

Silicon Stories

Carlos A. Leyva

Recovered from the archived 2010 Silicon Stories website

Contents

Introduction

Dirty Little Secret

- The Early Years
- Big Oil Company
- The Emperors Are Naked
- Big Oil Is Cool
- The Rexx Underground
- The One-Eyed Man Is King
- The Lawyer's Desktop
- Big Natural Gas Company
- Tulip Mania
- Buynothing.Com
- A Freudian Perspective
- The Usual Suspects

Competitive Advantage

- A Moral Dilemma
- Fear and Greed
- Michael (Porter) Is Still 'Da Man
- Simply Profound
- The Second Law of Thermodynamics
- Decisions in Progress
- Interesting Asides
- Commercial Software vs. In House Development
- Understanding Complexity
- Workflows and E-Business
- The Next Big Thing?

Process Patterns

- A Set of Heuristics
- Call to Arms Pattern
- Producer Pattern
- Bad Actor Pattern
- Key Abstractions Pattern
- Collaborator Pattern
- Planning the Plan Pattern
- Wandering Around the Desert Pattern
- Super Star Pattern
- Blame Game Pattern
- Summary

Movie Making & Software Development

- The Process
- The Story

The Missing Factory

Assembly Line Tools and Processes
Source Code Management (SCM)
Incident Management
Tools of the Trade

Architecture

So What?
Shack, House or Skyscraper?
Commercial Plumbing
A Set of Services
Market Drivers
Microsoft Worldview
ABM Worldview
Components
The Only Architecture That Matters
XYZ Applications

Talent Wars, E-Teams and E-Cultures

The Mission
Natural Leaders
Screwing the Pooch - Whose Team is this Anyway?
The Any Good Manager Fallacy
For the Love of the Game
The Economic Unit of Value Creation
Don't forget the "e"
The Environment
Us Versus Them
The Furniture Police
Talent Connoisseurs
Preventing Burnout
Human Resources as a Competitive Advantage

Knowledge Management & Infotainment

Telling Stories
Reward the Teachers
Small is Beautiful
Information and Entertainment
Screwing the Pooch Revisited
Interactivity
Experiencing the Brand
Context-Sensitive Rich Content
Big Automation Company
War Games
Smart Appliances
eLearning
Individual Responsibility
Emerging Technologies and Research
Telling Stories Revisited

Epilogue

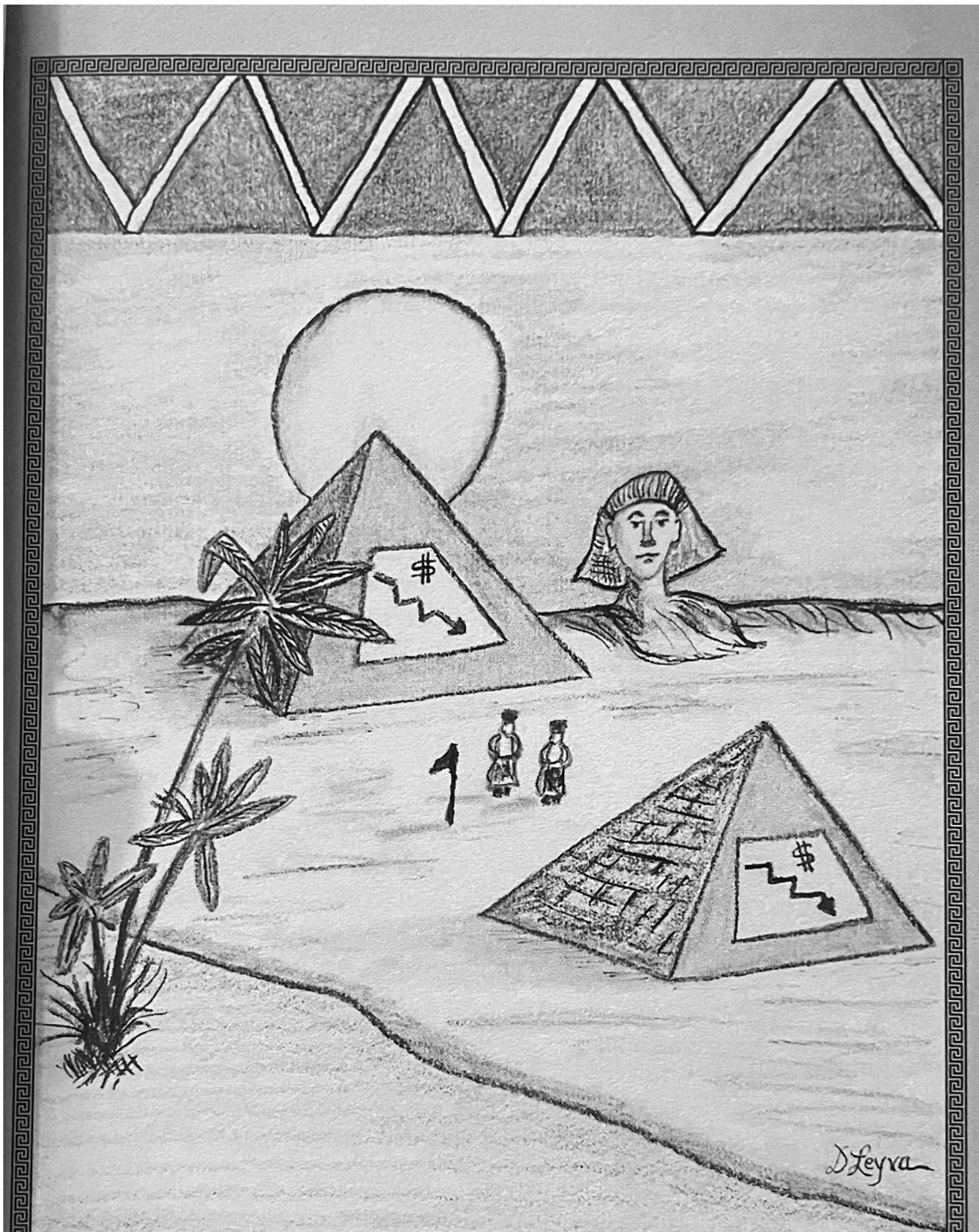
The Soft Stuff is the Hard Stuff

Prisons or Schools

The Good Old Days

About the Author

About the Illustrator



D. Leyra

Introduction

Silicon Stories is a serious book about business and technology, but it should not be taken too seriously. Many of us in the technology industry have the self-destructive tendency to let the sound of our own wheels drive us crazy. The irreverent flights-of-fancy and the tongue-in-cheek style of Silicon Stories are intended as a governor, reducing the often mind-numbing blandness of techno-speak.

Plato wrote, "There is no such thing as teaching, only remembering. We have learned everything there is to learn, the more we seek to learn, the more we must realize that seeking and learning is all remembrance" (Meno, 79C-81A). Many authors appear to be converging on similar memories at the same time.

In that sense, Silicon Stories is not groundbreaking. However, it is often the angle at which light shines upon a thing that allows us to see something new in it. There are numerous angles in this book that hopefully contribute significantly to our collective memories.

The first three chapters are chock full of information and set the stage for the remainder of the text, therefore they should be read in their natural order. Subsequent chapters were written to stand-alone and can be perused in whatever order the reader's interest dictates. For you techno-humor aficionados, please be sure to read chapter 5, "The Missing Factory".

If this book makes you laugh, that is goodness. If it makes you laugh at the author, that is even better. I have had a damn good time laughing at myself. You may even have an epiphany or two. Enjoy.

Chapter 1

Dirty Little Secret

This chapter sets the stage for the remainder of the book. It spells out, in very simple terms, why I believe that we are not nearly as successful in our software endeavors as we could be. I will refrain from quoting failed projects data that is too often quoted. If you are interested, you can visit the Standish Group's website at standishgroup.com. These are the folks who track the numbers.

I will also refrain, for the most part, from focusing entirely on dot-com failures, although some stories related to this era are relevant and personal. If you really want the low down on dot-com failures, you should read *ComputingFailure.com* by Robert Glass. Also, if you have a seriously morbid interest in following the demise of dot-com and other contemporary companies, then read *F'd Companies: Spectacular Dot-Com Flameouts* by Philip Kaplan.

The fact that we are not as successful as we could be in our software endeavors, which now includes almost all business endeavors, is not widely in dispute. There is a tremendous amount of literature on the subject, most of which focuses on technological aspects of the problem. Although some chapters in this text do touch on technological issues, I will attempt to provide causal analysis by focusing on historical, organizational, and sociological aspects of the software dilemma.

I have been a student of business and computing (in that order) during my entire professional career. I have deep intellectual curiosity in both domains; however I am, by far, more practitioner than academic. For the most part, this book is a collection of my war stories and experiences as reported from the front lines.

Information is free. Insights are rare.

— C.K. Prahalad

Hopefully, the stories and anecdotes in this book will provide insight that will prove useful to business executives, IT executives, and other practitioners. If that happens, I will certainly be pleased. However, I must confess that this is not the sole, or even the primary, reason for my telling of these anecdotes. They are in my head, and I have a need to tell them. It is, in part, an attempt to exorcise some of my own demons. I am very passionate about computing. It has totally consumed a large part of my waking hours for many years now—mostly, but not always, in a positive way.

Chapter 1: Dirty Little Secret

The Early Years

Before blurting out the “Dirty Little Secret,” I need to set the stage and appropriate context. I graduated with honors in 1984 from the University of New Orleans. (Not bad for a kid from the “wrong side of the tracks.”) I obtained a degree in Business with a Computer Science option. Prior to that, I participated in a family-owned business, which afforded me early software and management experience.

Upon graduation, I interviewed with the Big Oil Company on campus in May 1984, and was asked why they should have an interest in me. My response was that I had some software experience, and that I was very, very good at what I did. I also added that my academic track record supported this fact.

Apparently, they liked what they heard and I was asked to fly to Houston for additional interviews. I interviewed with three managers in Houston. One manager, in particular, asked me why I thought I could compete? I was somewhat offended by the question and shot back: “What makes you think that I couldn’t?” Needless to say, I was not only confident; I was cocky. (In time, aging and the realization of our own mortality have a way of making us more humble).

Why the confidence? Because I knew I was a developer. I had an aptitude for it. I had a need to do it. Corny as it may sound, I felt a calling. I had done work in BASIC, studied COBOL, FORTRAN, Assembly Language, and Pascal. I had breezed through tough programming classes (data structures, operating systems, analysis of algorithms, etc.) and loved every minute of it. I was armed and dangerous. I was ready to do some very cool stuff!

Chapter 1: Dirty Little Secret

Big Oil Company

I started work at Big Oil Company in June 1984. I had my suspicions about whether the work there would be cool, but they offered me what seemed like (at the time) more money than I ever thought I would make. Cool be damned! I took the job! Before my first real assignment at Big Oil Company, I attended an eight week on-the-job-training program in COBOL I & II, JCL, TSO, and Mark IV. Mind numbingly, boring stuff, but I was getting paid to learn, and that was cool.

My first assignment was supporting procurement systems. I was more than a little disappointed to find myself in support. Didn't they know I was a developer? However, this assignment turned out to be a great experience for me, and it provides some of my early anecdotes.

Developers as Second Class Citizens

It didn't take me very long to figure out that developers (support or otherwise) were, at best, second-class citizens at Big Oil Company. The real talent was headed for the management ranks. This technology stuff was something you just had to get through along the way. It was kind of like surviving boot camp. Don't get me wrong, I had nothing against moving up, getting a bigger office, making more money, wearing nicer clothes, etc. (Business casual was still many years off.) The only problem I had was that I liked the technology stuff.

I knew that there was a fork looming down the road, and like Yogi Berra, I would take it. The fact that people were only tangentially rewarded for technical skills was not that big a deal, from a business perspective, when you lived in a more or less technically stable world where IBM always provided the answers to your questions. IBM would even answer questions that you hadn't thought of yet.

Needless to say, the technology world did not remain stable for very long. IBM did not have all the answers. Everyone had a need to get smarter, especially the folks that were responsible for the technology. However, reward systems and organizational cultures utterly failed to recognize this fact, and still have not caught up more than 15 years later.

Chapter 1: Dirty Little Secret

The Emperors Are Naked

The Dirty Little Secret is that most IT organizations' executives (CIO, directors, managers) have fallen woefully behind the technology curve. They are the decision makers and caretakers of the technology, yet they have become incapable of providing the leadership required. As such, their organizations are unable to leverage the enabling technologies that are threatening to transform business itself.

This fact is the direct result of two important factors: (1) organizational reward systems and cultures, and (2) the absolute and relentless explosion of new technologies. The following is a short list of the technologies that have had an important impact on the corporate computing universe:

The PC was the shot that was heard around the world. It was the first of many destabilizing technologies to hit the corporate beachhead in the last 20 years. This relentless attack continues, with no end in sight. At best, savvy IT executives have struggled mightily to keep up with the buzzwords. But attending seminars and reading the rags is no substitute for leadership.

Why is this a Dirty Little Secret? Because, it is widely known, yet never discussed! Professionals that have cut their teeth on newer technologies recognize that there is something dreadfully wrong as soon as they hit the job market. It doesn't take them long to realize that the generals, their lieutenants, and most of the sergeants are all ill equipped for battle. They know enough to be dangerous to themselves, their subordinates, and their organizations. That's it.

Now that I have dropped the bomb, I would like to digress somewhat and share a few more anecdotes. After six months at Big Oil Company, I was part of a team that had rolled out a new purchasing system to one of its refineries. I spent quite a bit of time on-site, helping the end-user community and the developers back in Houston fight through the inevitable beta site implementation bugs. I made a concerted effort to build strong relationships with both teams, and was instrumental in facilitating the dialog and reducing organizational friction.

Chapter 1: Dirty Little Secret

Big Oil Is Cool

Several things changed the un-coolness of Big Oil Company for me on the technology front. The first was the realization that processes, data and databases are not only important, but also way cool. Systems (technical aspects only) are composed of code, processes and data. Of the three, processes and data are more important than code. They represent the real world objects that we are attempting to model. They have a symmetry and a technological elegance all their own.

Relational technology was coming into its own, and Big Oil Company was an early adopter. I was still a geek at heart. I still loved to code. However, the importance of processes, data and databases loomed large. And they would continue to do so after the emergence of object-oriented technology took the industry by storm. Many developers who cut their teeth on PC's completely missed this part of their education.

The second thing that was very cool was my discovery of the VM/CMS operating system and an IBM scripting language called Rexx. VM/CMS was IBM's mainframe equivalent of Unix. It was almost entirely command-line-based, and rumor has it that it grew out of the IBM geek community's personal computing needs.

The purchasing system that I supported ran on VM/CMS based IBM 4381's onsite, at the refineries. It had a need to communicate with the central MVS-based inventory systems that ran in Houston. It was therefore a distributed system, possessing many of the needs and complexities inherent in today's distributed systems.

One of the early unfilled needs was a convenient way to download production code (enhancements and bug-fixes) to remote machines. I filled this void by writing a source code control and distribution system entirely in Rexx. This was no small feat, since it involved asynchronous messaging and an event-driven programming model. The combination of Rexx and VM/CMS provided the necessary language and architectural support for the job.

My love affair with Rexx and VM/CMS rekindled a passion for coding, and it allowed me to network with the emerging rogue community of Rexx hackers that were popping up in nooks and crannies of Big Oil Company. My team would later leverage this architecture to develop a completely integrated and distributed EDI system that linked Big Oil Company's purchasing operations with its strategic vendors nationwide.

About this time (1987), the quality revolution had already taken America by storm. Phillip Crosby was Big Oil Company's favorite author (Quality is Free , Quality Without Tears). His books were required reading and distributed to the entire organization as part of an internal quality program. Every employee was a member of at least one quality team. Each team was required to come up with a project that would save Big Oil Company money. The team I was on selected software reusability as our project, primarily because I pushed for it and volunteered to do most of the work.

Chapter 1: Dirty Little Secret

The Rexx Underground

I was plugged into the Rexx underground, and I was keenly aware that some of my colleagues were doing fantastic work that could, and should, be leveraged by the entire organization. In particular, there was a developer on the west coast who was doing amazing things with Rexx (screen building tools, search tools, etc.). A few of us were already leveraging this stuff, but apart from the insiders, the rest of Big Oil Company was oblivious. I put some credible numbers together, got the support of a senior executive, and presented my case in front of the IT executive committee.

I must have been waxing poetic about the virtues of reusability and happened to use the phrase “in this business” (meaning the software business). I then began speaking of reusability as the Holy Grail (or something to that effect). Before I could finish making my point, however, one of the executives interrupted me and yanked my chain. He said, “What business might that be?” I hesitated for a New York nanosecond and responded with, “the oil business.” That was the politically correct answer. What I should have said was, “Listen moron, you have 2000 people in IT, and whether you know or not, you are in the fucking software business.”

