

Response to the Fact-Tree Case Study What is wrong with the Indian Software Industry?

This is an article I wrote in response to the Fact-Tree Case study published by Dr. Vasudeva Varma. The Fact-Tree case study examined the factors that enabled the growth of the Indian Software Industry in the last decade. One of the major findings of the case study is that the vehicle for the phenomenal growth of the Indian Software Industry is the investments into the process methodologies like SEI-CMM certifications and other process quality methodologies.

In this article, I argue very strongly some of the weakness that are inherent in the process methodology as the only way to grow, and try and expose the problems inherent in such an approach.

This article is written with lot of angst and passion. At some time, I plan to re-write this paper in a more dispassionate manner.

Nagaraju Pappu

pnr@canopusconsulting.com

<http://www.canopusconsulting.com>

<http://canopusarchives.blogspot.com>

What is wrong with Indian Software Industry?

Nagaraju Pappu

It is a very well written paper and a very readable one. The technique that is used to writing the paper is very nice - you used a story telling format, and all the people who are mentioned come across as actors on a stage. This is a very good technique, which made the paper very interesting. Another important point that comes across very clearly is their commitment to a way of building their organization. It is very important to stick to a core value and not be defocused - in all the phases in the company, they encountered problems which were new for them, but they kept coming back to how to solve those problems using their philosophy - which is a process centric organization, and they found ways of reapplying that again and again in all phases of their organization. This point came across very well in the paper. Whether we agree with their philosophy or not (that is a process centric organization is the right thing to do or not), still it is a valuable lesson to all young entrepreneurs that having a core value or a philosophy is a must to succeed. This is an important lesson in this paper - and in this sense, the case study is very valuable.

There is an inherent weakness in conducting interviews for such studies, because the developers are not as articulate as managers. For example the developer's voice that the process is a hindrance to their creativity is only an indication of a much bigger issue. If the authors of this paper have tried to investigate it further and more deeply, it would have provided a very valuable perspective to the Indian Services industry as a whole, because this is a common problem across this industry. Now, I represent the voice of a programmer and there are many others like me (for example Eric Raymond, the Agile Programming community) - who can articulate the 'developer' concern as well as any senior manager in these companies. In the rest of my comments, I will try and provide this perspective - it is a perspective addressed to the main actors in your paper - the managers of the FactTree Global Solutions. I will try and argue why their current philosophy is not scalable to their next growth plans (in a way I will try and solve the case myself).

In the paper there is a strong sense that the process centric philosophy is not welcomed by the 'developers' easily - they need to undergo a lot of training even to accept its merits on the basis of larger organizational interests and goals. There seem to be a great divide - the people of the organization and the people in the organization who have to bear the brunt of the process on themselves, who feel stifled by these processes and do not understand the value of this in the context of their jobs. This is an important issue that has to be studied in further details. How can something that is the core of the organization growth become a sore point to the engineer who does the actual work? The indication by the seniors in the organization that the programmers do not understand the value is an explanation that is not convincing. Later in this document I will provide an example of another company which had a similar growth, but where the 'process' is designed to help the programmers - but that was done very differently. The company is Oracle, and I will describe their growth later, but first some other points.

In various discussions by the seniors and how they got the programmers and others accept the process does in fact come across as indoctrination and the process centric organization seem to create an environment of conditioning. The people who give in to this conditioning survive in this company, and the people who don't will have to leave this company early in their careers. This company will have to do a study to find out whether this hypothesis is indeed true.

There are some other interesting and important questions that we have to raise: What is the growth path for a programmer in this company? Is there a senior programmer in this company in the same rank as Irfan, Pallav and others? Or, the general growth path is to become a "manager" in the end? How can a high technology company not have senior programmers in their ranks? Is there a management cadre and a technical cadre in the company? Most Indian Services companies are today facing this problem. What generally happens is that a programmer joins this

industry in one of the companies - but, he doesn't get to work under the guidance of accomplished programmers - they learn by trial and error, most of the focus is on process, because the seniors also do not understand programming in its true sense, therefore - a programmer doesn't really get to understand his craft in detail, the process experts are interested mainly in documentation, and not in problem solving, and quality of work output is really very poor (you mentioned this in your paper) - in a couple of years time, this untrained programmer claims experience, and becomes a module lead, tech lead, project manager, account manager and so on.

