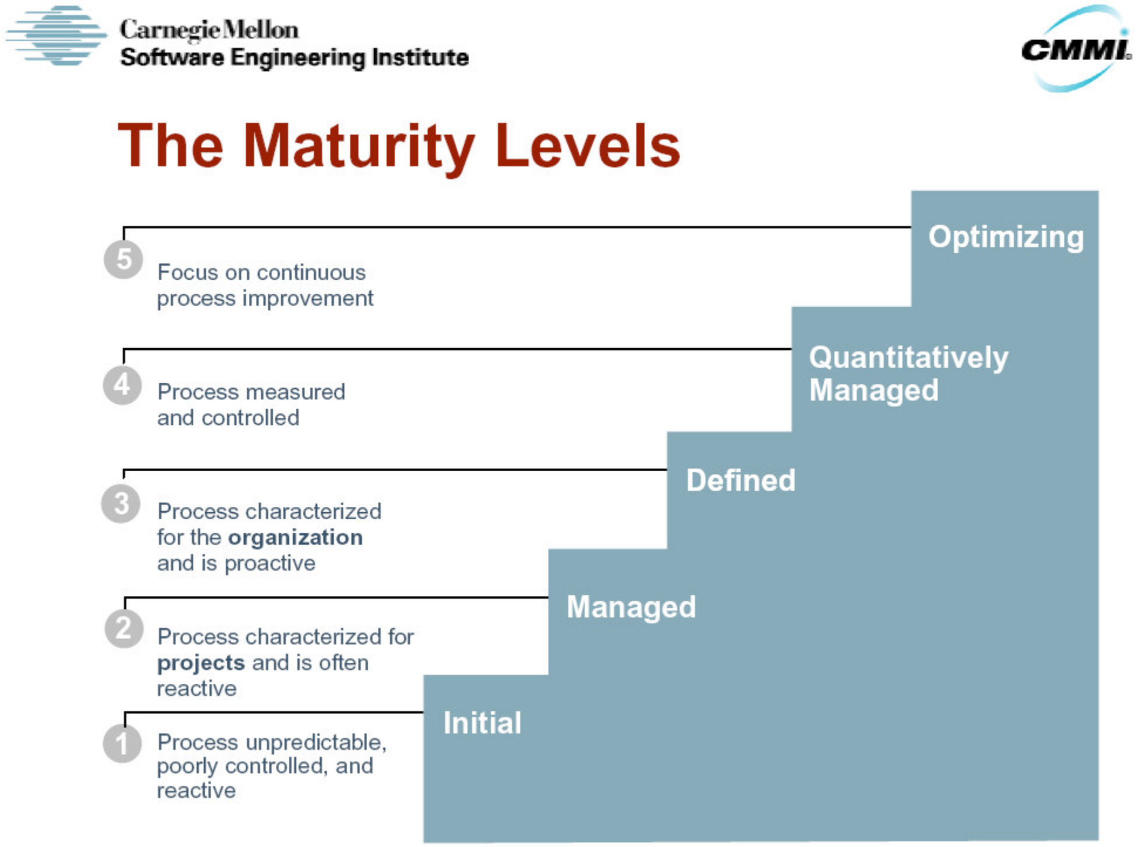
1. **Software Requirements Engineering –** The creation of artifacts that define the needs of a real world problem. This may include product, processes, functional and non functional needs.
2. **Software Design –** Defining the systems and components that will meet the requirements.
3. **Software Construction –** The creation of executable artifacts through appropriate languages, coding and unit testing.
4. **Software Testing –** The verification of software for correctness according to requirements.
5. **Software Maintenance** – Revisions and additions to existing software applications.
6. **Software Configuration Management –** The ability to identify the software's configuration at points in time with the goal of maintaining its integrity and traceability.
7. **Software Quality Engineering –** The task of defining what the quality process should include given the cost of quality.
8. **Software Engineering Management** – Includes the portfolio's, programs and projects required in delivering software into production.
9. **Software Engineering Infrastructure** – Tools for design, tracking bugs and releasing and organizing code for reuse, to name a few.
10. **Software Engineering Process** – Rational Unified Process is one approach.

That is a long list and not one that is easily mastered by an organization. Adding additional risk is the fact that most projects do not take into consideration the softer side of usability, job design and on a grander scale the entire business' set of processes. Even if the software meets the requirements through high quality and low defects, many organizations forget to include optimizing business processes and job design - How should the software be used in the current job. Is the job design correct? On a larger scale, what business processes are involved? Are the business processes right? If the processes are wrong or ineffective, then the software created is not likely to get used by the people in the business units.

What is a Good Gauge of Fitness Level for Software Engineering?