Not long after my “quality” presentation, I landed a position on the hottest IT project at Big Oil Company. I had developed a reputation as a technologist with good business skills. I was also teaching “C” at Houston Community College during this time. The powers that be assigned me to the architecture and framework team, which had been conceptualized and in early design for about a year prior to my assignment. The project was not being called a “project.” It was called a “program.”

The program’s charter was to develop, from scratch, a purchasing and financial system for the entire Exploration and Production organization. By the time I transferred, the program team was comprised of three layers of IT management with about fifty people in all. There were also a number of folks from the business side that had been assigned full-time to this effort. The program King Fish was a high flier from one of the business units. The kicker was that the proposed system was to be completely PC-based. Big Oil Company was a Big Blue Shop, and OS/2 was poised to take over the PC universe. This was really exciting stuff back in 1987.

Chapter 1: Dirty Little Secret

The One-Eyed Man Is King

I must confess that during the year I spent on that project, I thought most of the IT decision makers were both stupid and clueless. Some of my colleagues and peers were not far behind them. What seemed obvious to me never even appeared on their radar screens. I did not have all the answers, but I could sure as hell tell you, with some degree of certainty, what was not going to work. With the passing of years, I have come to a far different conclusion. Just as the mountain is often better appreciated the farther you are from it, I realized that, while most of these folks were certainly clueless, they were far from stupid.

These were some of Big Oil Company's best and brightest. They had graduated from some of the most prestigious schools in the country. They were ambitious and very interested in their careers. What made them clueless was neither a lack of intellect or a desire to learn. They were attempting to rationalize a problem for which they were entirely ill prepared to conceptualize, let alone solve. The complexity of the problem and the technologies that were evolving (and morphing weekly) left them at sea on a rudderless ship.

Information Sickness

They were experiencing a form of information sickness. Information sickness is related to information overload, but its symptoms are more intense and long lasting. An article in Style Magazine by R.L. Rutsky entitled "Techno-Cultural Interaction and the Fear of Information" provides interesting insights into both. What follows is a quote from this article:

As these analyses of postmodern or contemporary culture suggest, too much information is often figured as a kind of loss: as the denaturing of a larger meaning of context, the breakdown of wholeness and cohesion, the failure of understanding or comprehension. The categorical and narrative structures by which we understand and interact with the world are being transformed by massive amounts of information, leaving us with a sense of being overwhelmed, disoriented, and lost.

I would later become well acquainted with information sickness in an up-close and personal way. This business is brutal, and it is anything but glamorous in the trenches. So, it is time to have an honest conversation.

Chapter 1: Dirty Little Secret

The Lawyer's Desktop

The idea was to provide an integrated package at a price that was suitable for the small to medium sized law firm. It would provide time and billing, case management, and calendaring modules. There were individual products on the market that met some of these needs, but not an integrated solution. We both had significant experience working with data models, so we were confident that we could reverse engineer a coherent model based on existing products. At least on this score, we were quite successful. With respect to almost everything else, it became painfully clear that our ambitions far exceeded our ability to execute the project.

I made a significant number of strategic business and technology blunders that ultimately provided me with a world-class education, but at a very steep price. I will share the story with you, even though, a decade later, I still cringe at the emotional and economic toll it took on me, and on my family.

Failure and Wisdom

Our achievements speak for themselves. What we have to keep track of are our failures, discouragements, and doubts. We tend to forget the past difficulties, the many false starts, and the painful groping. We see our past achievements as the end result of a clean forward thrust, and our present difficulties as signs of decline and decay.

—Eric Hoffer

I certainly have learned far more from my failures than from whatever limited success I have attained. The lessons learned from bringing The Lawyer's DeskTop to market were many. One of the most important lessons I learned, all too well, was the reality of risk. You cannot begin to understand risk until you put your own money on the table. The risk is commensurate with how bad it will hurt you to lose it.

Risk is that knot in your stomach that entrepreneurs learn to live with, but never get accustomed to. When all was said and done, I estimated that this education cost me about \$150K USD. While this might not seem like a big number by dot-com standards, when it comes out of your own wallet, it hurts (bad). Obviously, all business ventures include a certain amount of risk, but software ventures are riskier than most. I will use my personal experience as an example for the kinds of things that can go wrong.

When I started The Lawyer's DeskTop, I thought I had all the tools that I would ever need:

Chapter 1: Dirty Little Secret

Big Natural Gas Company

Big Oil Company could lose an estimated \$100 Million USD on a project and have it go virtually unnoticed outside of the organization. Big Natural Gas Company's fate would be quite different. It was the early 1990's, and client-server technology was all the rage. PowerSoft had developed PowerBuilder, a product that enabled the IT masses to finally break into the MS Windows development universe. Prior to PowerBuilder, only the initiated priesthood was capable of making sense of the Windows 'C' based API hieroglyphics. The combination of PowerBuilder on a Window's 3.1 client, and Sybase running on a Unix server, created the client-server platform for mainstream IT.

The Young Turks

The young Turks in senior IT management positions at Big Natural Gas Company were quick to seize the opportunity. They managed to convince the CEO that competitive advantage could be secured by becoming an early-adopter of the client-server craze. They had plenty of help from a local emerging PowerBuilder-inspired consulting company. It was, to say the least, an ambitious plan. All of Big Natural Gas Company's strategic enterprise systems (pipeline scheduling, nominations, and gas accounting, to name a few) were to be converted to the new platform. This new platform would allow the company to integrate its customers into its operations in an unprecedented way. (Sound familiar?)

Anyone familiar with pipeline systems will attest to their complexity. They support a large customer base. They maintain complex contractual relationships between the pipeline and its customers. They must process a significant number of transactions that occur on a daily basis, and within rigorous time constraints. The mathematical complexities of systems such as pipeline scheduling are not like those of your run-of-the-mill back-office systems. It is safe to assume that hundreds of man-years had been spent developing the MVS mainframe based systems. The young Turks did not take a toe-in-the-water approach. Knowingly or not, they had convinced the business executives to bet the company.

In many respects, PowerBuilder was an ugly (and in its early incarnations, barely usable) product. It was slow (interpreted) and had a fairly large footprint. What made PowerBuilder a compelling technology was an internal object called the Data Window. The Data Window allowed PowerBuilder clients to interact with SQL databases in powerful ways. It provided both developers and sophisticated end-users an elegant window to their data. The combination of SQL and PowerBuilder provided a viable (if somewhat slow) platform for departmental client-server computing.

The problem was that Big Natural Gas Company was not porting departmental systems to the new client-server platform. They were porting their strategic enterprise systems. These mainframe-based systems supported hundreds of users and a large number of transactions. The combination of MVS, COBOL and DB2 (although not cool by today's standards) provided a solid

architectural foundation that was field-proven.

Not Ready For Prime Time Player

The fact that client-server was an immature platform for these kinds of systems is an understatement. If there were voices of caution that dared to speak, they were drowned out by the certainty of the young Turks, and by corporate America's love affair with all things new. What follows is the story of an industrial giant's demise.

I was hired to work on the C++ based Unix backend. PowerBuilder was used for all the client-side GUI work, and C++ was used for the heavy lifting on the server. The lion's share of the staff (upwards of 120 employees and contractors at the height of development) worked on the client, even though the lion's share of complexity existed on the server. In addition to the nominations and scheduling systems, the server hosted a number of infrastructure related systems, such as batch job scheduling.

The Wild West

There was some serious top gun talent hired to work on the server-based systems (Billy the Kid, Wyatt Earp, and Doc Holliday, to name a few). Even though the gunslingers had, more or less, delivered on their end of the bargain, the whole system began to implode after I had been on the job about 6 months. About this time, bad rumors started circulating. Big Natural Gas Company had not been able to accurately account for pipeline transactions for over a year, and customers were having difficulty entering transactions over dial-up lines.

Executive management was becoming impatient with the burn rate and the endless unfulfilled promises. Employees were mandated to work 60-70 hour weeks, and were obviously showing signs of burnout and fatigue. System performance was going from bad to worse. What was the executive team's brilliant response? Spin out a software company to recoup the investment by selling the system to competitors. Hey, if your systems are fucked up, you might as well level the playing field by fucking up the systems of your competitors, right?

A Slow Descent to Hell

Needless to say, the rings of software hell just got deeper and deeper. Every customer that it managed to land sued the ill-fated spin-off. The young Turks escaped to the west coast, where they found work in unrelated industries. Billy the Kid founded a very successful software company. Wyatt Earp disappeared into the technology underworld. And as for me, I am just hoping I can finish writing this book before tuberculosis, or some self-inflicted wound manages to kill me. Big Natural Gas Company got out of Dodge as well; a competitor bought it at a fire sale price (so I hear). The estimated back-of-the-envelope loss related to this endeavor: \$500 million USD (easily).

So what's the point? The point is not to assign blame, although rumor has it that the young Turks are back in town and are up to their old tricks again. Caveat Emptor! The point is that when patterns begin to emerge, there is an opportunity to make sense out of chaos. The chaos becomes

somewhat predictable, paradoxically. There are lessons to be learned and insights to be gained that can improve the probabilities of success.

The rest of this book is an attempt to highlight recurring patterns. Patterns emerge in all naturally occurring complex systems. They provide the keys to a better understanding of their inner workings. The same holds true in the development of complex software systems.

Chapter 1: Dirty Little Secret

Tulip Mania

It would not be fair to close this chapter without a few stories related to the dot-bomb universe. Looking back, we were so stupid. How could so many knowledgeable people have been fooled? What were the social and organizational dynamics that enabled this folly? The bad actors have all come clean with mea culpa confessions. We are so much wiser now. Bullshit!

We were all seduced. We were seduced by our love affair with new technologies. We were seduced by our own abilities to create compelling stories that have no basis in reality, yet capture our imaginations and beg to be believed. Above all, we were seduced by greed and easy money. We got fucked and we deserved it. Our own hands carried us there.

The fact of the matter is that we are experiencing discontinuous change at an increasing rate, and unless we are collectively smart enough to learn real lessons, we will wind up making the same mistakes again. The bad news is that we fucked up. The good news is that as individuals, and as a society, we almost always learn more from failure than from success.

But if you conclude from our failures that new economy thinking and technologies are dead, then you are going to miss the next boat out. Entropy forces us to respond with complex information technologies in order to make sense out of emerging business ecosystems. We have no choice but to embrace chaos and complexity, and to create order (and value) from it.

Speaking of chaos...

It was the summer of 1998, and I had spent the last 3 years running the local professional services organization for a Northwest Computer Company. I was ready to get back into the software game. The Internet was making software exciting again, and the last place I wanted to be was on the sidelines when the revolution happened.

I got a call from an old acquaintance from my Big Oil Company days. He and some partners had started a company a few years earlier. Its function was to perform data rationalization (cleanup) of legacy data. The primary customers were the oil companies who were looking to cleanup their legacy inventory data. It turned out that his company had recently been bought by one of the emerging B2B players, and they were looking for talent that could take their software to the next level.

Chapter 1: Dirty Little Secret

Buynothing.Com

I started working at BuyNothing.com in September of 1998. My first assignment was to enable the existing software with asynchronous messaging functionality. BuyNothing.com was a Microsoft shop, at least within its Houston operations, and so it was decided that MSMQ would be the messaging technology of choice. The idea was that the Houston operation would maintain a centralized database of rationalization rules (pattern matching) and the software that utilized them.

The combination of rules and software would be leased to our customers on a per line item transaction basis. In other words, they would get to use our stuff to cleanup their data, and they would pay us for every line item cleaned. The continuous bi-directional flow of rationalization rules and all the accounting would be handled over the Net.

Technically speaking, the idea sounded more than a little wacky. But hey, when you are the new kid on the block, you don't start making noise right away. MSMQ 1.0's API turned out to be fairly robust, but the message management tools and utilities were almost nonexistent. You could definitely create message queues, send and receive messages, and track them (most of the time). However, when messages got lost, or appeared as though they had not been sent, it was difficult to determine exactly what had gone wrong. This was frustrating, but somewhat expected for a 1.0 product.

I convinced my boss, the acquaintance that had recruited me (who was the V.P. of Technology) that we needed to log all in-bound and out-bound messages so that when things went wrong, we could recover by resending. This was not a stroke of genius on my part. The EDI system that my team built for Big Oil Company was built on top of asynchronous messaging technology, and it was a requirement to have full logging and resend capabilities. Convincing my boss that this was a requirement took some doing. I was surprised just how hard he pushed back. After finally convincing him, the procedure later saved both of our asses on more than a few occasions. I was more than justly vindicated.

The V.P. of Marketing had convinced one of his contacts at GorillaB2B.com that this pattern matching technology was the shit, and they agreed to beta test it. A couple of us spent a week on the west coast getting them up and running. With some trials and tribulations, we managed to get the MSMQ messaging stuff to work using a VPN across the net. The actual "product" itself was a completely different story. It had no user's manual, was buggy, was terribly slow, and was difficult as hell to use. The developer that accompanied me spent a significant amount of time teaching these folks how the Houston operations team used "backed doors" around the software to increase productivity. Needless to say, GorillaB2B.com did not become one of our largest or happiest customers.

Meanwhile, back at the ranch, it seems that I had created a monster. The current incarnation of the software used a Visual Basic DCOM server and an MS Access database to provide multi-user functionality to clients across the network. The problem was that the VB based server was single

threaded. As such, it could not support more than three simultaneous users, let alone the 15-20 that we were staffing up for. I had been on the job less than a week when I started recommending that we convert to SQL Server and “fix” the single threaded VB server problem. However, I was soon diverted to do the MSMQ work, and I decided it was best to fight one battle at a time. After the failed beta at GorillaB2B.com, we had egg on our face, to be sure. But we had also proved that the asynchronous messaging technology was viable (more or less).

Chapter 1: Dirty Little Secret

A Freudian Perspective

Before proceeding, I need to provide some background information on this V.P. of Technology I have been referring to. At Big Oil Company, he had always worked on the business side of the house. He was definitely a closet hacker (part of the REXX underground) and a developer wannabe. He had attained his master's degree in computer science at night, no small feat. And for many years, he had worked on this pattern recognition problem. Finally, he had built a very interesting, but mostly academic, prototype.

He had leveraged this prototype into a significant payday, and I was willing to give the devil his due. However, this was his first real technology job, and it is safe to say that we did not see eye-to-eye. He all but came out and stated that anything that couldn't be done in VB and MS Access was not worth doing (first love syndrome). In other words, this individual had a seriously small techno-dick complex and was extremely insecure about anything that he did not understand. He was willing to protect his ego and myopic intellect at all costs.

It was decided, against my vehement protests, that we could solve our internal scalability problems using MSMQ. Each content operator would be given a single user version of the app and a copy of the database on their local workstations. MSMQ functionality would keep all workstations synchronized via messages (essentially, transactions passed across the network).

Why was this so bad? First of all, as previously mentioned, transactions had a way of disappearing into the network bit bucket. Queues would become corrupt and stop sending messages. The user communities' frustration quotient would soar. We had workarounds for all of these problems, but the "solution" was difficult to support (and that's an understatement). Moreover, there was a relatively straightforward multi-user client-server solution available.

Don't get me wrong; I liked MSMQ. The technology had significant promise. Microsoft worked closely with us to resolve issues when they occurred. The problem was this: when all you have is a hammer, every problem starts to look like a nail.

About the time of the MSMQ wars, I was promoted to Product Manager, a job I had basically been performing anyway. The local development team reported to me. My team eventually implemented the aforementioned multi-user client-server solution and began to add work process management to the existing offering. No more lipstick-on-a-pig enhancements. This pig needed a major facelift!

My boss was busy managing his East Coast development team and had little time for us, which was a fact that everyone appreciated, and none more than I. The fact that he was too busy to fuck with us was a blessing. That he had time to fuck with anything at all was a curse. The East Coast development team, in conjunction with one of the major consulting firms ("the usual suspects"), was responsible for developing the online presence. This included the presentation of electronic catalogs, as well as the efficient and effective searching of the electronic catalogs. Get the picture? It was basically BuyNothing.com's knockoff of Amazon.

Chapter 1: Dirty Little Secret

The Usual Suspects

I have a challenge to issue. Can anyone show me one example of any of the usual suspects producing, on a work-for-hire-basis, an innovative product or platform that has gone on to dominate its category? These folks are business consultants and project managers. They are damn good at what they do. If you want to implement SAP, CRM, Supply Chain Management (name the latest acronym), then give them a call. But if you want help developing next generation (or this generation, for that matter) software, then call someone else. Or else, the likely outcome will be that they walk away with a ton of money, and you are late to market with an average product (if you are lucky).

Software innovation is not what they do! Now let's be fair. The reason that the usual suspects are often brought in is to cover some executive's ass. Let me give you a clue, you cannot cover your ass when developing an innovative offering. It doesn't work that way.