They all "hate" programming work in their hearts, simply because they never learned it themselves. They did not learn it simply because there was no one to teach them this craft. This is a major problem confronting the Indian IT industry in general today - we don't have world class programmers, and we don't know how to get them. In general, people in this country, especially the people who have engineering degrees are intelligent and hardworking, but there is a question of training and teaching. We don't have educational institutions which can take a student and make him into a world class programmer. Our teachers themselves are not programmers themselves. If you look at universities in US, there exist many teachers and academicians who built systems themselves - from Dijkstra (THE operating system, Algol compilers) to Douglas Schimt (ACE distributed computing framework and TAO orb). The amount of open sources - be it databases, operating systems, compilers, various tools etc is amazing in US. In fact, most successful companies start their products from open sources.

Let's accept it, converting a problem or need into a set of programs is a highly involved and specialized craft - a program is not simply a set of sequences of execution. The body of knowledge of developing SYSTEMS is sadly lacking in India, plus the knowledge of the craft. I can give you some examples that you can relate yourself. Take the case of InfoDream and the tale of two programmers - Kamlesh Vyas and Rahul Datir. Both came from similar back grounds, both were simple, easy going people. Both were hardworking people. Kamlesh had the advantage of a mentor which Rahul lacked. In the end this reflected in the system they built. Rahul's applications suffered from severe performance problems because no one told him how to build server class applications. He used applets and tested them in a single user environment with a global locking mechanism to make them work in a multi user mode. In the end, the company paid a very heavy price for this. Kamlesh on the other hand went from strength to strength because he had access to two mentors who provided them with both the conceptual understanding, problem solving and the important domain understanding (language processing).

Another example from IIT-K days: TVP never did any programming review with me, he used to only give a specification and some high level conceptual ideas - but he had no knowledge of the actual nuts and bolts of programming. I had to learn this on my own - and it took a lot of time, trial and error. I had to rewrite the application several times myself to better organize the system into separate concerns. Same goes for you - your supervisors never offered any guidance in term of how to build systems - and the end result today is that Anusaraka is a highly unusable system, millions of rupees were spent on this project - but, in my view it failed because they did not understand how to build a system out of their theories.

Let's come back to the main theme of this article - Fact Tree and their process orientation. I would describe Fact Tree from the perspective of a programmer. I can guarantee that in Fact-Tree, there are no source control systems that act as a single repository of all the projects they did in the last 15 years. They wouldn't have good build tools that they developed themselves and so many other technology management practices would definitely be missing. For example, do they have a framework of performance verification tools? I suspect most of their projects fail on "non-functional" requirements. I can also guarantee that a day in the life of an employee in this company is very stressful - they are fighting fires all the time, they are under constant pressure to deliver on time, their long meetings tend to be 'beating around the bush', they have to constantly justify the delays to their clients - and who ever can do a good job of this gets all the

promotions, and in general, when it comes to the actual 'technical' work - they tend to be clueless.

The programmers have learnt how to 'scare' their managers and seniors, quoting all kinds of jargon (this module is multi-threaded system and it is very difficult and time taking to debug it, in J2EE, session beans cannot do x or y etc..). This leads the managers to inherently 'mistrust' the programmers - and they in turn resort to pressure tactics to get their work done. They think J2EE, .NET etc are solutions instead of as engineering tools. They twist their lack of programming knowledge around and say that it is better to "buy" and assemble solutions rather than build them. Their idea of moving up the value chain means that they should focus more on the core business problems and not on "technology issues". There would be pockets of technical excellence - a handful of "techies". However, these people are perceived to be very arrogant and difficult to work with, but a necessary irritation that must be tolerated. They think that since "process" helped them to solve difficult problems of growth in the past, it must be useful to solve the current set of problems as well, and that they only have to find a way of re-inventing this again. If this is indeed true, the Fact-Tree managers should read the rest of the article....