I have seen many companies over the last few years sending their people off to CMMI (a.k.a., Capacity Maturity Model Integration) training and showing preferential treatment to CMMI Level 5 certified vendors – mostly offshore companies. CMMI provides a good document driven framework to complete software projects. It also provides guidelines for rating the maturity of your software engineering processes. Because it is a document driven approach to software engineering, it works very well for products and applications where the markets do not change very much. Unfortunately, many companies that are selecting vendors based on having high levels of CMMI certification are in markets that move very fast or in the last few years have begun to change more rapidly. Slow moving markets include businesses such as Nuclear Reactors (e.g., GE) and the Space program (e.g., NASA). Fast moving markets include the financial industry (e.g., securities and banking). Typically, these companies suit a model driven approach.

Even though CMMI does not fit all situations, it has a good guide for estimating SE maturity in an organization. The SE maturity is split into 5 levels as shown in the diagram below.


Carnegie Mellon
Software Engineering Institute
CMMI
The Maturity Levels
5
Focus on continuous process improvement
Optimizing
4
Process measured and controlled
Quantitatively Managed
3
Process characterized for the organization and is proactive
Defined
2
Process characterized for projects and is often reactive
Managed
1
Process unpredictable, poorly controlled, and reactive
Initial

Microsoft is known to be at level 3. Given the criteria for each level, it makes sense for a market driven software company to be at level 3 or perhaps 4. Most enterprises need to have most of level 3 well implemented but are usually struggling at level 2.

Do a Reality Check, What is Your Company's SE Fitness Level?

Be honest, does your organization have these processes substantially under control. Listening to your gut is not a bad idea (see the article "Go with your Gut" below).

2. **Managed**
    a. Requirements Management
    b. Project Management
    c. Project Monitoring and Control
    d. Supplier Agreement Management
    e. Process/Product Quality Assurance
    f. Configuration Management
3. **Defined**
    a. Portfolio Management – Organization Environment for Integration
    b. Programs – Integrated Project Management
    c. Organizational wide Process Focus and Definition
    d. Organizational wide Training Program
    e. Risk Management
    f. Integrated Teams
    g. Integrated Supplier Management
    h. Verification
    i. Validation
    j. Requirements Development
    k. Technical Solution
    l. Product Integration
    m. Decision Analysis and Resolution

Level 2 is a reasonably sized list of areas that a company can tackle. Unfortunately, I can't say the same for level 3. In prior versions, specifically the older CMM, there were about 7 areas to focus on. Now there are 14. This is far too many areas for one level. Most companies will never be out of this level and it provides less incentive to be certified at level 3. You will end up with many companies at level 3. Also, I put the 14 areas in an order of importance for an enterprise.

SE Best Practices Scooped from Your Competitors Will Likely Not Work Well at Your Company.

Toyota is arguably the best car manufacturer in the world. They have been giving away their production methodologies to competitors for years. They even give plant tours and lectures to other car companies. Still, competitors fail at implementing these BEST PRACTICES even when they have systematic plans. Why so? There are two major reasons. One, best practices in one corporate architecture will likely not work with another – they work best when the organizations have strong similarities. When I say corporate architecture, I mean the culture, organizational and economic structure of the company. Two, companies often look for a quick fix to a major problem or outsource the problem entirely. This casual form of best practices implementation is the most hazardous because the logic of why it works for a top performer and whether it will work in another is not investigated. Take for example one bank, let's call it the BigBank.

BigBank tried to inject UML modeling into their design process by purchasing a banking "best practices" framework for the financial industry- "code in a box" as management thought. UML modeling had worked well at many other institutions and this was seen as a means to improve the bank's software processes. Unfortunately, the effort failed but only after $30 or $40 million dollars was expended. What BigBank overlooked was the fact that the other top performers that used UML software design had hired those with an engineering mindset and gave them extensive training. Unfortunately, this was not true in either the full time staff or any of the 100's of personnel BigBank had hired from consulting companies.

Go with Your Gut, But Include Good Evidence in Making Decisions!

There is a battle going on between your gut and the rational centers of your body according to papers in the January 2006 issue of Harvard Business Review. There are two excellent articles on decision-making – one based on neuroscience and the other centered on evidence based decision making in medicine. Together these articles confirm what many managers suspected but were not completely certain of how to interpret, that they should trust their gut *instincts*.

In Morse's article "Decision and Desires" based on neuroscience findings, he coalesces the latest research that indicates that ignoring your gut (instinct) goes against 1000's of years of biological fine-tuning that has helped you survive to this point. The author also demolishes the notion that emotions should be kept out of decision-making. **Extensive research has proven that humans have a hard time making decisions if emotion is kept out of the decision making process.** So much for rational only thinking. Pfeffer's article on "Evidenced based Management" indicates that managers should make decisions based on good research findings versus biased information such as vendor marketing. A major risk for practitioners is neglecting wider based evidence in lieu of their clinical experience. Some major problems for managers, Pfeffer says, is relying on practitioner strength, marketing hype, outdated beliefs and casual benchmarking. Unfortunately, it is difficult getting good evidence because often there is too much of it, it does not quite apply or stories and testimonies takes the place of sound results based on research and independent investigations.