Inmates Running the Asylum

At about this time, BuyNothing.com hired a new CEO. He was some Neo-Nazi motherfucker from a large software manufacturer who was known for having a factory that cranked out idiots of his caliber. I met with him on several occasions. He inspired anything but confidence and trust. The last time we met, he actually threatened to sue me and a colleague from Houston for developing some innovative ideas that we thought might help save his sinking ship.

He was one of 'dem foreign nationals who have the attitude that all Americans are stupid. Hey, if you weren't stupid, you would be CEO by now, right? He happily informed me that he could care less if everyone in Houston quit. I happily informed him that he could kiss my ass. Welcome to the new economy!

It felt good telling a CEO to kiss my ass. This market was hot, and I had game! Not only would I leave, but I would also take my whole damn team with me, if the opportunity presented itself. I guarantee you that they were more loyal to me then they were to him. Treat key employees with disdain at your own risk. Industrial age arrogance will get you killed. Don't believe me? Read the "Talent Wars" chapter for insights into the new balance of power.

EnergySavings.com

I was getting about fifteen job leads a day, and I was happy to see that things hadn't changed all that much during my 3 year hiatus from the business. I ended up running the local office of EnergySavings.com. I asked for (and got) more money than the V.P. who had hired me. This is primarily because I chose to take a significantly reduced options package in order to get cash in hand. (You can't spend worthless paper).

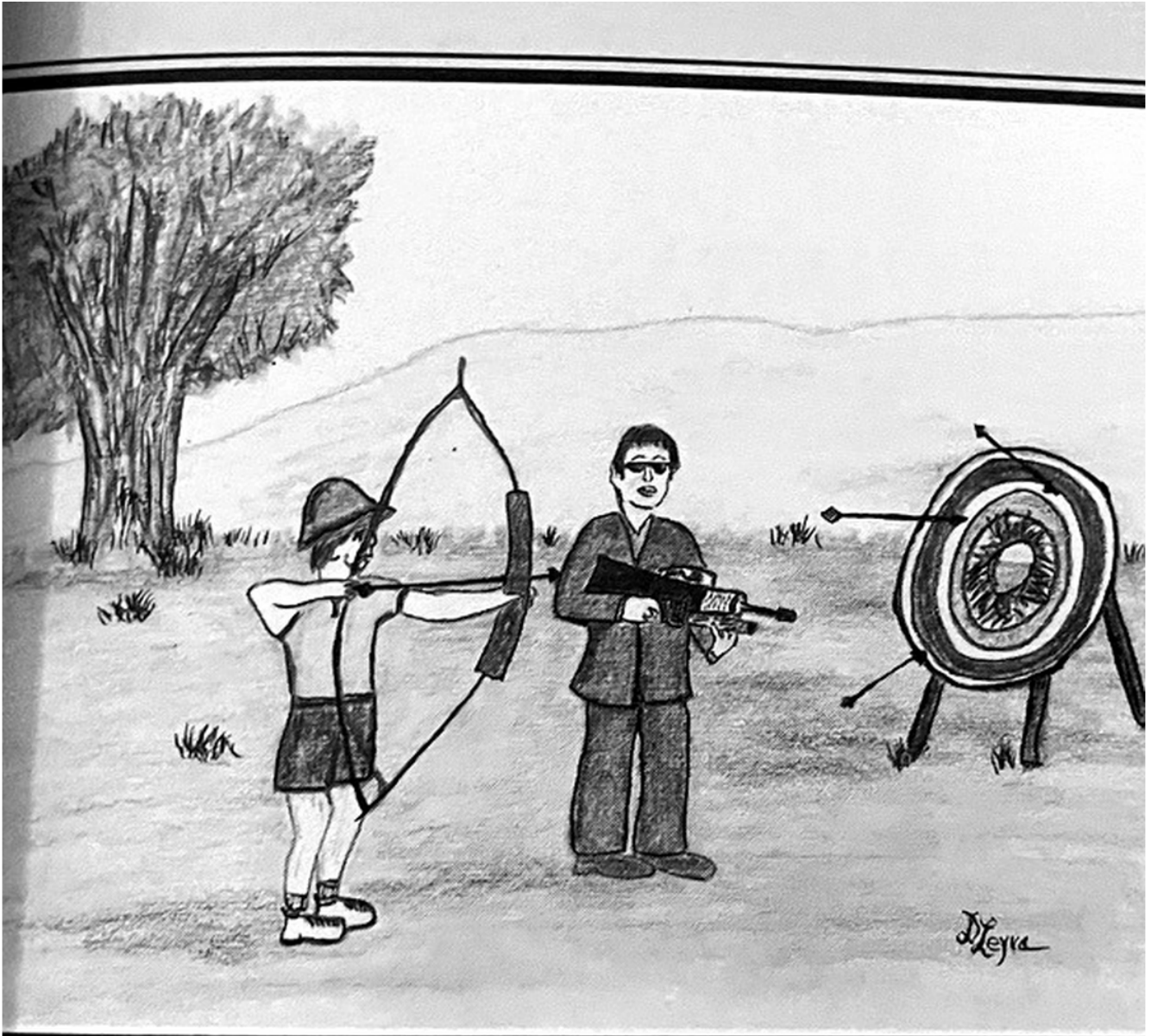
By now, I had no illusions about how dot-com mania was likely to end. If I had suspicions that things were fucked up in the software business before, I now had a smoking gun. It was becoming

clear that smart bricks-and-mortar companies would eventually dominate the Net. You know, companies with real business models and real revenues. But for now, it was time to make hay while the sun was shining, because sooner or later, we were in for a dry spell. I started at EnergySavings.com as a hired-gun architect, and 3 weeks later, they asked me to run the Houston office.

EnergySavings.com was a de-regulated electricity play. They had started in California and were moving aggressively into other markets as individual states began to de-regulate. By the time I came on board, they were already starting to lose their grip on reality. Rumor has it they struck a deal on the east coast—a deal that was costing them \$1 million USD a day. Try making that up on volume!

The Houston office existed due to historical reasons. There was a talent pool here that understood back-office energy systems, and one that had no interest in moving to the valley. The average age of the Houston development team was approximately 25. They had done an incredible job of bootstrapping these systems into a working model. That said, the systems needed some re-architecting in order to scale. That is what I was hired to do.

My team quickly put together the “As Is” versus the “To Be” views of our world. The plan for the “To Be” was approved by the executive team with one caveat: I was to attend a week long course on a new middleware package that EnergySavings.com had purchased, and then figure out how to integrate it into our plans. I had worked extensively in the middleware space in my past lives, so I was anxious to expand my denominator. The middleware company was located in the valley, and I attended (along with staffers from the west coast operation).



COMPETITIVE ADVANTAGE

Chapter 2

Competitive Advantage

There is a time in the affairs of men,
Which, taken at the flood, leads on to fortune.
In such a full sea we are now afloat,
And we must take the current when it serves,
Or lose our ventures.

—Shakespeare

In order to speak coherently about competitive advantage, I must stand on the shoulders of a giant. Michael Porter's definitive work on the subject remains the bible on this topic, so much so that competitive advantage and sustainable competitive advantage are now a permanent part of the world's business lexicon. In this chapter (and those remaining), I hope to provide some insights into broad trends in information technology (i.e. the world according to Leyva). I also hope to show how they might impact a firm's ability to gain and sustain a competitive advantage. Unfortunately, I often find it difficult to speak without a white board, so I will enlist the assistance of a number of my favorite graphics. Bare with me, I promise to do my best to keep you from snoring.

Two Tears in a Bucket

Goddamnit! I don't curse. I just use some words as adjectives.

—Dwight D. Eisenhower

Before moving on to serious matters, I must address an issue that is near and dear to my heart. I sent the first chapter of this book to my inner circle of friends and colleagues. Many responded with enthusiasm (probably out of kindness). The one consistent criticism, or item of commentary, that they all voiced was the political incorrectness, or inappropriateness, of the profanity I use in this text. I asked them if I had made my point, and there was universal agreement that I had. I asked if I had made them laugh. Again they conceded that I had. I asked if it sounded like me. They responded with more laughs and confirmations. Then I told them to fuck off. (Just kidding.)

So, what does this have to do with competitive advantage? Plenty. I believe that, apart from brilliant and clever strategies, competitive advantage can only be sustained, going forward, by having an honest conversation with the marketplace. This point is made most eloquently by a group of individuals who wrote a wonderfully human book called the Cluetrain Manifesto . Check it out at www.cluetrain.com, or better yet, go to Amazon and buy a copy for yourself, your friends, your kids, and for any other human being that you care deeply about. (No shit.) Here's a quote from the book:

We are not seats or eyeballs or end users or consumers.

We are human beings and our reach exceeds your grasp. Deal with it.

Okay, Carlos, let me get this straight. In order to attain competitive advantage, you have to be willing to say Neo-Nazi motherfucker in print? No, not exactly. But you must be willing to have an honest conversation. I wanted to write this book for my friends and colleagues in the technology trenches. I wanted to tell these stories in my own voice. I wanted it to sound like the stories I tell when shooting the shit and drinking a few beers at the local dive. Yes, I also wanted to be taken seriously professionally, but not at the expense of sounding phony.

An individual who hired me in a consulting capacity, as Chief Architect (in charge) at Big Automation Company, recently reminded me of a comment that I had made to him regarding his persistent offers of a permanent gig. I told him that if I took a permanent gig, I would be less inclined to tell him, “Fuck you, asshole. You’re full of shit.” And I have, from time to time, behind closed doors.

The point is that I am not knowingly going to help you do something that I believe to be absurd, or just flat wrong—not for all the tea in China. I am not going to lie to you or otherwise misuse your trust for personal gain. That is part of my value proposition. That is part of what I bring to the table. If you want someone to kiss your ass, then don’t enlist my help, or that of my company. If you are looking for an extremely capable individual, who has (not often enough) the occasional flash of brilliant insight, then I am probably one of the better candidates that you are ever going to find. However, I must warn you, I will probably use the word “fuck” (usually not at inappropriate times, however), as well as all of its wonderful derivatives.



A Moral Dilemma

Chapter 2: Competitive Advantage

A Moral Dilemma

I spent 3 years running the local professional services organization of Northwest Computer Company. I learned more about business (as opposed to technology), during this time, than in all of my previous years combined, or since. I have wonderful memories and enduring friendships from this period in my life. However, life at Northwest Computer Company caused me unmitigated stress and took a serious toll on my health. There were several reasons for this, many, but not all, were of my own making. I will share a particular dilemma that troubled me beyond belief, and which is relevant to this discussion of competitive advantage and its sustainability.

In the early 1990's, Northwest Computer Company was the shit. They had managed to string Intel processors together (using their flavor of Unix) in a scalable architecture that was more compelling than anything else on the market. By the time I arrived on the scene in 1995, Sun, IBM and HP were whacking them upside the head. They responded with a number of clever, but ultimately unsustainable, strategies.

The first strategy was the result of a brilliant insight. The client-server revolution had left a gaping hole, with respect to the dearth of knowledge available in the marketplace regarding the new architectural paradigm. While their competitors were busy pushing iron, Northwest Computer Company radically revised their sales model to lead with architecture. The fact that some of their sales teams, and their nascent professional services organization, could speak to these issues was a significant differentiator that led to big wins. When this strategy hit the wall (i.e. the competition caught on) they discovered Data Warehousing.

Data Warehousing and Big Unix Iron

Data Warehousing was a client-server technology that required big Unix iron. The idea was to lead with professional services (data warehousing design and implementation) and bundle the iron as part of the deal (i.e. change the rules of engagement). The implementation of this strategy required more depth in their Professional Services (PS) organization. I was hired as a project manager in the local Houston office, and I was subsequently promoted to professional services manager. The position was responsible for working with account teams as pre-sales support (i.e. providing domain expertise) and managing resulting PS engagements once a sale was concluded. The strategy was far more successful on the west coast and in the U.K., where the sales teams and the PS organizations got it and had the depth of talent to implement it.

Frankly, the strategy was broken for many reasons. However, the problem was primarily that it required a level of sophistication that was not universally available. Instead of selling a product, the sales teams were mandated to begin selling solutions. Never mind that this required a completely distinct mindset, one that the majority of account teams were totally ill prepared to market.

Many prospects, at least in the conservative oil patch, were suspicious. They quickly surmised that we were leading with PS in order to sell iron. By 1995, the Unix market had already begun a quasi-de-facto standardization process (to the degree that this was possible in the Unix space). Prospects were extremely reluctant to commit to a niche Unix vendor whose market share was decreasing visibly.

This dichotomy in marketing strategy resulted in a moral dilemma for me personally. I believed wholeheartedly in solutions. I believed that well executed solutions were in the best interest of our potential customers. However, our solutions, provided by our capable PS organization, could readily be implemented on our competitors' iron (often for less cost and reduced risk). In other words, I only partly believed in our value proposition, and from the point of view of Northwest Computer Company, not the most important part.

Chapter 2: Competitive Advantage

Fear and Greed

Knowing this in my heart of hearts, after 3 years, I still decided to accept a position as an account executive. Why? Because of ambition, greed, and stupidity (mostly stupidity). Account executives ruled the roost (position power), and rightfully so. Nothing in business happens until a sale is made. If you closed a few big deals, you could be in high cotton.

It was an outside long shot, but I was a born gambler. I was a kid who had grown up in the streets, and I figured that no matter what, I would always land on my feet. Not this time. I had gotten damn good at technical marketing, could wax poetic in front of an informed crowd and be convincing, but marketing ain't sales. When you are selling someone a million dollar solution, you have to look 'em in the eye and convince them that this deal is in their best interest. I couldn't do it because I didn't believe it. And even if I could have, there was no way on God's earth that I was going to endure these long sales cycles. (A sales cycle of a year or more was not uncommon.) I didn't even like hardware!

I started feeling an impending sense of doom. I was going to fail at this and there wasn't a fucking thing I could do about it (or so I thought). I went into a severe state of depression and had to take a medical leave of absence for 3 months. I never went back.

Okay, Carlos, so what is the moral of the story? We already knew that you were a loser. The moral of the story is that honesty counts, professionally and personally. A disingenuous (at best) value proposition is not sustainable. Period. A bait and switch strategy won't work. The marketplace is much too savvy. At best, it will lead to Pyrrhic victories and short-term gains.

IBM is an example of a company that has apparently cracked the code. They are wildly successful selling both big iron and professional services. I have no special insights into IBM, but they have most certainly managed to achieve success by having two totally separate business units.

Big Computer Company eventually bought Northwest Computer Company. For the talented employees of Northwest Computer Company, things did not work out so well. But they most certainly did for the shareholders.

Chapter 2: Competitive Advantage

Michael (Porter) Is Still ?Da Man

Competitive Advantage is an activity-based theory of the firm. To compete in any industry, companies must perform a wide array of discrete activities, such as processing orders, calling on customers, assembling products, and training employees. Activities, narrower than traditional functions, such as marketing or R&D, are what generate cost and create value for the buyer; they are the basic units of competitive advantage.

I will attempt to highlight how information technologies can impact these activities in a manner resulting in distinct advantages. Porter's concept of activities is virtually synonymous with Michael Hammer's concept of process, as brilliantly put forth in *Reengineering the Corporation*. This book rocked the business world when it was first released in 1994. It kicked off one of the largest monetary windfalls that the consulting industry has known in recent history.

Many would argue that the revolution failed. Certainly there were more than a few high profile failed projects to point to. But the same could be said about the Quality Movement, SAP, CRM, the Web, etc. The reason for most of these failures has more to do with our inability, from an organizational perspective, to adequately handle complexity and chaos than with the soundness of the underlying ideas and business premises.

The Usual Suspects Revisited

Another reason why these quasi-religious movements fail is because the usual suspects (and their poorer first and second cousins) pitch the "idea du jour" as if it were the Holy Grail. The search for the Holy Grail, and the implementation of the wisdom derived from it, always costs millions of dollars, involves a small army of people, requires a program, and is strategic to the survival of your enterprise. Nothing short of a complete overhaul will do. Talk about making mechanics and lawyers appear charitable.

However distasteful the results, we should be careful not to throw out the baby with the bath water. Much of what the business press subsequently labels as fads, are actually important pieces of the puzzle. Put together enough pieces and perhaps you will gain some insights into the big picture.

The danger lies in simplistic thinking. There are no silver bullets. Implement every one, or any one of these programs, and the results are almost guaranteed to produce anything but competitive advantage. Why is that? Because it seems obvious to me, to the point of absurdity, that a sustainable competitive advantage cannot be derived from something that your competitors can readily implement.