What are really the growth issues that are facing Fact-Tree and other similar large services organizations in India today? The first problem is that the days of 100% growth are over. Today these companies grow in the ranges of 10-20% (all of them fudge their numbers to report such growth rates, but the real rate of growth is much less), and in a few more years these growth rates will come down to single digits. This is a reality today. In this context, how will these companies keep their spectacular profit margins intact? If the salaries and running costs grow at the same rate as today, then keeping up with the current margins itself may prove to be a miracle. The second problem that is confronting the industry is the fall in the dollar rupee exchange rate - which is expected to fall further to their realistic levels in this decade as the Indian economy picks up, and the debt oriented dollar continues to lose its value further.

The third problem that is confronting the industry is peculiar to Indian Industry alone - so far, the advantage India was primarily one of cheap labor - it may continue to be so for some time, but one cannot grow only on this basis alone. The story told to me by a senior account manager working for a very large services company is an eye opener in this regard. He said that their clients are now complaining about the efficiency of the Indian companies - their point simply is that the Indian companies charge very low hourly rates, but they take extraordinarily long hours to get the job done. Let's say that the hourly rate in US is \$100, and a US programmer takes 10 hours to complete the job. But, the Indian company charges \$20 an hour, but takes 200 hours to do the same job. Now, bring the east European companies - who are technically very strong, and charge half the rates that are prevalent in US - the Indian companies are up against a formidable competition. An east European company does not need to grow to the size of the Fact Tree - they can do a much better job with one tenth of the number. So, the number game will not work in another few years. Add to this the cheap labor competition from other countries like China - the survival of Indian companies is a big question in the long run.

We have understood offshore model very well, better than any body else - but this is no longer a knowledge that can be considered intellectual property. These are the economic and growth related macro factors that are effecting the growth and survival of these companies.

There are other problems as well - the kind of work we have to do today is mission critical, co-development of systems and services. This requires an intimate knowledge of computing science, programming and very core technical knowledge - these days christened as "architecture". Largely today's services organizations' capabilities can be summed up as : offshore delivery, process knowledge, account management, and a large pool of "skilled" technical people. There is no other intellectual property other than this - in some cases, they may claim domain expertise, but this is questionable and debatable intellectual property in any case.

If you visit the websites of most of these companies, they advertise their development methodologies and quality practices as their core strength. A methodology in itself is not a product or a solution. Even if we accept that these companies were able to fine tune these methodologies to such a degree that they can be called “products” - these methodologies are a variations on the twenty year old water-fall methodology. By definition these methodologies are designed for large scale, big turn-key implementations. Today we are asked to complete a project from conception to deployment in under 4-5 months of time, with an average team size of less than 10. The Water Fall methodologies fall flat under such circumstances. The weakness of such methodologies are already well documented and argued by people like Tom DeMarco, Fred Brooks and Gerald Weinberg (People’s ware, mythical man month and psychology of a computer programmer) and many other people, therefore I do not need to go into the details of this issue.

However, I would like to make another important point with respect to the quality of software systems and how these methodologies approach that topic. How does a typical project plan look like that is done using these methodologies - there is some time for requirements and functional specification, time for high level design (these days this is called architecture unfortunately), low level design, implementation and then testing. What gets tested is implementation - that is code. How do you uncover the design problems in code? Any way, how do you know at the time of planning, how many “bugs” will be found, and the nature of such bugs? And, how do you at the time planning know how much time and how many resources are required to “fix” the bugs? The answer is there is no way you will know this apriori. If that is the case, then how can you confidently say that testing and bug fixing takes 10% of the development effort? You have to consider the possibility that you may have to completely rewrite the entire application. So, strictly speaking, you can give an accurate project plan only after you completed testing - correct? This explains all the reasons for the delays in projects, quality problems in the application and the general stress levels in the software industry. This is a problem that companies like Fact-Tree must address. They sing the six-sigma song as a cure to these problems, but six-sigma is not yet completely understood in the context of software. In summary, the following issues must be resolved:

- The “process-centric” investment got us effectiveness, but the current challenge is efficiency - does the same method solve this diametrically opposite problem?
- Can Fact-Tree grow to 100,000 people in the next few years and maintain the same bottom line? Will the current structures apply in the massively large scale company?
- What should be direction of this company - should it grow in numbers? Or should it grow its intellectual property so that it can do 10 times more business, in half the time with the same work force? There really is no option
- Indian companies do not make investments in genuine R&D. Microsoft spends close to 6-8% of its revenue in R&D (which is a huge amount of money). Same is the case of many successful technology companies. How come Indian companies do not have a genuine investment in R&D? If such an investment exists - what should they be focusing on? Is it possible to develop some intellectual property on which services can be offered - especially for horizontal companies like Fact-Tree?
- If the mantra of next decade is fixed fee projects as opposed to Time and Material type projects, what type of methodologies are suited and how do you adopt them?
- How do you bring a culture of genuine expertise and provide a growth path for technical people?

I shall try and answer these above questions in some detail. First we must address the “process centric” thinking and find out the inherent loopholes in this thinking. In your paper you quoted some analogies and rationale given by Irfan - the process and quality guru at Fact-Tree. These analogies and arguments are very clever - no junior programmer can rebut these arguments. I refer you to Eric Raymond’s Cathedral and the Bazaar series articles. There is one question I would like to ask Irfan - how does he explain the phenomenal success of Linux? Linux is one of the massive undertakings ever in the history of the IT-Industry, it is a spectacular success - managed by a group of open sources programmers all over the world. They have no “process” or “standards” -- in the conventional sense of these terms. But, there are many open sources projects - like Linux, GNU Emacs etc., which have a life time of more than a decade, participation of thousands of programmers from all over the world - who do not take any payment, and delivered reliability, quality and efficiency that are very elusive in normal projects. I quote Eric Raymond from his Cathedral and Bazaar :

“Traditionally-minded software-development managers often object that the casualness with which project groups form and change and dissolve in the open-source world negates a significant part of the apparent advantage of numbers that the open-source community has over any single closed-source developer. They would observe that in software development it is really sustained effort over time and the degree to which customers can expect continuing investment in the product that matters, not just how many people have thrown a bone in the pot and left it to simmer.

There is something to this argument, to be sure; in fact, I have developed the idea that expected future service value is the key to the economics of software production in the essay [*The Magic Cauldron*](#).

But this argument also has a major hidden problem; its implicit assumption that open-source development cannot deliver such sustained effort. In fact, there have been open-source projects that maintained a coherent direction and an effective maintainer community over quite long periods of time without the kinds of incentive structures or institutional controls that conventional management finds essential. The development of the GNU Emacs editor is an extreme and instructive example; it has absorbed the efforts of hundreds of contributors over 15 years into a unified architectural vision, despite high turnover and the fact that only one person (its author) has been continuously active during all that time. No closed-source editor has ever matched this longevity record.

This suggests a reason for questioning the advantages of conventionally-managed software development that is independent of the rest of the arguments over cathedral vs. bazaar mode. If it’s possible for GNU Emacs to express a consistent architectural vision over 15 years, or for an operating system like Linux to do the same over 8 years of rapidly changing hardware and platform technology; and if (as is indeed the case) there have been many well-architected open-source projects of more than 5 years duration -- then we are entitled to wonder what, if anything, the tremendous overhead of conventionally-managed development is actually buying us.

Whatever it is certainly doesn't include reliable execution by deadline, or on budget, or to all features of the specification; it's a rare 'managed' project that meets even one of these goals, let alone all three. It also does not appear to be able to adapt to changes in technology and economic context during the project lifetime, either; the open-source community has proven *far* more effective on that score (as one can readily verify, for example, by comparing the 30-year history of the Internet with the short half-lives of proprietary networking technologies—or the cost of the 16-bit to 32-bit transition in Microsoft Windows with the nearly effortless upward migration of Linux during the same period, not only along the Intel line of development but to more than a dozen other hardware platforms, including the 64-bit Alpha as well). “

Though the above article is written in the context of Open Sources development, the questions that it raises are very valid. Eric Raymond goes on to investigate what the “processes” really are meant for and their value. He makes a very interesting observation - that the process is the product. If you have four groups working on a compiler, you get a four pass compiler. The process more or less defines the end product. It is not uncommon in Fact-Tree like companies that their projects have to follow their organization structures - requirements, functional specifications, architecture, design, coding and testing, documentation, quality adherence and so on. The project tends to have these distinct phases, because their process demands it. Their organization structure also reflects their process.