Managers should be asking, **"show me the evidence"** of why he/she should be making a decision when confronted with information. If they do that, many people will think a lot harder when making recommendations. To become evidence based, avoid becoming captious about the work of subordinates; see the organization as an unfinished prototype by running pilot projects and small experiments.

Be sure to get a copy of the January 2006 issue of HBR.

Rational Software Architect – Continuing on Where Rational Rose Leaves Off.

As you might already know, IBM is no longer issuing licenses (officially) for Rational Rose Enterprise Edition. Rose Enterprise was a bugging half-alive, half-dead product that really had no improvements since Rose 98 – that was 6 years of gravy train for Rational and frustration for designers.

In January 2005, IBM launched the next generation of software engineering tool called **Rational Software Architect** (a.k.a., RSA) with slimmed down versions for developers/implementers (Rational Application Developer) and modelers (Rational Software Modeler)**.** Here are our observations, the good, the bad and the ugly.

The Good

1. **RSA is more like flying a jumbo jet**. The problem with jumbo jets is that not just anyone should be flying one. If you've got one "ringer" on your team give it to him. Chances are that at most companies the ringers are all off fighting fires.

2. **Day-to-day developer and Architect activities work fine**. As of July 2005, a RSA proof of concepts run by Toronix, demonstrated a port to RSA/RAD of a customer application worked well. RSA 6.0.1.1 (Update Dec. 8, 2005) with the typical cycle of design, code, compile and test on the test server, worked well.

3. **The code audit tool allows for automated static reviews of the code but is slow and does not scale well.** In the olden (Golden) days of the 80's and 90's there was time for manual code reviews. But in the internet age, the more we can automate, the sooner we get the project completed with increased productivity. The audit tool is a great built-in feature that can be used to set project standards or to evaluate quickly an applications conformance to best practices. It may also point to where coding errors might be occurring. It is no substitute for dynamic testing but rather works in conjunction with it.

4. **Light learning curve for WSAD and eclipse users who knows UML.** RSA is eclipse based and follows the look and feel of Rational Application Developer (RAD) and its predecessor WSAD. There is also a built-in structure to organizing design and code artifacts that lightens the organizational burden of Architects.

The Bad

1. **RSA is still a resource hog**. Compiles and builds are still slow but that may also be true of RAD 6. Compiling and running audits at the same time is impossible as well as doing any real work until the tasks are completed.
2. **RSA still has very weak reverse engineering capacity.** TogetherSoft's ControlCenter now owned by Borland had class and sequence diagram reverse engineering capacity in 2001. This functionality is still not available in RSA.

The Ugly

1. **RSA has a hefty price at $11,000 US.** Larger customers may get discounts. The steep price makes it tough for projects with limited budgeted to engineer their applications.
2. **The code audit feature swamps the system it works on.** The code audit feature is not scalable. At one customer site, the audit ran for 35 hours before completion. It was a large workflow application. However, it is our view that developers should be able to adjust the CPU assigned to the code review thread. It also uses lots of memory often over 1 Gbyte.

Recommendations

1. RSA should be run on machines with a minimum of 1.5 Gbytes perhaps 2 Gbyte.
2. Consider the new dual CPU Centrino Laptops or Workstations.

Road Warriors, Business and Family Travelers

Here are some travel tips in order of greatest time and money savers:

1. **Forget getting boarding passes at the airport get them on-line up to 24 hours before departure.** I see people using their time poorly by standing in line to see an agent or at the automated terminals. For travel in North America, with most airlines you can check-in on-line and choose your seat. Continental Airline is currently the best at this from my experience.

2. **Get a CANPASS Air card for US/Canadian travel that takes 30 seconds or less to get through immigration.** This is a BIG time saver when there are long line-ups on Mondays and Fridays when travel is at its peak. The program works for both Canadian and US citizens. Not all ports of entry are active yet in both directions but will be by mid 2006. You can use it for business or leisure travel. Although you do need to have your passport and CANPASS card with you, the system authenticates you solely by an eye scan. The downside is that the

logistics can take up to 60 days before you receive the pass. You also have to consent to a background check and go to the airport to get your eye scanned.

3. **Forget the Blackberry and WiFi go with the new Broadband PCMCIA cards for real wireless high-speed internet access.** Ok, you might be able to get both if the company will spring for it. The Broadband cards are available at all carriers – Verizon, Sprint and Cingular in the US and Rogers and Bell Mobility in Canada. The down side is reception may be spotty on the outskirts of major metropolitan areas.

4. **Car Rental GPS navigation will save you time and money.** For $10/day you can rent a Hertz navigation system as a car rental option. I have been using it across North America and it is very accurate. It will save you time in getting to airports, customer's site and possibly from driving accidentally into high crime areas. The downside is tall buildings may block reception so you will lose signal and the car's position will not be updated.