That is not to say there aren't good reasons to implement these solutions. If you had 30-year-old accounting systems, then a SAP implementation probably made sense. If your "customer facing" processes and systems are outdated, then by all means, develop an ROI for a CRM implementation. Just don't be conned into believing that you are going to gain a competitive

advantage. One could make a strong argument that these solutions are part of the cost of doing business, and nothing more. In order to derive a competitive advantage, you will need to add your own secret sauce, perhaps using these solutions as a launch pad.

Chapter 2: Competitive Advantage

Simply Profound

The beautiful thing about Michael Porter's analysis is that it is both profound and easy to understand at the same time. There are only two sources of competitive advantage:

That's it. There is another interesting variation, however. Do one or the other within a specific industry segment.

So, which activities within the value chain should you focus your scarce information technology resources on? Focus on a limited set of activities that you believe will produce the desired advantage. You don't have the money or the time to focus on anything else. This is basic b-school stuff, although you wouldn't have known it by the craziness that went on during the dot-com implosion. Information technologies have the bewitching quality, when well executed, to allow the pursuit of both (i.e. low cost provider and differentiation) simultaneously, yielding advantages that have been rare to non-existent heretofore.

I will make a concerted effort to link strategy with implementation (the "what" and the "how"). This disconnect is a root cause of catastrophic information technology failures. Almost always, the players on both sides of the equation are disparate functional organizations and, like the U.K. and the U.S., are separated by a common language.

The dichotomy results in the Blame Game process pattern (see next chapter). Often the resolution of this dichotomy is referred to as aligning the business strategy with the technology strategy. Although this phrase sounds profound, it is much too broad and vague to be useful. The alignment must happen at a more granular level (i.e. the activity or process) to be actionable.

Again to borrow from 'da man (M.P.):

Activities provide the bridge between strategy and implementation. When strategy was defined in terms of broad positioning concepts, a clear separation between strategy and structure was meaningful and useful. "What" and "how" were comparatively distinct. Armed with the recognition that the firm is a collection of discrete activities, however, previously common distinctions between strategy, tactics, and organization blur. Strategy becomes the particular array of activities aligned to deliver a particular mix of value to a chosen array of customers.

Specific activities within the value chain must become the targets of our information technology efforts.

Chapter 2: Competitive Advantage

The Second Law of Thermodynamics

The Second Law of Thermodynamics and Related Nonsense

Fasten your seat belts; this flight is likely to encounter some cognitive turbulence. This is where Spanish Harlem intersects with the second law of thermodynamics and information theory. You are about to enter the Leyva triangle, and there are no guarantees that you will emerge the same (or at all) on the other side.

Any serious student of software development is familiar with the term Design Patterns (singleton, model-view-controller, iterator, etc.). If not, then check out the book by the same name . At the risk of appearing academic (God forbid. I have a reputation to maintain), I am introducing the term Process Patterns. Process Patterns relates to the software development process itself, as opposed to specific design issues (i.e. artifacts that can be implemented in code).

Chaos and Recurring Patterns

I have observed a number of recurring patterns (usually after ingesting large quantities of peyote) that have an interesting and significant impact on our ability to develop systems better, faster and cheaper. There is nothing especially novel about the term “Process Patterns.” Other authors have used the term “Rules of Thumb” as a way of describing specific insights. Patterns are an integral part of chaos theory, and they convey information, allowing us to detect order within apparent randomness. Process Patterns attempt to do the same for the chaotic nature of software development.

What do the codes used for sending messages back from spacecraft have in common with genes or a molecule of DNA? How is it that the second law of thermodynamics, a physicist’s discovery, is related to communication, so that we can speak of the “entropy” of a musical score, or a page of text, or a conversation [or the software development process]? Why are knotty problems in the mathematical theory of probability connected with the way we express ourselves in speech and writing? The answer to all these questions is “information,” and the very fact that a single concept can link so many diverse ideas is an indication of its great generality and power.

The introduction of chaos theory, information theory, and entropy is necessary because these fields of study contain the necessary theoretical framework(s) and lexicon for talking about something that is messy; random; complex; unpredictable; gnarly; and, more often than not, downright confusing. In other words, it gives us a way to discuss the software development process and the complex information technologies that result from it. The goal is to develop an understanding of the demons at work during the creation of software.

The premise is that information technologies will continue to be the prime mover in the endless pursuit of competitive advantage. If this is so, then all stakeholders have a need to become smarter faster. Process Patterns are way of metaphorically speaking about the demons. While some patterns appear to name roles, they are intended to describe behaviors, as opposed to

individuals. These behaviors either increase entropy (a bad thing) or decrease entropy (a good thing).

I will use the following useful, if somewhat simplistic, definitions:

Damn, the boys in the hood are wondering if I have completely lost my mind. That notwithstanding, we shall press forward and see if we can make some sense out of this techno-babble.

Chapter 2: Competitive Advantage

Decisions in Progress

The following equation, from Ludwig Boltzmann, a top gun physicist of German descent, who lived in the 1880's, is engraved on his memorial stone in Vienna. And although amazingly simple, it lays the theoretical foundation for the rest of the chapter:

$$S = k \log W$$

When starting development of any complex system, the number of possible ways that the system could be arranged probably approaches infinity (i.e. it is a really fucking big number). The “W” is quite large; therefore, the system is in a state of maximum entropy. Order is imposed, and entropy is reduced, by reducing the number of decisions in progress (DIP). By making intelligent, methodical and relentless decisions, constraints are imposed. As constraints are imposed, the number of possible arrangements are reduced, order increases, and entropy decreases.

Competitive advantage, derived from applied information technologies, will be achieved by increasing the velocity by which the DIP yields to order. Process Patterns, or the lack of them, affect the DIP and have a powerful impact on whether your current project will actually contribute significantly to the end game (i.e. competitive advantage).

Chapter 2: Competitive Advantage

Interesting Asides

Before proceeding with the Process Patterns discussion (see next chapter), I propose a slight detour in order to consider some interesting asides, and to clarify certain contextual matters. The graphic below depicts an interesting problem that the auto industry is faced with.

Eighty percent of the new car delivery process is spent waiting for information! The implication is that better information is required so that quality decisions can be made faster. In other words, in order to improve this process, the DIP must be reduced dramatically. Information technology has a role to play in this effort, but it would be naïve to treat this as an engineering problem. Why is that? Because the critical part of the

problem is almost certainly related to the number of players involved in the process, and the fact that these players happen to be human beings. Human communication is chaotic and expensive. This was the golden insight that Fred Brooks had in *The Mythical Man Month* when he discovered that adding additional staff to an already late project would make it later.

Human Dialog as a Cost Factor

Human communication is expensive because human beings are messy; random; complex; unpredictable; non-linear; lazy (at times); and, more often than not, totally irrational. Language, as a communications protocol, is one of the most complex technologies on the planet. It is a beast that refuses to be easily tamed. How we communicate and collaborate with each other will need to improve dramatically if we are going to significantly compress the DIP within any non-trivial process.

Heuristics that improve interpersonal communications, within and across business processes, are as much, if not more, a part of the solution as any enabling information technology. The development of Killer Apps (i.e. systems) will necessarily need to include knowledge from a number of different disciplines, including psychology, anthropology, linguistics and biology, to name a few.

Process Patterns aim to improve our ability to communicate about the software development process by illustrating behaviors that have a significant impact on our ability to execute. They are a set of mental models that attempt to provide a common language related to specific problems and issues. If you buy into the premise that human communication and dialog is an important aspect of the software development process, then purchase (and study) *Dynamics of Software Development* by Jim McCarthy. This book is filled with wonderful anecdotes and profound insights, told from an industry insider's perspective.

Chapter 2: Competitive Advantage

Commercial Software vs. In House Development

I will use the example from the auto industry to help illustrate a subtle, but important, distinction between software development in commercial organizations, whose final product is not software (e.g. Ford Motor Company), and software development in the commercial software industry (e.g. Microsoft). Both use similar tools, methodologies, and human capital during the software development process; however, the organizational structures that drive the process are quite different.

IT departments tend to view their market as the internal customer organization (marketing, engineering, HR, etc.). It is a myopic view that tends to severely constrain potential solutions. A broader, more holistic, cross organizational worldview is required for efforts that are intended to create competitive advantage. Corporations have now entered the creativity business in a big way (yet most don't know it), and they must realize quickly that you can't get the genie back in the bottle (i.e. this is not your daddy's Ford Motor Company). There must be a willingness to experiment with some of this fuzzy logic if progress is to be made.

Companies that produce commercial software on a large scale are usually organized in a manner that mandates, and facilitates, communication between senior management, marketing, program management, development, and R&D in order to achieve specific results in the marketplace. This communication, although not always optimal (depending on the set of players), is built into the process.

The communications process for non-commercial software companies (as it relates to software development) tends to be ill defined. It also tends to be slower and less rigorous, negatively impacting both time-to-market and quality. There are a number of historical reasons for this, including the scarcity of institutional memory, as it relates to software development. The chapter entitled Movie Making and Software Development includes a description of an organizational model that attempts to address these deficiencies.

Microsoft Goes to War

What would Microsoft do if they considered exploiting the new car delivery process for competitive advantage? Well, I can't say for sure, obviously, but I can surmise a damn good guess. First of all, their best and brightest would study the problem to determine if the resulting advantage would be compelling. The probability is quite high that they would crack the code and determine that the problem is a human communications problem as much as a technology problem. They would then weigh this advantage vis-à-vis their current portfolio of competition killer projects to determine relative importance to the organization. If, after this process, this project were at (or near) the top of the list, they would declare war; arm their best soldiers; pull together a top gun, cross-functional team; appoint a field general; and get after it with a vengeance!

A technology plan would be developed that included, at a minimum, what should be contained in the next couple of releases. All of this reduction in DIP activity would happen in an incredibly short period of time. No amount of resources would be spared until the goal was achieved. Top management would be all over the team's ass to produce. Why? Because they are at war! Most corporations are totally incapable of this type of mobilization.

The discussion of concepts such as the DIP, chaos, and the 2nd Law is not meant to undermine the importance of strategic thinking and raw technology. Both remain incredibly important. However, the message from this chapter is that while these are necessary, they are not sufficient. Corporations will only be able to sustain a competitive advantage by applying these amorphous ideas (e.g. reducing the DIP)—in conjunction with strategic thinking and enabling technologies—to a set of activities within the value chain in order to produce the desired results.

Hopefully, the upcoming Process Patterns discussion will help to clarify the fuzziness. Top decision makers can no longer abdicate responsibility for technology decisions solely to the technology people. Technology is so tightly interwoven with the set of activities that you are attempting to impact that it is difficult to separate the two. Learn to pay attention, or risk funding one disaster after another. An often-quoted definition of insanity is to “continue to do the same things over and over and expect different results.” Allow me to translate this into Leyva speak for you:

If you continue to use the same fucked-up processes, you will continue to achieve the same fucked-up results.

One additional point of clarification: corporations need not transform themselves into Microsoft in order to derive competitive advantage from applied information technologies. The capability to produce insanely great software is not required. The creation of damn good software, in conjunction with things that you currently do well, is all that is required. In fact, the resulting work product will be, more often than not, a system (as defined in Chapter 1) that produces great results, as opposed to a product.

That said, the organizational model that is required to produce software-based intellectual capital does not have to possess all of the same characteristics that are found in successful commercial software organizations. However, there is much to be learned from seriously studying the best practices that have emerged from this industry. Keep in mind that you don't have to be the fastest, or the first; you just have to learn to out run the competition.

The ability to learn faster than your competition may be the only sustainable competitive advantage.

—Arie de Geus

(when head of planning at Royal Dutch Shell)

Chapter 2: Competitive Advantage

Understanding Complexity

While it is certainly true that the tools and methodologies of the software development process have improved dramatically, it is equally true that the complex problems we are attempting to solve are growing faster than our ability to comprehend them. This is not a temporary aberration. It is axiomatic that our technologies will always lag behind our understanding. Therefore, progress cannot be made if we keep insisting on deterministic solutions. Progress will only be made when we begin to look for the deep order within chaos. That is, when we begin to recognize meaningful patterns.

The first chaos theorists, the scientists that set the discipline in motion, shared certain sensibilities. They had an eye for pattern, especially pattern at different scales at the same time. They had a taste for randomness and complexity, for jagged edges and sudden leaps .

We must learn to amplify patterns that increase the probabilities of success, and minimize (or eliminate) patterns that increase entropy. Our sensibilities must become more attuned to the descriptive, rather than the prescriptive. This is out of necessity. It is the best that we can do (for now).

The “X” Diagram

I will use the “X” diagram below to illustrate why IT organizations have had such a difficult time delivering viable technology enabled solutions. IT organizations came into existence in order to provide a bridge between the business world and technologies that were beginning to have an impact on the business landscape. Initially, their role in life was to automate existing manual processes related to back-office systems (accounting, payroll, purchasing, etc.). The rectangle depicts the solution space that was IT’s purview.

As you recall from Chapter 1, in the days of IBM domination, all four aspects within the rectangle were essentially stable, certainly by today’s standards. However, with the advent of the destabilizing technologies previously mentioned, each aspect individually, and all of them taken together, began to go non-linear (i.e. they became chaotic). At the same time, IT was increasingly being asked to innovate, as well as automate.

The established order began to unravel, and it continues to reel from the onslaught of discontinuous change that wreaks havoc on their ability to cope. This technology stuff is no longer only an IT problem. It is an organizational problem. One that requires a radically different approach if we are going to leverage its benefits in a meaningful (and cost effective) manner.

We are never going to conquer complexity. All that we can do is become better acquainted with it, and develop an eye for the jagged edges and sudden leaps. We need to build into our processes an understanding that, at the inception phases of a complex system, we know very little about the end game. We have some vague ideas about the story we are creating, but only the creation process itself will reveal the true, and final, meaning.

The whole question of complexity is one that has proved surprisingly subtle when investigated by modern theorists. Complexity is not just a matter of a system having lots of parts, which are related to one another in non-simple ways. Instead, it turns out to be a special property in its own right, and it makes complex systems different in kind than simple ones, enabling them to do things, and be things we might not have expected .

It turns out that the whole is not only greater than the sum of its parts, but also quite different from it. Many of the E-Business systems that we are currently developing are more complex than anything that we have attempted previously. We must learn to eat the complexity elephant one bite at a time. The big bang theory is a recipe for disaster. Let me provide an example to help illustrate the complexity involved in such systems.

Chapter 2: Competitive Advantage

Workflows and E-Business

The graphic below is a simple, yet representative, illustration of a typical workflow process within any organization. Transactions are captured by your front line data entry employees and deposited into your information systems as data. Knowledge workers subsequently massage the data and attempt to turn it into information. Information is subsequently routed to individuals that require it.

This type of workflow happens, in most organizations, via loosely coupled manual processes. Sure, individual components of the workflow (i.e. transactions, analytics, and messaging) are automated, but the workflow itself is not. Attempts to automate workflow have met with mixed (mostly poor) results. The low success ratio is due mostly to the complexity inherent in applying enabling technology across internal organizational boundaries, for reasons that have been previously expounded upon (ad nauseum).

Now, consider what is involved in a typical E-Business system. It usually involves applying enabling technology across corporate boundaries in order to reduce inefficiencies within the supply chain. In essence, workflows from different corporate entities are being linked together in what is now called a business web (b-web). There is no doubt that b-webs, enabled by web services technology, will have a powerful impact going forward.

But let us be clear about one thing: the complexity involved in making b-webs real is greater than anything we have attempted heretofore, notwithstanding compelling technology and vendor claims to the contrary. The technology offerings keep improving, and should certainly be leveraged, while at the same time recognizing that most (but not all) of the complexity lies within our processes and our abilities, as human beings, to effectively communicate.

Chapter 2: Competitive Advantage

The Next Big Thing?

You must be kidding. There will always be a next big thing! It is a meaningless question. Hopefully, this is not the question that you had foremost in your mind at this point in the discussion. The question I would be asking is “What can I do to improve upon this dreadful state of affairs?” Thanks for asking. The answer is simple, but its implementation is daunting: improve your organizational process literacy with respect to software development.