To answer these questions, I would like to introduce another term - discipline. The Fact-Tree type of processes may give effectiveness, but they do not give efficiency. Efficiency comes from discipline. To understand this term more closely, let's examine the growth of another company (Oracle) - which had the similar humble beginnings like Fact-Tree and became one of the most powerful technology companies the world has ever known.

Oracle was started by a group of four young techies - they made the first implementation of an RDBMS engine, from the published specification of the relational database theory. The first implementation was done in FORTRAN. Like Fact-Tree, they had some early breakthroughs, and they realized the importance of listening to the customer's voice very early in their business. The business grew - it was a right product at the right time - rapidly, and they had some major customers like Boeing Corporation etc. The company grew beyond expectations in about first 8-10 years. Their first real “product” like implementation was Oracle version 3.0 - after that they never looked back.

In early 90's the company became really big (more than 2000 employees I think) - and they posted their first loss. They quickly realized that they are operating like a startup and there is an urgent need to put in place financial controls, strong accounting principles, and a well integrated marketing, sales, support and development divisions. They realized that they are now supporting the product on multiple different platforms and the number of platforms and customers are growing in leaps and bounds.

This is when they had to restructure the company - they divided the company into multiple units - the base development, base porting and product line groups were the basic structure of the development organization, and customer support and marketing, consulting are the other core divisions. The base development is concerned with developing the product in a robust fashion (the kernel, the data storage structures like btrees, indexing, table spaces, the sql-language parsers, the various database access tools, protocol support like TCP/IP and other protocol support development. The layout of the base development is in terms of how “technically” you relate to a RDBMS engine. They had some supporting libraries like - memory management libraries, parsers, data access libraries etc., which are common to all the applications). They used to develop each application on a different platform - because of the application's closeness to a platform. For example, the kernel was first developed on Sequent multi-processing machines, the tools were developed on SunOS etc.

The job of the base porting was to take all the implementations of various different applications and port it to a base platform - which is SunOS. The reason for SunOS was the majority of platforms supported by Oracle were all variants of Unix, and SunOS was a very good reference implementation of Unix operating system. By the time the “code” came to the base porting group - it was re-organized differently. Their view of the system was in terms of platforms - not in terms of the “technical” layering of the product. So, they reorganized the product into a different structure - kernel, tools, supporting libraries. The supporting libraries were all brought to be on SunOS - and the entire product was divided into platform independent code and platform dependent code. From this point onwards, only the platform dependent code needs to be modified or implemented by different platform groups. So, the base porting group would

make a release to the product line groups - and the job of the product line groups is to “port” the product to their platform, develop any tools, applications, installation programs, platform specific documentation etc. which is relevant to their platform and release the product to the support groups.

The job of platform groups is also to support all the customers of that particular platform - but they never supposed to modify the base code, because this would effect all platforms. This way the company exploited the fact that very little amount of code is platform specific. Another advantage of this approach is that from one version of product to another version of the product - a major portion of the code can be retained, for example - the database access methods rarely change. Once this is tested and works, this portion can be used in all versions. In other words, the organization of the product was done according to the rate of change of the different components. The less frequently changing components were at the bottom of the pile and the more frequently changing ones are at the top of the pile. Using this approach, the company was able to cut down the porting cycle to just 40 calendar days. A porting cycle means that from the time the base porting kit was released to the time the product was shipped to the customers.

They also organized the product documentation along the same lines - only installation and platform specific documentation was modified by the porting groups. The platform independent documentation was also organized in terms of its longevity - Oracle Concepts rarely changed, features of the kernel would change more often and so on.

In this process, the company realized the importance of certain programming standards - because the same code will have to run on massively parallel machines as well as low end desktops like MSDOS. So, they constituted a team of “stake holders” from base development, base porting, porting groups, testing groups, support groups - and formed a committee and evolved Oracle C Coding Standards (OCCS). All the code had to be made conformed to this standard - but the question is how? They developed automated tools (called Olint) which could check the code against OCCS - and only the code that is passed by Olint was allowed to be checked into the source control - the source control would reject code that did not have OLINT signature.