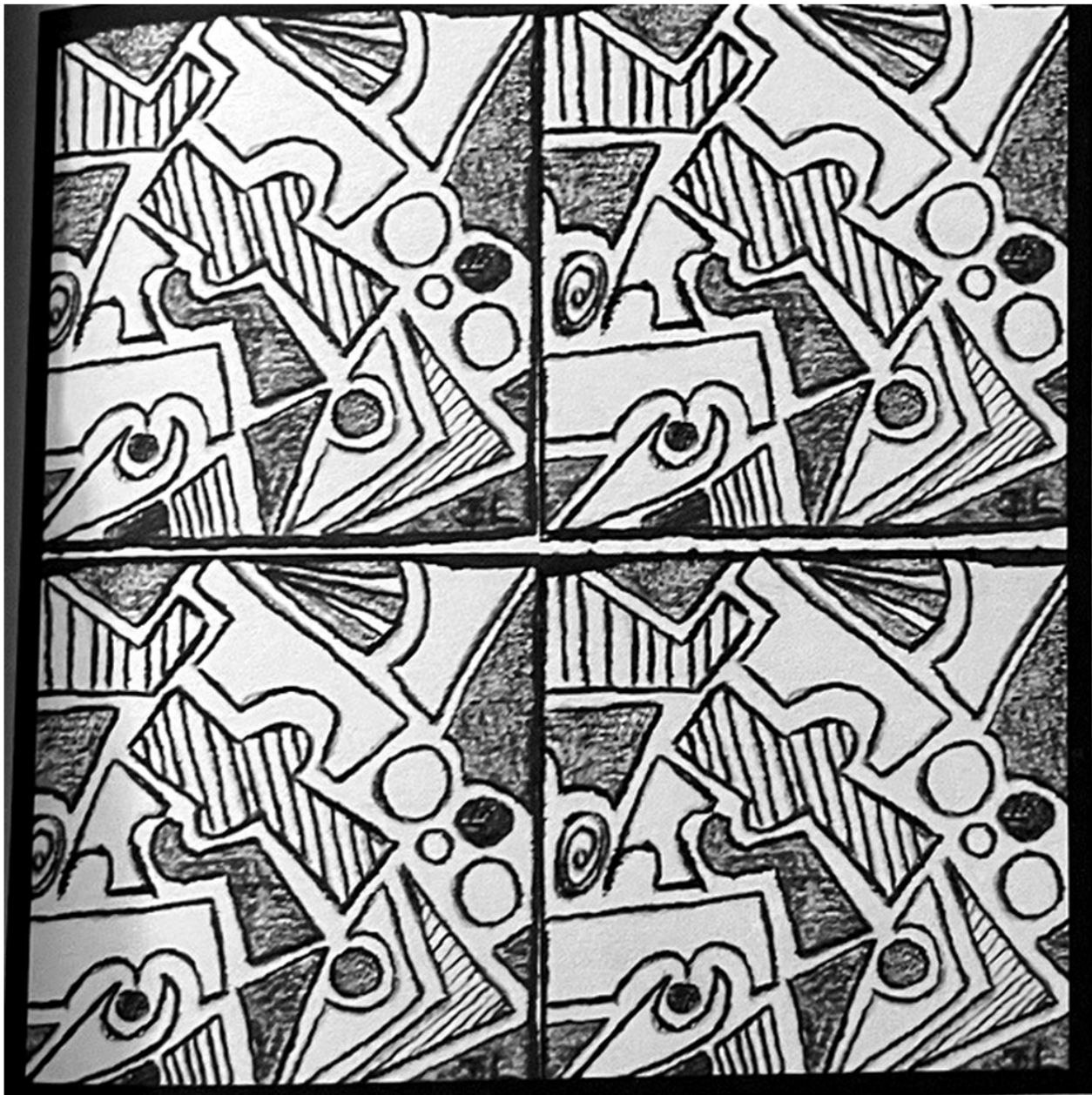
There is no question that software will become even more strategic going forward.

Much of a corporation’s expertise and procedures are represented in its software. Corporate software is rapidly growing in complexity. Packages, templates, software objects, and automated development tools will allow increasing complexity as time goes by, encapsulating an ever-growing body of human know-how. Some corporate software has become a valuable strategic resource that could not be replicated quickly by competing organizations.

When American Airlines was highly profitable, its CEO, Robert Crandell, was asked whether he would sell the airline. He replied that given the choice of selling the airline or selling its software, SABRE (which links the airline to booking agencies worldwide), he would rather sell the airline; the software was more valuable.

The remaining chapters in this book focus on heuristics that aim to improve corporate literacy in the software development process. The intended audience includes nearly all stakeholders within the firm, especially senior executives and information technology professionals.

Obviously, one need not become an IT professional in order to acquire the necessary literacy. However, a sustainable level of organizational literacy is required, since nearly everyone within the firm will be required to participate in these creative endeavors as we all build the software-centric enterprise. It is my firm belief that, for the foreseeable future, the software-centric enterprise will be the only type of enterprise that will consistently attain competitive advantage.



PROCESS PATTERNS

Chapter 3

Process Patterns

To some physicists Chaos is a science

Of process rather than state,

Of becoming rather than being.

Process Patterns are equivalent to a meme complex (i.e. a group of mutually supporting memes) that metaphorically relate to the software development process. Adaptation of these memes, by a software development organization, should result in a cultural transformation consistent with the goal of leveraging technology for competitive advantage.

This transformation, along the lines alluded to in this chapter, and described from different angles in subsequent chapters, is the secret sauce that has for the most part eluded business computing organizations (and others as well). It is more important by far than the latest wiz-bang technology. In order to transform the culture you must infect the organization with the appropriate memes, and disinfect it of inappropriate ones. This implies a deep understanding of the desired results, even as you walk down a road less traveled.

Work in Progress

My intent is to provide a starter set of patterns, a hint of the possibilities. Hopefully, other practitioners will contribute to this effort over time. I will be more than happy to aggregate the content and publish it on any appropriate sites (e.g. www.theb2bdepot.com). Most patterns deal with the non-linearity of the communications process during software development.

Non-linearity means the act of playing the game has a way of changing the rules. In terms of the software development process, who the players are and what they do (i.e. to, with, and for each other) matter. Most Process Patterns relate to the social and organizational interactions of the players, and the consequences of these interactions.

The premise and overriding principle of Process Patterns is that they impact the manner in which the players interact. Positive patterns improve the human dialog among team members by orders of magnitude. Improved dialog dramatically reduces the DIP, resulting in quality and time-to-market advantages as described in Chapter 2. The willful use of Process Patterns drives cultural changes by allowing the formation of creative and productive spaces.

Patterns are Good & Bad Behaviors

Some patterns are negative and increase entropy (e.g. Bad Actors & Blame Game) and need to be understood so they can be recognized and eliminated. However the majority of patterns are entropy reducing and positive in nature. These patterns need to be quickly introduced into the meme pool and propagated. The Missing Factory is more of an organizational pattern, as opposed to a process pattern, and will be discussed in Chapter 5. The remaining patterns are discussed next, in no particular order. Many of the pattern names selected turn out to have connotations

within the film industry. That was not the intent when I started down this road but it began to make more and more sense, so much so that I decided to dedicate an entire chapter to the similarities.

I must re-emphasize the fact that pattern names, while appearing to name roles, are intended to describe behaviors. These behavioral patterns, although principally embodied in a specific role, increase in effectiveness the deeper they are embedded within the consciousness of the team. The wider and deeper the distribution the more effective communication becomes among team members. In other words, any and all team members should be capable of invoking a specific pattern when necessary, their individual talents permitting.

Chapter 3: Process Patterns

A Set of Heuristics

I need to clarify one important point before proceeding. Process Patterns can be considered interesting heuristics but are not intended to replace iterative methodologies such as the Rational Unified Process (RUP). In fact, Process Patterns can be thought of as observable behaviors that provide insights and are complementary to such methodologies. I am a strong believer in iterative development and often use a modified version of RUP as a development guide.

The following Process Patterns will be covered:

Chapter 3: Process Patterns

Call to Arms Pattern

Call to Arms concerns itself with motivating the troops to perform at the extraordinary levels required to deliver compelling solutions. The pattern should be delivered early by the executive sponsor (e.g. the Producer) and subsequently (at critical points in the project) as deemed necessary by the Director. The importance of the mission must be constantly reinforced.

If this is a mission critical project, and by definition any project purporting to deliver a competitive advantage would be, then a number of key (and important) stakeholders must take an interest in it. You know you have succeeded in infecting the extended team (i.e. all the players as opposed to just the IT players) with this meme when you see the pattern used by individual, non-management contributors.

Ned Johnson, the man primarily responsible for turning Fidelity Investments into the financial powerhouse it is today, delivered many Call to Arms messages by his mere presence:

His chief passion seemed to be computers: the installation, maintenance, and use of the giant mainframes that would be the backbone of the company in the 1980's was what most engaged him, and he was constantly in the computer room, watching, asking questions, tinkering with the machines. Naturally, his staff thought that he was pouring too much money into computer capacity, and viewed his obsession as folly. Naturally, they would be proved wrong as soon as the bull market [of the 1980's] began.

In the many years that I have spent in this business, I have rarely witnessed key executives paying close attention to their technology projects, even those deemed to be mission critical. How do you think the employees at General Electric might have felt, in the days of Jack Welch, if he started hanging out with a project team? My guess is that they would have been motivated beyond belief. The Call to Arms meme would spread through his organization like wild fire.

Delivering the message is important, but the pattern will replicate itself faster by the things that you do.

Supporting Cast Pattern

The Supporting Cast pattern relates to the edification of certain individuals and groups that have immense power to contribute significantly (or not) to a project's outcome. Depending on the project, the supporting cast might include prospects, customers, domain experts, quality assurance, technical writers, office administrators, contractors, junior staffers, and secretaries. It is important for the Director to make a concerted effort to propagate this meme by including the supporting cast in appropriate meetings and celebrating their contributions. It is critical that the supporting cast be edified and made to feel important because:

Do not confuse this pattern with altruism. If you value the prize (i.e. competitive advantage) then you will propagate this meme out of pure selfishness. Oftentimes the best insights come from places where you least expect them.

Chapter 3: Process Patterns

Producer Pattern

The Producer pattern is all about ‘da money. It embodies the business and fiscal implications of the current project. The Producer pattern captures the viewpoint of the executive sponsor who is risking their reputation, and oftentimes their money, on the current venture. When this meme propagates to the entire team the work becomes grounded, helping team members keep things in perspective. The graphic below represents three tradeoffs that must constantly be balanced in any project.

It is often all too easy, especially for technologists, to forget who is paying the rent, and for that matter, that the rent needs to be paid at all (no matter what)! The Director must imbue the team with this pattern so that all project stakeholders focus on business and fiscal responsibility, or at a minimum, develop a keen awareness of these issues. Maintaining a business perspective often facilitates making appropriate tradeoffs when team members are debating alternative strategies.

Show ‘em the money and the team will learn to have a healthy respect for it.

Director Pattern

The Director is all about leadership and vision. This pattern is responsible for promoting the project’s story and ensuring that entropy is fought every step of the way. It is principally responsible for meme propagation and insights. It concerns itself with timely decision making (or breaking) and maintaining the tenuous balancing act between creativity, business, and technology.

The Director is the creative genius and the rock. The pattern (and the role) that project team members must turn to when (not if), from time to time, they find themselves Wandering Around the Desert. It provides both food and drink at the appropriate times. It accepts responsibility without attempting to assign blame.

The Director is also a problem-solving pattern. It concerns itself with the project as a whole; part inspiration, part perspiration, part genius, part sheer will and determination. It is the heart and soul of the endeavor, willing to sacrifice (almost) anything for the end game.

Where there is no vision, the people perish. The Director is the keeper and the defender of the flame. It is the Director’s responsibility to confront Bad Actors and to inspire great performances from good ones. The Producer may put up the money but it is largely a bet that the Director will deliver the goods. This pattern (and role) requires passion:

Movies aren’t for me a piece of life. It isn’t a job, it’s everything. You can’t make them unless you are obsessed. They tend to take you over. They guide your dreams as much as they guide your waking. There’s no escaping movies.

Director Phillip Noyce

(The Bone Collector, 1999)

The same holds for projects that contain software as a principal component. Without the Director meme great software is incapable of coming to life.

The Director must be an embodiment of what Jim McCarthy calls the group psyche:

Your personal authority stems from knowing how to create the team, to initiate its growth, to double and redouble its growth, while empowering the team with good techniques and procedures for creating great software and shipping it on time. Your vision is of the group psyche and of the product of the group psyche, and of how the two relate. Your goal is to be a genuine authority in these matters, not some bogus institutional authority...

Chapter 3: Process Patterns

Bad Actor Pattern

The Bad Actor is an individual, in a specific role, whose behavior tends to increase, rather than decrease, the entropy of the system. It is the responsibility of the Director to modify this individual's behavior early in the game (or at any other time the pattern is manifested) or eliminate the individual from the role (not necessarily the organization).

Bad Actors tend to have lots of questions but appear to be totally incapable of contributing to a solution. They tend to postpone or outright sabotage the decision making process. They are cancerous, and if their behavior cannot be dramatically improved, must be cut out, or risk killing the patient.

Just Say No

Big Automation Company hired a project manager to lead the development of their next generation simulator. This individual appeared to have all the right credentials and pedigree. She had a solid technology background, had project management experience, and had worked on similar products at different companies. I was the chief architect, but it was her project.

Initially I coordinated some Breaking Ground activities in order to get the ball rolling. I gradually eased myself out of the process in order to let her assume a leadership role (i.e. what she was hired to do). After three months, it became agonizingly painful to endure the endless excuses as to why she could not develop a meaningful project plan.

This was the next generation simulator. The current product had been extremely successful in the marketplace but had become long in the tooth (understatement). The plan was to re-architect and re-develop the product using state of the art technologies and add some features that would leapfrog the competition.

In other words, the next generation product had to functionally do everything that the old product did, and a little more, only with a different set of base technologies. We knew the "what", it was the "how" that was in question. This individual's primary complaint was that no one had sufficiently established the requirements. Never mind that it was her responsibility.

Shoot 'em if you have to

Finally, after numerous (far too many) attempts to rectify the situation, my patience was stretched to the breaking point. I was forced to call bullshit. Most of the requirements were embodied in the current product. The task at hand was something that a competent project manager should have been able to handle with aplomb. There was another dynamic at work here (i.e. fear, uncertainty, and doubt). The entropy was increasing at a significant rate and impacting the entire team.

The product development manager eventually took the appropriate action and removed this individual from the organization (there was not another suitable position for her to assume). The removal of Bad Actors is a painful process for all concerned. However, it comes with the territory

and must be handled in a professional but expedient manner.

Elimination of Bad Actors dramatically improves the script.

Chapter 3: Process Patterns

Key Abstractions Pattern

Key Abstractions are intimately related to the project's story line. It is a term borrowed from the Rational Unified Process (www.rational.com) and given a somewhat broader meaning within the process pattern context. In technical terms key abstractions refer to entities (i.e. people, places, and things) about which persistent information will be maintained. There should be widespread agreement among all project stakeholders relating to the centrality of these entities because the entire story is subsequently built around them.

At the project's inception we are dealing with first order Key Abstractions. For any non-trivial project there will exist second order Key Abstractions, third order Key Abstractions, and so on. It is next to impossible during inception to deal with any abstractions beyond those considered to be first order. Moreover, it is not necessary to do so. Subsequent abstractions will become obvious as the project's iterations progress and will become easier to recognize and/or define at the appropriate time.

Once Key Abstractions (of whatever order) have been identified it becomes necessary, and plausible, to identify the processes that act upon these entities. Given a set of processes and a set of Key Abstractions a reasonable iteration plan can be developed.

Focus on Key Abstractions in their relative order of importance.

Breaking Ground Pattern

Breaking Ground relates to getting started—duh. You might think that this pattern is only manifested at the start of a project, but you would be totally mistaken. Each non-trivial project is composed of hundreds, sometimes thousands, of smaller sub-projects that are not explicitly captured in the plan. Often experienced project hands even find it difficult to get started on these. What then happens is akin to the writer's block phenomenon. Sub-projects often require small R&D efforts and can send team members down a number of tangents. Critical intervention can jettison the effort by reducing the DIP.

It is the responsibility of the Director to ensure that Breaking Ground happens early and often. At the project's inception this usually means developing the project story and coming up with the first order Key Abstractions. Until the project story, as embodied in the Key Abstractions, is reduced to paper, nothing can happen and nothing should happen.

If it hasn't been written down, it hasn't been said. Be sure to institute a lightweight process that requires that all important and relevant project decisions be documented.

—D. Leyva

The Breaking Ground pattern should be invoked whenever the Director suspects that writer's block is preventing progress. The Director must be constantly vigilant in seeking out Breaking Ground opportunities since team members are often reluctant to come forward of their own

accord. They are, more often than not, likely to thrash about and ask for help only out of desperation.

Keep your shovel in hand. Breaking Ground opportunities are everywhere!

Chapter 3: Process Patterns

Collaborator Pattern

The Collaborator refers to the team's neuron cells. These are principal meme receptors and propagators. The Collaborators love white boards and heated constructive arguments. Winning or losing is not the crux of the matter. The improved understanding that results from the battle is the prize.

In order to increase the velocity of this meme the Director must ensure there are ample war rooms available for impromptu engagements and that team members feel “safe” to explore ideas without fear of retaliation or humiliation. Team members that are hopelessly infected should be encouraged to engage in the fray as often as possible. Those not infected need to be seduced onto the battlefield.

One of the side benefits of this pattern is the identification of Bad Actors—you bet your ass! Bad Actors easily tire of war room battles and they tend not to participate regularly. Why is that? Well, it has mostly to do with the fact that arguments against a position are only considered valid if a better alternative is proffered. If a better alternative is not offered up for dissection then the contestant is called on the carpet for whining. A contestant that consistently whines is almost always a Bad Actor.

Although there is certainly a role for Super Stars it is the contribution of every team member that collectively defines success. Individual stars must be rewarded relative to the team's success and relative to their contributions as Collaborators. However, that said, do not hesitate to adequately reward a Super Star.

Collaborators run the project's central nervous system.

Organizing Principle Pattern

Organizing Principle refers to context and connectedness. Rarely in today's software universe does a product, either for external release or internal consumption, stand-alone. One-off offerings are headed for a short shelf life and will quickly find themselves with a permanent residence in history's dustbin.

The project team must keep in mind related stories that the current effort plugs into and / or otherwise has to play nice with. These stories can be standards, product suites, architectures, add-on offerings, protocols, etc. The Organizing Principle must be made as explicit as possible during the development of first order Key Abstractions and propagated continuously during the project's lifetime.

The Organizing Principle is an important forward looking pattern that must be constantly re-visited in order to ensure a degree of seamless integration with surrounding stories.

One-off offerings are DOA before launch.

Chapter 3: Process Patterns

Planning the Plan Pattern

Planning the Plan relates to the planning process itself and not the concrete deliverable. It is the entire team's responsibility to participate in the process, however, during the inception phase of the project, the responsibility for Breaking Ground falls most squarely on the shoulders of the Director.

The Producer at this point has tentatively decided to pursue the project and has some vague idea about the details of the value proposition. It is up to the Director and the team (management, sales, marketing, technology, etc) to clarify the value proposition and devise a strategy for attaining it. In order to expedite the process the Director must frame the problem by developing a "straw man". The straw man contains the Director's initial thoughts related to the following non-exhaustive list, depending on the project:

Once the Director has gathered and (crudely) documented his or her initial thoughts he or she meets informally with key stakeholders to surface the ideas and refine them. The straw man is a device that helps plan the plan. It is pure tradecraft—far removed from anything that vaguely approximates a science, anyone that claims otherwise is completely (and utterly) full of shit.

The straw man should take no more than a couple of weeks to develop. Once developed it is time to meet with the entire team for a kickoff discussion. The purpose of the kickoff discussion is to interrogate the straw man. The interrogation(s) results in a list of tasks required to fill the holes (i.e. identify risks, gather additional requirements, develop rigorous use cases, further define project economics, etc., etc.). Filling the holes should take (approximately) 2-8 weeks, depending on resources, complexity, etc. If the holes cannot be filled to the team's satisfaction, then throw the fucker out and start again.

At the point that holes have been filled the inception phase of the project is complete. Do you have all the answers that you need? Can pigs fly? The RUP defines only four project phases:

The Elaboration phase contains iterations that are aimed at reducing risks and getting real answers. OK, so what's the point? It is an exercise in futility to drag the Inception phase on and on in search of perfect requirements (use cases, economics, or any other fucking thing). Software is n-dimensional, temporal, and organic, a living and breathing beast.

You will never be able to model the beast on paper. If you want to understand the beast then you need to interact with it! Only by interacting with it can you determine if the beast should live or die. This determination is made during Elaboration.

Elaboration iterations should be planned in the same way. Develop an iteration straw man. Get some feedback. Refine it. Meet with the team and get task estimates. Refine them. The planning process should be as iterative as the development process. Expedite it by Breaking Ground early and often.

If you make it to the construction phase then, by definition, you have all the answers you need to give the beast life (or so you think). During Construction you build the beast. During Transition you cut the umbilical chord and hope that the world treats this thing that you have hatched kindly. Hopefully it survives birth, adolescence, college and goes on to have a long and distinguished career.

Planning is a motion creation tool. Planning reduces entropy by organizing chaos. A plan is outdated and wrong, with respect to specifics, as soon as it is committed to paper, however without one you cannot proceed. It is a development tool and nothing more.

Learn to plan the plan and don't hesitate to modify it when necessary. The actors must help write the script.

Chapter 3: Process Patterns

Wandering Around the Desert Pattern

Wandering Around the Desert relates to what goes on during the middle of a project. During Inception and early iterations of Elaboration you are generating excitement, anticipating the prize, building momentum, identifying and eliminating risks, and selling the story. There is a perceptible buzz around the team's activities. You know that you will be challenged and you gear up for it.

During the middle of the project the long hard winter sets in. You start to believe that you have answered the disbelievers, eliminated the most serious risks, and have all the executive sponsorship you need to get you through the night. All that remains is flawless execution and a tremendous amount of work. The excitement diminishes. The work begins to take its toll. You can't hear the buzz. Things start to slip.

In every project, there comes a time when you wish with all your heart that there was some way out. Everything is in doubt. Cynicism and mutiny are in the air. No one has seen land for months. The crew is suffering from scurvy, the larder is empty, the stormy seas seem endless.

Welcome to the middle game.

It is time for the entropy police (i.e. the Director) to go on patrol 24/7. The Director knows that the team and project are a long way from being out of the woods. In fact, the forest and trees converge and then disappear. All that is left is miles and miles of desert and a very unreliable compass called your instincts, which by definition can't be very good otherwise they wouldn't have led you into the fucking desert!

The only ray of hope is that the Director knows the desert. Is capable of finding food and water at the appropriate times. Recognizes landmarks that lead to nearby villages wherein the council of elders can be called upon. Knows when to push and when to rest. Is constantly vigilant about team members succumbing to sunstroke. Mitigates disputes that arise as team members start to lose faith. Knows that progress must be made each day even though there is no guarantee that you are walking in right direction.

In the middle of the desert bandits that you thought were friends will try to ambush you. You must anticipate their arrival and forewarn your team of their inevitable appearance. They will strike with greater force and frequency the closer you are to emerging from the your long and arduous pilgrimage.

The final problem of middles is often the most frustrating. Even if you have built a coalition and involved key stakeholders, the critics, the skeptics, and cynics will challenge you. They will be strongest and loudest not at the beginning but in the middle of your efforts, just when the project itself is not quite ready and thus is most vulnerable. It is only then the possible impact of the change becomes clear. And those that don't like it have had time to formulate their objections and harden their positions...And now that it looks like the venture might succeed, the threat to those that oppose it increases.

The desert can be conquered but, no matter how many trips you have made, it always poses a life and death challenge. Do not celebrate too much during early stages because the desert and the bandits await you. Be ready.

Chapter 3: Process Patterns

Super Star Pattern

Super Star refers to exceptional and bankable actors. These are individuals that are head and shoulders above their peers (e.g. Michael Jordon and Larry Bird) and tend to make everyone around them better as well. Every project team should consider itself blessed if it has a Super Star (more than one may be problematic) in its cast. Not only do these individuals pull their weight, they often pull more than their fair share.

Do you believe that it is reasonable, and just, for Super Stars to be compensated only marginally better than their peers? Would Al Pacino or Robert De Niro be happy with this type of compensation knowing they are the box office draw? It is widely known, for example, that the best developers are capable of producing an order of magnitude more quality “stuff” than those that are average. In business computing organizations I have never seen this compensation chasm adequately addressed. You do not have to pay Super Stars 10X more than everyone else, but you do need to recognize their achievements and provide preferential compensation.

All this talk about the importance of teams should not prevent us from treating special individuals in unique ways. Money should always be part of the equation but preferential treatment need not stop there. Paid vacations, T1 lines at home, use your imagination! Yeah this will drive HR up the fucking wall but so what? I will go out of my way to accommodate this type of individual, as long as the reward system does not turn them into a Bad Actor.

If you are a Director and you are asking yourself how do you know that someone is a Super Star then you should fire yourself. A good Director always knows.

Super Stars need to be treated and compensated as such.

Chapter 3: Process Patterns

Blame Game Pattern

The Blame Game is the most insidious of patterns. It is anathema to team dynamics and causes a cancerous growth, that once it reaches a certain magnitude, becomes next to impossible to remove. In business computing organizations the Blame Game is a team sport (often played to the death) between end users and technology. In companies that ship commercial software the sales and marketing organizations replace end users as contestants. In both cases symptoms and remedies are the same.

Before suggesting a remedy, let's examine the root cause and discuss the reasons why the game always has a tragic ending. The premise is that you have two disparate organizations jointly responsible for project deliverables (e.g. finance and technology). The inevitable dispute centers on that most venerable of words within computing history, requirements.

The business people are supposed to provide correct requirements. The underlying assumption is that they know what they want and they know what they are doing. Both assumptions are almost always incorrect. Business people have (almost) never been trained in the development of formal requirements. They normally do not think in terms of systems, entities, and relationships between entities, or any other of the arcane artifacts and constructs that technology people require in order to get what they need. They also perceive (correctly) that all of these activities are a royal pain in the ass and an additional burden on top of their already busy schedules. They see the requirements gathering process as all downside and no upside. Is it any wonder that they are not active, willing and enthusiastic participants?

When technology people abdicate their responsibility for requirements they are setting themselves up to lose. They are also setting up a scapegoat. Requirements must be coaxed out of the business people. Technology people must become business savvy enough to facilitate this process. They must develop "straw men" that provide a point of reference. Business people are significantly better at telling you what is wrong than what is right. They are far better at telling you what they don't like than what they do like. It is a dereliction of your responsibilities and inexcusable not to have a better understanding of the business. Quit blaming the business people for incorrect requirements.

When business people "willy-nilly" change their minds in the middle of a project. When they ask for "must have" features without considering costs. When they ask for functionality that is obscure, absurd, and often ridiculous when considered in the grand scheme of things, they are setting themselves up to lose. They are also setting up a scapegoat. Business people must become more techno-literate. They must learn the basics of the process. They must become team players and accept their share of responsibility as well as their fair share of the load. They must do this in order to ensure team success and get their fair share of the credit. Quit blaming the technology people for your carelessness and afterthoughts.

There is only one remedy. There cannot be two teams on any project that is mission-critical and supposed to provide competitive advantage. There is one team. The Director reports to the

Producer, everyone else reports the Director (either directly or indirectly). The Blame Game ends because everyone is to blame. There is only one Berlin. The wall must be permanently and irrevocably blown up and dismantled. We live together and we die together, anyone that continues to sit on the fence gets shot first.

Eliminating the Blame Game is a major first step toward a successful blockbuster movie and a major first step toward creating one cross-functional (non dysfunctional) team.

Chapter 3: Process Patterns

Summary

Damn almost an entire chapter and I rarely exercised my favorite adjectives. I am definitely fucking up. But not as bad as you are [Mr. or Mrs. CEO, CIO, VP, VC, and Chief M.F. In Charge] if you don't start paying attention! Follow Ned Johnson's lead, hang out with the geeks, ask questions, tinker with the technology and participate in the process.

But I must warn you. If you can't stand the heat, stay the fuck out of the war rooms; it's awfully hot and messy in there. Check you titles at the door, they won't do you any good. By the way, bring some value to the battlefield, the geeks are getting damn good at recognizing Bad Actors.

It's about one B-I-G idea: innovation/a "top-line" obsession.

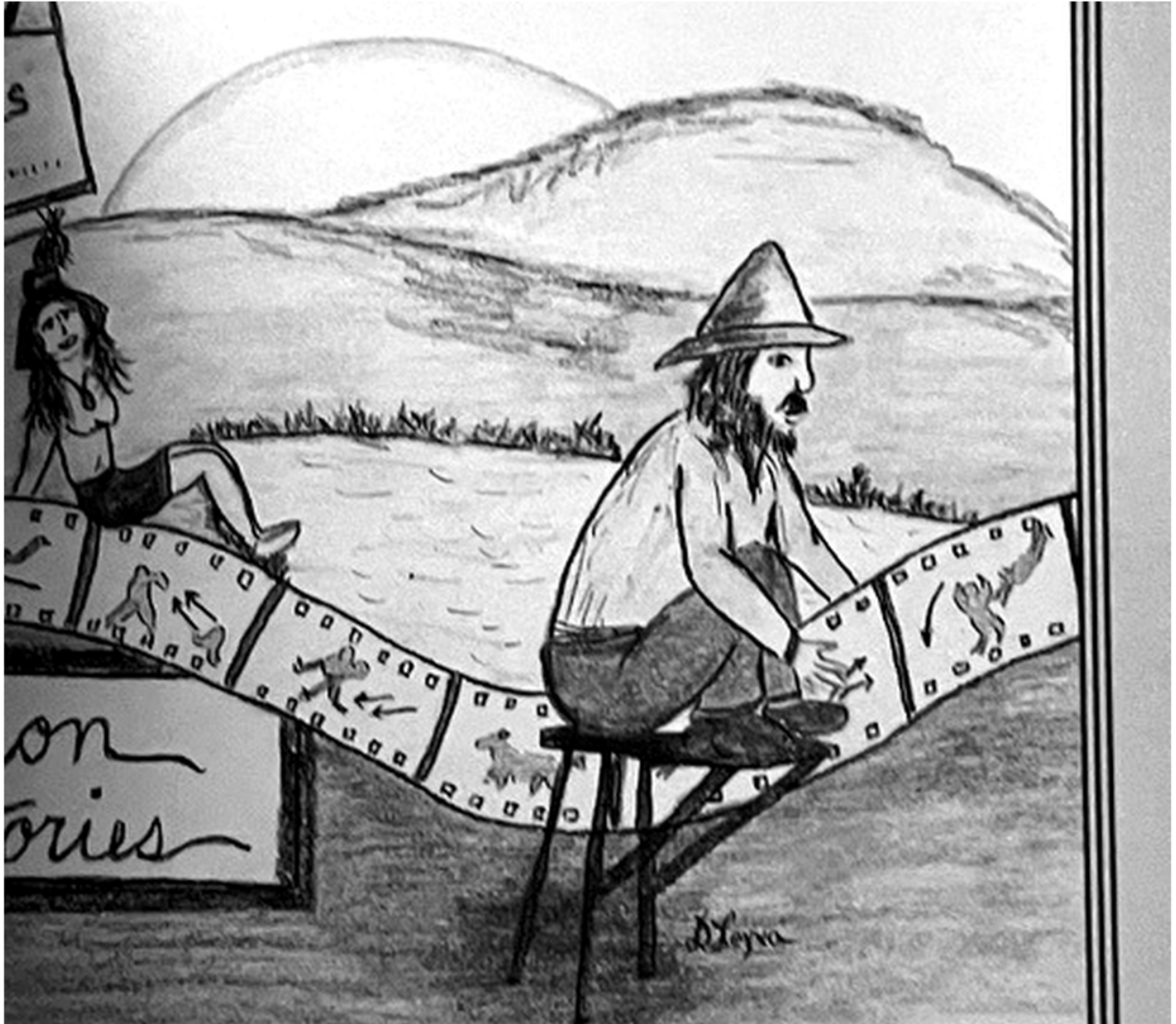
And in particular it is about process innovation within creative spaces. Although this chapter (and book) focuses largely on the software development process, all business processes will soon be impacted. No stone will be left unturned (watch out for snakes). Software will permeate everything it touches, and with it creativity as well.

Far better to dare mighty things, to win glorious triumphs, even though checkered by failure, than to take rank with those poor spirits who neither enjoy much nor suffer much, because they live in the gray twilight that knows not victory, nor defeat.

—Theodore Roosevelt, 1899

You are in the movie business. Hire yourself a Producer, a Director, a Supporting Cast, and as many talented young actors as you can find. Inspire them and they will create the future for you. I guarantee that you won't be disappointed!

During tulip mania there was nothing but bad scripts, bad acting, and everyone was busy blaming everyone else. No wonder things turned to shit! Fear, greed, and chaos are no substitute for creativity—they always lead to broken dreams and endless nightmares.



MOVIE MAKING AND SOFTWARE DEVELOPMENT

Chapter 4

Movie Making & Software Development

This chapter draws on the similarities and parallels between movie making and software development. I will be stretching the limits of artistic license since I am certainly more familiar with one than I am with the other. In fact, I know very little about the intricacies of the movie making process, and the research that I have done is a poor substitute for experience. However, that said, there are broad themes that seem to apply to both, and in particular, I believe that the art of software development, is just that, art.

Software development is art with technological underpinnings just as movie making is art with technological underpinnings. Both are complex, creative, expensive and risky story telling vehicles. Despite the similarities that exist between the two, due to the nature of the creative processes required for both, it is the parallels between the economics and the organizational dynamics that I find fascinating and relevant.

The Players

Consider the players in the movie making process. Not only do you have the big name players: the producer, the director, and the actors; you have a small army of little name players that all make significant contributions: writers, costume designers, cinematographers, sound engineers, editors, musicians, animators, stunt men and women, hair dressers and so on.