Very soon the company realized that the central source control has to be different from source control systems that each project would use. So, they bought the source from a big source control company, modified it according to their requirement and constituted a separate group that would maintain the source control for the entire company - this is because, end of the day, the source is their bread and butter. Each group is free to use their own source control systems internally - but when it comes to “releasing the product” - they had to contact the central source control department and obtain the appropriate release symbols and source paths. As mentioned earlier - this made the job of structuring the code according to the requirements made easy. The base development group can maintain their own way of source tree internally, but they had to check it into the appropriate tree structure in the central source control.

They also had to solve another important problem - their code is their IP. But, there are thousands of programmers who work in the porting groups - how could they protect their code from being easily understood by every one? Another problem is as the company becomes big - it becomes difficult to identify bright and talented people - and move them up. They solved both these problems in one shot - they understood that if they restructure the code slightly differently and make it difficult to figure out, then the bright people would figure it out any way, and they would approach the right people in the company to understand how it is done. However, Oracle did not stop here - as the company matured, the product offering changed, they made necessary changes to the organization hierarchy and organization structures as well. Once Java was available - they implemented a “virtual machine” and made it part of the kernel and other tools - so that they don’t have to write platform specific code in all releases. If the

virtual machine is available on all platforms, then rest of the applications just need to link with the VM.

Plus by this time, the markets and the environment had changed drastically - internet brought distributed computing, application servers and so on. At this time, the company reorganized itself along the product offering - instead of base development, base porting and product line groups - they restructure the product into Kernel Groups, environment groups and tools groups.

This technology management was the key to Oracle's success - many other database companies came and died - like Sybase, Informix etc., but Oracle always went from strength to strength.

The key lies in their dynamic organization structures that are closely tied to their product structures. Eric's comments that process is product can be understood from Oracle's example. In fact, in successful companies the product is the process. If you look at how Linux development is organized, you will find a very similar story. The word "process" is never used in these companies, they use the word discipline - and discipline is enforced and is always there as an undercurrent.

Another interesting fact about Oracle is that the so called processes were designed to relieve the burden of the programmers, they never increase their burden. Oppose this with the Processes of Fact-Tree - and you get a very revealing story. Another interesting fact is that the "processes" were less management oriented but they are more to do with technology management. Compare this with the knowledge sharing exercises, mentoring processes of Fact-Tree.

Oracle also formulated two different kinds of career paths - every one joins as MTS (member technical staff) - from here onwards, depending on their inclination - they would move either in technical lines - SMTS, Principal member, staff member, architect and fellow of Oracle. The other line is project manger, development managers, group managers, directors, vice presidents and senior vice presidents. The fellow of oracle has the same or in some cases more authority than a senior vice president. Oracle was able to retain their more experienced programmers and technical people in this way, and this enable them to provide an environment of training to their young technical staff - by working in an apprentice mode, they learn a lot. But no one ever got the impression of being stifled by organizational processes, quality standards etc.

There are some interesting parallels in Oracle and Linux development - if you follow how Linux started and matured - you will notice that both had a very similar discipline and practices - in fact, the story of Oracle and Linux match point to point.

I would now attempt to generalize these observations to all software development - be it open-sources, product engineering companies or services organizations. The list is as follows:

- All successful technology companies have strong engineering teams coming from strong engineering backgrounds. You will find very few MBAs in Oracle, very few MBAs in Linux and very few MBAs in successful technology companies.
- Most successful software development projects, products never use waterfall methodologies. These methodologies are the biggest crime ever perpetrated on the software industry - they are too simplistic, there is too much overhead and they come from a wrong analogy. Software is manipulating ideas, and therefore you can never use an assembly line or manufacturing methodology and apply it to software development. It is a series of problem solving iterations - and therefore you will find that most successful software development groups use release early and release often philosophy - they don't use long release cycles.

What is wrong with the Indian Software Industry?