No producer or director in their right mind would consider working with anyone who is not at the top of their game in any one of these disciplines. Movie making is risky business and you cannot afford to work with amateurs when even the best of the best often get it wrong. In order to recruit this small army of top talent you have to deal with the numerous agencies that represent them. Pulling together the talent required to make a major movie is a monumental project management task.

Consider the players involved in the software development process. Not only do you need developers and visionary leaders; you need the services of an entire supporting cast: architects, database specialists, network specialists, technical writers, domain experts, content creators, copy editors, web designers, usability experts, graphics and multi-media experts, sales and marketing and so on. All of these players are required to tell a story in software.

What are the chances that at XYZ Corporation, which is not in the software business per se, you will find all the top talent you need to tell the story in house? If the story you are currently creating is mission critical and will cost in the millions, can you afford to work with amateurs in any one of these disciplines?

Within XYZ's current software state of affairs the probabilities are high that your supporting cast will not be ready for prime time. For many supporting cast members this may be the first time they tell a story in software. The supporting cast is likely to be both talented and full of potential, but these characteristics are no substitute for experience. Chances are that both quality and

time-to-market will suffer.

Chapter 4: Movie Making & Software Development

The Process

Movie making and software development are process and project driven endeavors. They both have a well-defined beginning, middle, and end. They require the collaboration of a wide array of talented individuals, all motivated and focused on producing a unified and coherent deliverable. Project risks run high and catastrophic results are legend.

The successful completion of a movie is a miracle of sorts. The process demands that a large number of independently artistic people come together with a single goal—the successful rendition of an idea to film, finished within a restricted time frame and budget.

Replace the quote above with—the successful rendition of an idea to silicon—and it works equally as well for software development. Although there certainly are a large number of bad movies produced each year, I would argue that Hollywood, on the whole, does a much better job of managing large and complex projects than we do in the technology business.

Assuming that this premise is valid, the question that begs asking is why? Perhaps one answer is that, for the most part, only top talent gets to play. You could probably build a strong case for this argument; however, I believe that there is a more subtle idea at work. Movie making is by definition a creative and chaotic process—something that no one disputes. On the other hand, we have tended to treat software development as an engineering process—a potentially fatally flawed premise.

Certainly no one would argue that there does not exist a pervasive and influential engineering component within the software development discipline. Just as no one would argue that a movie's technological underpinnings are trivial. However, story telling is central to the movie making process, whereas the idea of software as a story telling vehicle has only recently surfaced as a meme that is being bandied about.

If you start with the premise that software is primarily story based, then the entire dynamic of the process shifts. More emphasis is placed on the creative relationships and conversations taking place among cast members, than on the technology required to render the idea to silicon. The human factor becomes front and center. Everything else becomes an important, but secondary, consideration.

Chapter 4: Movie Making & Software Development

The Story

In the movie business a great screenplay is everything. A great director, and great performances, will do little to improve a poor script. The result of a poor script is almost certainly guaranteed to be a bad movie. Even though writers often feel that they do not get the credit they deserve, everyone in Hollywood recognizes their importance.

It starts with the written word—it's a familiar dictum, but somehow it keeps getting forgotten along the way. No filmmaker, irrespective of his bag of tricks, can ever forget his commitment to the written word.

—Steven Spielberg, director/producer

You must create the story that your software will render in silicon, before you ever write a line of code, before you gather requirements, before creating a project plan. It must be written in quality prose and enjoyable to read. It must then be vetted with all the stakeholders that will help deliver it: management, product marketing, product development, trusted customers and peers, etc.

If you cannot sell the story to these stakeholders then real customers will never buy it, because it will never ship, or if it does, you might well wish that it hadn't. The story is the shared vision that the cast must believe in and are signing up to commit their creative energies toward. To be sure, it will evolve over time, but as it does the entire cast must be brought along with it.

Rarely have I seen the marketing story and the silicon story aligned in a manner required for successful execution. Organizations that behave as if these two stories are separate will continue to produce less than desirable results (b-movies), and will, more often than not, bring to market products that are complete disasters at the box office. The marketing story and the silicon story are one and the same, and must be treated as such from the project's onset.

The Studio

Let's explore the movie studio; in combination with a production house, as a metaphor for a business entity that organizes the production and distribution of stories-in-silicon. The creation of blockbuster stories-in-silicon is increasingly becoming a game of high stakes poker. The graphic below depicts how Hollywood has dealt with the disintegration of what was once a completely vertically integrated industry.

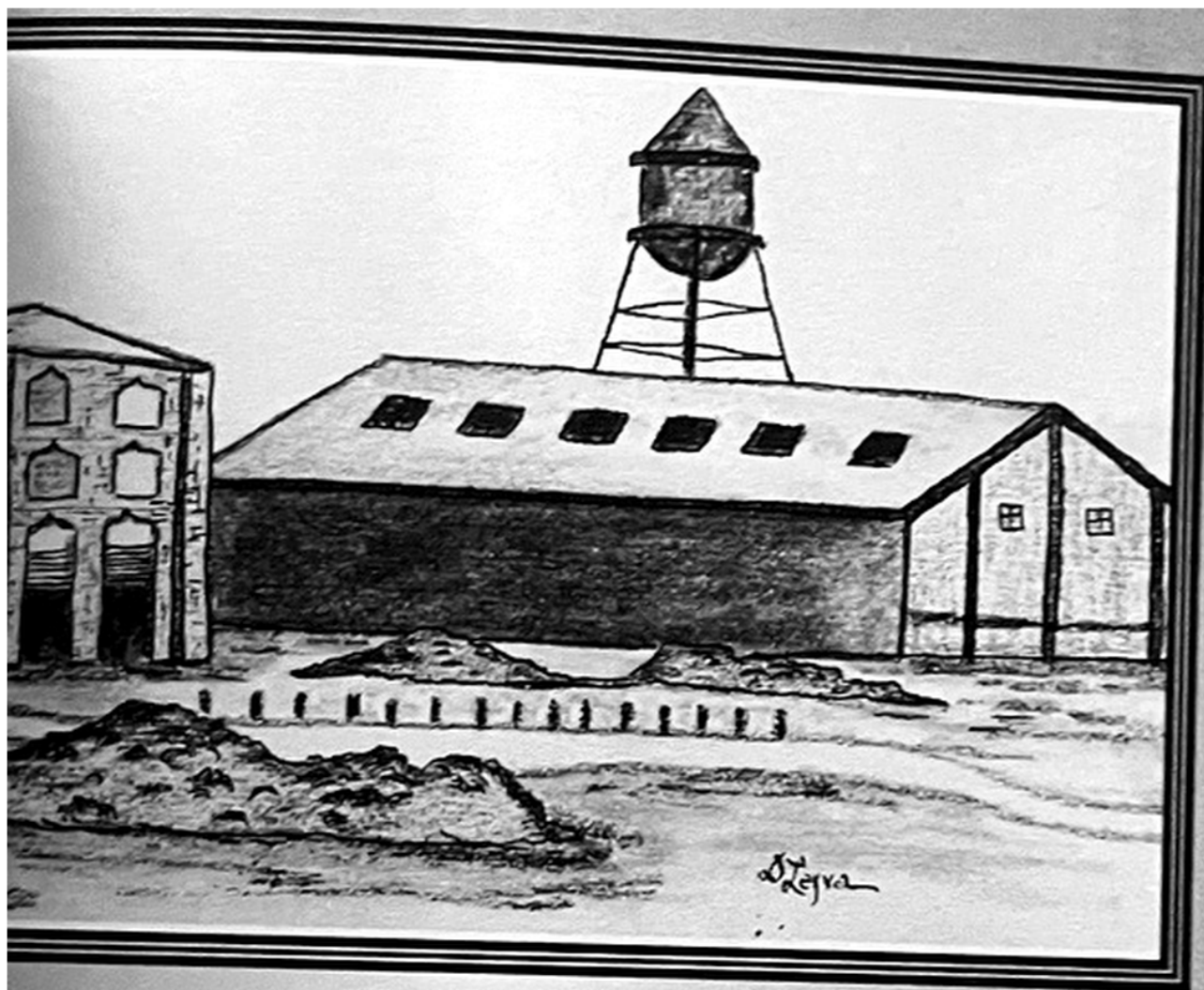
I believe that in the very near future the software industry will begin to migrate to a similar model. Why is that? On the surface there are many similarities that are already apparent. Increasingly many of the players that participate in the software market are freelancers. On any non-trivial project you are likely to find freelance developers, technical writers, graphics artists, and a host of other supporting cast members.

The major difference is that Hollywood has taken this model much further up the business and creative food chain. In Hollywood the producers and directors are highly influential hired guns

that often make or break the financial success of any given project. However, the studios and their investment partners own the properties, and because they ultimately supply the money and take the lion's share of risk, get the last word as to what ultimately gets produced.

Corporations that are in the business of producing stories-in-silicon would probably benefit enormously from adapting a somewhat analogous model. They do not necessarily need to be in the creation business, what they need to do is manage the packaging, branding and the distribution of the resulting properties. Although this model leaves them in control of the business, they will only be successful to the degree that they can build effective working relationships with the best available creative talent.

The major difference from this model and today's predominant software model is that most organizations continue to be vertically integrated with respect to software creation and production. The economic forces that have created the Free Agent Nation are likely to become even more powerful going forward. The result will be that often the best available creative talent will be more available outside of your organization than within it.



THE MISSING FACTORY

Chapter 5

The Missing Factory

Despite the evidence that I have presented in previous chapters pertaining to software development as a creative art form, there is an aspect of software development that is pure manufacturing. The immature, and at times non-existent software manufacturing processes that I encounter in organizations I consult with, constantly appalls me. I find this to be true in organizations of all sizes, despite the lip service that everyone pays to ISO 9000 and the Capability Maturity Model. Not that I am a big believer in heavyweight processes that force developers to produce more paper than work product, far from it. However, the cavalier manner in which most software is manufactured would make Henry Ford roll over in his grave.

The Road Less Traveled Starts Here

!!!Missing Text!!! Why a Factory? Because even though I am a big believer in software as art, once the inspiration has been codified in silicon, automation needs to kick in, both to ensure quality and to protect our intellectual assets. Most developers understand the basics of how things should be done, but they are often not given the time to establish adequate factory processes. In addition, they are rarely given funds to buy the tools that they require, despite widespread evidence, anecdotal and otherwise, that there are tools available in the marketplace that could dramatically improve their productivity. Or worse yet, the tools have been purchased but have been permanently relegated to “shelfware”, since no one has time to learn how to use them.

Why a Factory? Because even though I am a big believer in software as art, once the inspiration has been codified in silicon, automation needs to kick in, both to ensure quality and to protect our intellectual assets. Most developers understand the basics of how things should be done, but they are often not given the time to establish adequate factory processes. In addition, they are rarely given funds to buy the tools that they require, despite widespread evidence, anecdotal and otherwise, that there are tools available in the marketplace that could dramatically improve their productivity. Or worse yet, the tools have been purchased but have been permanently relegated to “shelfware”, since no one has time to learn how to use them.

Why a Factory?

Because even though I am a big believer in software as art, once the inspiration has been codified in silicon, automation needs to kick in, both to ensure quality and to protect our intellectual assets. Most developers understand the basics of how things should be done, but they are often not given the time to establish adequate factory processes. In addition, they are rarely given funds to buy the tools that they require, despite widespread evidence, anecdotal and otherwise, that there are tools available in the marketplace that could dramatically improve their productivity. Or worse yet, the tools have been purchased but have been permanently relegated to “shelfware”, since no one has time to learn how to use them.

Chapter 5: The Missing Factory

Assembly Line Tools and Processes

First of all let me start out by saying “Fuck ISO 9000 and the Capability Maturity Model”, I am not so much interested in attaining a state of automation nirvana as I am in ensuring that the organizations that I work with at least have the basics covered. Besides, I believe more in the common sense of developers than I do in a bunch of aging bureaucrats that haven’t done anything real in the last thirty years.

A Slight Detour

If you live in the valley you are probably rolling your eyes back in your head by now. I know that you find this shit boring. Even though y’all (raised in Nawlins remember) think you are God’s gift to silicon, most software is developed elsewhere, so fuck off, this chapter ain’t for you. Why don’t you go outside and polish your BMWs, or go to Starbucks and get a double shot of espresso, or hug a fucking tree? Everyone knows that the valley is full of pussies and ugly ass women—so I really don’t mind if you take offence.

Hey if you were really bad asses then you would go hack Bill’s security system or paint graffiti on Larry’s yacht, just for sport. Now these are no lightweight challenges. You don’t think that Bill is stupid enough to use his own weak shit on something as important as protecting Mamma Bill and Baby Bill? Nah, I’d be willing to bet that it is Linux and Java based security. As for Larry, well he’s got an in with the Lord so you won’t be touching that yacht anytime soon, talk about kick ass security.

In fact, I overheard Larry and the Lord having a conversation and the Lord told Larry that he is tired of all his trash talking and carrying on. The Lord said “Prove to me that you’re no fool, walk across Bill’s swimming pool”. Larry said, not only will I do it; I’ll do it on prime time television and rake in a billion greenbacks on your behalf. I will do this in your honor Lord, because even though Bill has more money, everyone knows that I have gotten laid more often, and I’m way fucking better looking.

The Lord watched in amazement as Larry walked across the pool and said, “That’s nice Larry, and I appreciate the billion greenbacks, I’m getting tired of watching these television evangelists milk the poor for money. But, I was just commenting to Scott that even though you are better looking than Bill, and you have gotten laid more often (not something to be too proud of since just about anybody in the known universe can make this claim), most of the innovation these days appears to be coming out of Redmond. Bill was blessed with far more imagination—last time I checked, the database was the last good idea you had.

Of course, Scott just lost that shit eating grin he’s always wearing, “Lord what do you mean by most of the innovation appears to be coming out of Redmond?” The Lord replied, “Scott, quit whining, you are still a box peddler, and a lesser evolved species. You missed the opportunity to ride the Java train to the next evolutionary plateau. You lack Larry’s good looks and Bill’s imagination. Just run along and be a nice little yuppie, I hear Jack’s got more time on his hands

these days so maybe you can squeeze in a few rounds.”

Getting Back On Track

OK, I must confess, I find this shit boring and painful too. I have never known a developer that gets excited about factory processes. It falls into the category of necessary evils, like performing paperwork after defecating and paying taxes. Well, I take that back, developers that sell products into this space often become religious about it, and rightfully so, they tend to do quite well, thank you very much.

If most factories weren't in such a state of disarray, this chapter would not be necessary. However, I feel compelled to discuss Factory 101 basics and add my name to the long list of people that the development community hates for pointing out things they know they should be doing, but often don't. If you happen to be the manager that has been tasked with factory building, then do not despair; the web contains all the raw materials you need to put a working factory together in fairly short order. The following topics are covered in no particular order. They by no means comprise an exhaustive list, God forbid I should ever have to compile such a beast of burden; exile in Siberia would be preferable.

I intend to provide high-level process insights while at the same time preventing this chapter from turning into a manual. Most product references are specific to the Microsoft universe since that is where most of my time has been spent in the last few years. However, there are certainly analogous products available from the Java/Unix/Linux camp.

Chapter 5: The Missing Factory

Source Code Management (SCM)

Managing source code is one of your most important job responsibilities as a developer, project lead, or manager. Anytime you have more than one developer working on a project then source code management, using a commercial grade SCM product, is a must. There are a number of relatively inexpensive (and some free) products available that cost should not be a consideration. In the Microsoft universe, Visual SourceSafe is part of Visual Studio and essentially free.

The Basics

There are two basic considerations when setting up a source code management process. Define a set of roles and responsibilities:

And define a set of environments that will be supported:

Each environment comprises a distinct code repository. Security access should be granted to specific environments based on an individual's role. A formal procedure with signoffs should be instituted for moving code between repositories. The movement of code between Development and Quality Assurance is usually where organizational standards are enforced (e.g. code reviews).

Naming Standards

Discussions concerning source code naming standards (e.g. for variables, methods, etc.) often stir up religious fervor among opposing factions. All of the leading commercial software providers have documentation available on their web sites with their recommended approaches clearly presented. The most important thing to do is select one of the recommended approaches and consistently adhere to it. There is absolutely no justification for rolling your own.

In Source Documentation

I am a big believer in the value of in source documentation, both at the module/class level and at the method level. Again, the exact template that you decide on is not nearly as important as getting developers to actually do it. The only feasible way I have found to encourage this practice is to convince developers that if they get in the habit of doing it as they go, it is not nearly as painful as they imagine. Adding in source documentation after the fact is extremely painful and almost never gets done.