- Even in Fact-Tree, if you conduct a survey of really successful projects (Where the clients were delighted with the quality, time and efficiency of the solution) you will find that the teams had very good programmers in them, they approached the problem from a technical perspective, they had fun working on the problems, they had time to re-write the application many times, they used very short release cycles, they followed very short requirements specification cycles and so on. You might even find that the team is mostly made up of reclusive, introvert people - who probably cannot speak very good English in any case. You may find that they used test driven development - they developed lot of test drivers internally, used very short, but precise documentation. And finally, you will definitely find that they used quality engineering rather than quality standards. You will also find that they never tried to hide any facts from their clients - if they make a mistake, they accept it openly, they share all the issues with the client openly and make the client a part of the project and success. They would have hated the Fact-Tree quality processes, but they managed most probably because they had a very strong leader backing them up all the way.
- Neither in Linux, nor in Oracle or in any successful software projects - one never heard of performance problems, reliability problems etc. All these successful projects accomplish one nearly miraculous feat for the waterfall people - that is the quality of their solution/product is always constant with respect to the non-functional requirements, and the only “bugs” they encounter are functional ones. This is the dream of the Waterfall people - which they can never realize.
- All such projects have strong technical people (engineers) as project managers who had roll up the sleeve attitude. They see the development as a series of problem solving exercise rather than as a process movement exercise.

Now, let us come back to the next growth of Fact-Tree and how they should reorganize themselves. Following are my key recommendations:

- They should remove the word ‘Coding’ from their vocabulary and use and respect the word programming instead.
- They should have a career growth path for technical people.
- They should realize that their processes are only a hygiene factor - and their use is limited to that much. They must also realize that their waterfall variation processes will always end up as too much overheads with too little real value. In the last twenty odd years, there is tremendous progress on the software engineering front - there are newer disciplines like agile techniques - which are an answer to the call of short cycle development. Fortunately, these techniques are developed by accomplished programmers themselves - so they work very well. However, agile techniques cannot be adopted in the current organizational structures in Fact-Tree, they have to re-orient themselves.
- They must realize what Irfan was championing can easily be accomplished by automation and internal tool driven approach. It is a matter of putting in place some good technology management practices. Fact-Tree must invest in internal automation and enforcing the discipline in a natural way rather than pushing it down the engineer’s throats. If such an approach is adopted, Fact-Tree may find pleasantly that it won’t need half of its managers.
- They must listen to the voice of programmers - instead of ignoring it constantly. It is valuable, and they have to address it honestly. A process that is viewed as an additional burden by the programmers is most probably that - a burden.

What is wrong with the Indian Software Industry?

- Their current “processes” and education is nothing but a set of beliefs - and belief is not a substitute for knowledge. If you have not built a system yourself, you have not programmed yourself, if you do not have the necessary technical knowledge - then you cannot design processes.
- They must invest in open sources - a way to build strong technical teams is to have them participate in open sources projects - be it Mozilla, Linux, Emacs, Elvin, Apache - what ever - there must be teams that participate in these projects and actively contribute to them. This way they learn how to program to international standards, they learn good programming and design techniques, and their programs get reviewed by world class programmers. By contributing to the open sources movement in an active fashion, Fact-Tree will benefit in the long run.
- People like Irfan and Anu are redundant in the current organization. They added a tremendous value in the early phases of the company. They should now move on to BPO companies - where they can add value. They are not in touch with technology and programming any more - and therefore, they have no role to play in the company any more.
- Fact-Tree must institute a new policy and vision that they will complete their work day in less than 8 hours. Any project manager who makes extraordinary time demands on his programmers must be penalized. This way every team will be forced to innovate, forced to build tools and infrastructure that brings in efficiency. Most process driven software companies in India work over time, all their employees are stressed out. What use is such a process that cannot ease the pain and bring value to day to day work life?
- They should move towards fixed-fee model as quickly as possible. This will force them to be more efficient in what they do and how they deliver. Re-usability must have a business goal - otherwise, it will never succeed. They must encourage internal competition - project groups must be able to compete for a project by submitting alternate proposals. Since Fact-Tree has lot of people in its roll - at least 50% of the projects should be done by two different teams, and best implementation should win.

If Fact-Tree institutes such measures, it may succeed in the next decade. Otherwise, its stock is not worth buying.