Chapter 5: The Missing Factory

Incident Management

What is an incident? An “incident” can be a bug that has to be fixed, a feature that has to be implemented, a customer question that has to be answered, or any other important item that has to be tracked until it is dealt with. Associated with each incident is a general description of the issue, a priority, a person that it is assigned to, and a status. In addition, an incident may be associated with one or more contacts (e.g. customers) or files (e.g. documentation or source code), and finally, an incident is associated with a specific project.

In the Microsoft universe, a product—Visual Intercept by Elsinore Technologies—is best in class and so tightly integrated with the Visual Studio IDE that it is indistinguishable from one of Microsoft’s own offerings. An incident management product is the “glue” that holds the development process together and an absolute requirement for all but the most trivial of projects.

Effective management of source code and incidents, using world-class tools and processes, make up 80% of the critical must have factory components! Homegrown alternatives should no longer be tolerated as acceptable replacements.

Backup and Recovery

Backup and recovery represents a set of strategies that determine how quickly an organization can respond after a disaster strikes. A disaster can be anything from a hard disk failure on your one and only mission critical web server to an act of God (i.e. tornado, hurricane, flood, etc) that brings the site down. In order to minimize downtime many factors need to be considered:

Usually what drives the decision is cost versus the amount of downtime (i.e. otherwise known as pain) the organization can tolerate. It is my experience that recoverability can be improved dramatically (i.e. vis-à-vis your current state) simply by focusing on low cost, but effective, best practices. These best practices tend to focus on process issues requiring minimal incremental capital expenditures. Providing absolute recoverability requires increased complexity and cost and is usually not warranted for work performed by development groups. However, if you are responsible for Etrade’s or Amazon’s production site, then that is a completely different story.

Here is a list of common sense practices:

Once you have a working process, destroy shit on purpose from time to time to ensure that it is actually working. Obviously this should be done in a controlled manner, but it must be done. The most elegant backup and recovery procedures can develop insidious bugs that go undetected for long periods of time. The only way to prevent this from happening to you is to test it often.

Chapter 5: The Missing Factory

Tools of the Trade

I am always surprised by the fact that even great developers often do not insist that the organizations they work for purchase the necessary tools that would make them even better and more productive. Debuggers are of course part of everyone's toolbox but other tools such as memory corruption detectors, source code coverage, and performance profilers go virtually unused. Part of the reason for this is the time crunch that developers constantly face—the last thing they need is yet another learning curve to climb.

The way around this quandary is to designate one team member as the “Tool Smith.” This individual is responsible for learning the innards of each tool within the toolbox and disseminating this knowledge to his or her partners in crime.

Documentation

If there is anything that developers hate worse than documentation, I have yet to discover it. Most developers would prefer Chinese water torture to writing a document in a natural language. I have, through trial and error, hit on a strategy that I have used with considerable success.

Establish a process that outlines the level and quality of documentation that is required. Make it clear to the team that you do not expect a polished novel or even completely grammatically correct prose. What you want is a brain dump that will be used as input to the documentation assembly line. Develop a standard set of templates that remove formatting issues from being a discretionary consideration. Provide basic training related to screen capture utilities, MS Visio, MS PowerPoint and any other tools required to do the job. Hire professional technical writers to take this raw input and produce a professional deliverable.

The Shipping Department

By shipping department I mean everything required for getting the actual physical product out the door. This includes tasks such as:

And a host of other vitally important details that need to be tended to. My only recommendation in this space is to hire the youth. Hire really bright young individuals that can get excited about this kind of stuff because they perceive it to be a stepping-stone into the biz. Help them create a working process and then celebrate their contributions.

1-800-TechSupport

Technical support in this day and age when it is often difficult to find a human being to talk to when problems arise (and they always do) is an area that should be perceived as an opportunity to derive competitive advantage. Do not staff this function with your weakest players. I am of the opinion that everyone should be required to do time in support. Not only can it be a personally rewarding experience, it is often one of the best inter-personal educational opportunities that

exist for the geeks among us.

Parting Thoughts

Factories are judged by the quality of the products that they ship. This is not a monument building exercise. Develop and institute these processes and get on with the business at hand. If you implement the basics described in this chapter you will be ahead of most of your competition and equal to the rest. If you choose to take it to the next level then you are a sick individual and have way to much time on your hands. I would never hire you.

This chapter has focused on factory best practices with respect to software development. Many of these ideas are applicable to other domains wherein soft goods are produced (e.g. the publishing business). You should consider adopting those ideas that you find useful.



ARCHITECTURE

Chapter 6

Architecture

Architecture is the stuff of the wires. That buzz you hear is the sound of imagination in motion. It is not science or technology. Architecture is the sound of builders collaborating. It is the convergence of design, form, craft, media and software, the creation of n-dimensional living and breathing info-spaces. Wired communities linked via electronic boulevards into virtual cities of the final frontier. Architecture is art and meaning, without it the techno-babble makes no sense. The wires are whispering and if you listen closely you will hear the songs that they are singing. They are the songs of a new form of urban planning.

Software Architecture

What is software architecture? I am often asked this question whenever I tell people that my company is an “Internet consulting company that focuses on architecture.” I find it quite interesting that I get this question equally as often from technology people as from people outside the industry. My elevator pitch is not yet perfected so I find myself stumbling for a few seconds before saying something like “architecture concerns itself with making sure that the various parts of a complex system (e.g. an eCommerce site) work well together.” Invariably the response I get is “so your company does web pages?” To which I respond, “yeah, well, that is part of it. But we also make sure that the web server, the application server, the database, the front end, and any required middleware work in tandem.”

In other words the question is not an easy or straightforward one to answer. We have a 2000-year history of architecture as it relates to the built world, and still the general public has only a vague understanding of its first principles. In the distributed computing universe, at least with respect to anything that could be called computing for the masses, we have at most a twelve year history (circa 2002), therefore it should not come as a surprise that even many practitioners within the technology industry are also confused when the word is used. Why twelve years? Because prior to the client-server revolution, most, if not all, of the architectural considerations came pre-packaged in the box. Application developers working on IBM mainframes paid scant attention to architecture. There was no need. It was a done deal.

Little wonder that mainframes were extremely successful machines, despite the fact that they became everyone’s whipping boy. They allowed developers to get busy with the task at hand, instead of worrying about the myriad of details required to ensure that the underlying infrastructure and plumbing worked to everyone’s satisfaction. Client-server computing in the early 1990’s changed all that. Architectural concerns became front and center.

I would like to propose the following succinct description (i.e. as opposed to definition) of software architecture:

Software architecture relates to constructs required to ensure that the parts of a complex system work together in a coherent manner. These constructs must ensure that the system scales and that it continues to function even though individual components may fail.

While not defining software architecture, it nonetheless captures the essence of its meaning. More formal and rigorous definitions mostly add obtuseness without providing additional clarification. One of the “three amigos” provides the following definition:

Architecture is the set of significant decisions about the organization of a software system, the selection of structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaboration of these elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaboration and their composition. Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility, economic and technology constraints and trade-offs, and aesthetic concerns. [Booch 1999]

With all due respect to brother Booch, someone I have great admiration for, I think I will stick to my short description just the same.

Chapter 6: Architecture

So What?

If I am the CEO of a Fortune 500 corporation, or a senior partner in a distinguished law firm, or the president of a mid-sized construction company, why should I give a shit about architecture? I am busy running a business. Besides, I pay my technology people damn good money to worry about these issues. I will tell you why. It turns out that your software architecture is not only important for your computer systems, but going forward, it will become more and more important to your existing and future competitive advantages.

In a knowledge-based economy it is your software architecture that provides the underpinnings and instrumentation for your most valuable assets. It determines the extent to which you leverage your legacy systems and your capabilities to mine the value located therein. It drives the success or failure of your merger and acquisition strategies. It has a huge impact on whether your top talent stays with you or finds greener pastures with your competitors. It will either help contain costs or become a constant drain on the bottom line. Pay attention to your software architecture or update your fucking resume. Your company won't be a fun place to work over the next five years.

I am always amazed by the breadth and depth of knowledge that the generals, turned CNN commentators, display with respect to a broad range of topics: complex weapons systems, geo politics, domestic politics, world history, foreign cultures, clandestine organizations, etc. They also appear to be extremely techno-literate with respect to advanced communications and other enabling technologies. Why is that? Because in the past their careers and the outcome of wars depended on it. Your career and the result of current and future competitive wars require that you grow your denominator in areas that were previously left to underlings. You don't have to own the issue; you just have to become literate enough to ask the right questions.

This chapter will introduce what I consider to be the key business drivers related to architectural considerations in software. I will discuss the limited number of options available for the procurement of commercial infrastructure and plumbing. I will outline why commercial plumbing is a necessary, but not sufficient, component of your architectural strategy. I will describe the impact that architecture has on the DIP and therefore on whether your current project portfolio has any chance in hell of achieving the promised advantages. In short, this chapter frames the architectural discussion as a business problem.

Chapter 6: Architecture

Shack, House or Skyscraper?

If you are building a shack in the back yard so that Lassie can stay (mostly) dry when it rains, then you need not give much thought to architecture. You could probably slap a shack together in an afternoon (2 weeks for me) that would meet Lassie's requirements. On the other hand, if you are contemplating building your dream home, you better plan on setting aside countless hours for ruminating through Architectural Digest and visiting with your architect of choice.

In order to build your dream house you will need to give significant thought to the use of space, the size of the lot, the location of windows vis-à-vis sunrises and sunsets, the number of stories, the size, location, and layout of various rooms, and the million and one aesthetic details that always seem to drive men crazy. In addition, your house contains a number of complex subsystems that will require your attention:

to name a few. Clearly the architectural requirements of your dream house are an order of magnitude more than those of a shack.

If you are building a skyscraper then you need to concern yourself with these subsystems and many more. However, it is not the greater number of subsystems alone that adds to the increased magnitude of architectural complexity in skyscrapers. The subsystems required for a skyscraper are more complex and they need to scale. There is also an increased need for redundancy. In short, building a skyscraper is a couple orders of magnitude (at least) more complex than your dream house.

It is often difficult to make accurate comparisons across disciplines but let's see if we can't beat this analogy to death. A typical enterprise eCommerce site, supporting thousands of users, probably contains a degree of complexity akin to building a skyscraper, and can often cost as much when all is said and done (i.e. including the hidden costs that no one ever talks about). Building a business-web that includes key partners and suppliers is analogous in complexity to building a plaza of skyscrapers that interact in ways that, heretofore, have never been imagined. Such an undertaking is clearly a different beast altogether.

It would be unconscionable of a CEO to construct a skyscraper, or worse yet a plaza of skyscrapers, to house his employees and organization without being deeply involved in the process. Yet these same CEO's, and their boards, are often clueless about architectural decisions being made within their firms that are of far greater import than any possible configuration of bricks and mortar. You will live with the results of your software architecture investments for the next twenty years, and as valuable (and important) as the commercial plumbing offerings are (see below), choosing between them will ultimately be one of the easier architectural decisions you make.

We shape our buildings: thereafter they shape us.

—Sir Winston Churchill

Chapter 6: Architecture

Commercial Plumbing

The emergence of a robust set of plumbing and infrastructure offerings, delivering on the promise of binary reusability of software components, from Microsoft (i.e. .NET/COM+) and Sun Microsystems (Java 2 Enterprise Edition: J2EE) is driving unprecedented change in the computing landscape and threatens to revolutionize the entire software development and delivery process. The computing world is once again polarized between a set of competing standards that provide, for the most part, analogous (despite the hype) enabling technologies required to deliver next generation products.

There is a difference in the nature and probable outcome of this battle that distinguishes it considerably from previous encounters: each competing standard will continue to dominate its respective space without the likelihood of a winner take all scenario emerging any time soon. You have the entire software industry split in two, with Microsoft on one side and the Anything But Microsoft camp (ABM: Sun, Oracle, IBM, AOL, BEA, Novell, and a host of others) on the other.

Microsoft, despite its legal problems, will remain the single most important software company in the industry, and one of the most powerful corporations on the planet. However, this time the gorilla is surrounded by a number of powerful chimpanzees that are collectively rallying around Java and the J2EE specification and will not be denied their share of the market going forward.

What is at stake for their current and potential customers? Nothing less than making a strategic technology decision that will affect how they manufacture and deliver software for the next twenty years. Both offerings are robust and viable today, and will continue to improve as time goes on.

Although the domain of this strategic decision is technology, the rationale that should be used in making a choice should be based on an organization's business drivers and other operational considerations. There is not a compelling technological superiority either way that allows for clear differentiation. I will provide a brief introduction to the core concepts of commercial plumbing and elaborate on why, although incredibly important, it is not enough architecture to take you where you need to go.

Chapter 6: Architecture

A Set of Services

In general, I refer to commercial plumbing offerings as architectures (which they are), however their actual implementation is based on a set of services that enable a component-based distributed computing reality (see Components). The following services are provided by each of the approaches:

Both architectures act as the “glue” between components, allowing unrelated software constructs (i.e. other components) to connect and interact in meaningful ways. Each approach allows grouping of objects into distinct components so that applications consist of plug-n-play parts that provide specific functionality.

Chapter 6: Architecture

Market Drivers

What market forces drove the evolution of the component-based software development approach and what specific problems are these offerings attempting to address? The following four bullets help answer this question.

Does either of these architectures completely solve these problems? No. But they both go a long way toward addressing most of these issues and by doing so enable the software development community to take a major, perhaps revolutionary, step forward.

Chapter 6: Architecture

Microsoft Worldview

Microsoft has once again managed to deliver a powerful computing platform and at the same time confuse the shit out of the market. This is typical Microsoft. It is also partly just the nature of the beast. I will leave the technical details of .NET to one of the many excellent tomes available in the marketplace. Besides, describing .NET is not nearly as important as recognizing that if you live in the Microsoft universe then this is incredibly important technology. Learn it. Use it. Or risk being left behind.

.NET is Microsoft's competitive response to Java and J2EE. Microsoft has once again bet the farm. Microsoft's competitive response to Java and J2EE is bold. It is so bold that they weighed the consequences of pissing off a large number of their customers and decided to move forward anyway. They had no choice! Microsoft's COM based technology was long in the tooth. They rode this horse just about as far as it could go. Competing against Java and J2EE with COM was bound to be a losing proposition. Java and J2EE had too much momentum for Microsoft to be able to hold back the market with a quasi broken infrastructure offering. Their only hope was to invent their way out.

So what did they do? They decided that if you can't beat 'em, out innovate 'em. C# and .NET are as good, if not better, than Java and J2EE. Love Bill or hate Bill .NET is nothing short of a brilliant response! Microsoft has once again leveled the playing field. That said (as mentioned previously) there are no winner takes all scenarios this time. Both offerings will continue to be viable for the foreseeable future and beyond.

Chapter 6: Architecture

ABM Worldview

Since its introduction in May 1995, the Java platform was adopted quicker than any other new technology in computing history. All major computing platform vendors (with the exception of Microsoft) are integrating Java technology as a core component of their products. With the phenomenal rise of application server technology from industry heavy weights like Oracle, IBM, BEA Systems, Sun, Borland, and others, Java and the Java 2 Enterprise Edition (J2EE) platform have emerged as a competitive threat to Microsoft's industry dominance.

The Java platform is no longer a fringe technology, a curiosity, or a solution in search of a non-existent problem. It has found a permanent home in the middle tier of three tier architectures, and having done so, is challenging Microsoft's COM+, and now .NET, as the industry's dominant architecture. The meteoric rise of the Java platform caught Microsoft asleep at the wheel, permanently changing the computing landscape. As I have discussed previously, Microsoft has responded boldly. It took a solid shot to the jaw but has served notice that this fight is going the distance.

Java and J2EE are hardware and operating system agnostic and this provides broad appeal to many companies that would rather not be locked into what they consider to be proprietary standards. Microsoft is attempting to market .NET as an open standard, but the market remains wary of such claims. Bottom line is that Java and J2EE are here to stay.

Whence CORBA?

CORBA continues to evolve and mature. While CORBA has faded to a distant 3rd position in the architecture wars, it is by no means a dying initiative. On the contrary, CORBA will probably continue to evolve and prosper, along with the emerging Web Services specification, as the de facto standard for integrating COM+/.NET and J2EE. That said CORBA should be used only as glue technology and not as a primary standard.

A Business Decision

Despite the endless hype that has and will be unleashed upon corporate America, the architecture wars are quickly becoming passé. Choosing the appropriate architecture is a strategic technology decision that should be based on the business drivers of individual organizations as opposed to an overwhelming technological superiority of one of the two leading offerings.

The architectural decision should be based on the realities of the labor market, where your organization and industry fit into the application food chain, interoperability with other important technologies that are important to your mission, time-to-market constraints, and technical support issues. In short, make a decision and move on. The only thing of interest is what you decide to do on top of your commercial plumbing standard of choice.

Do not believe for a minute that either one of these “offerings” will solve your architectural problems. You are just getting started. If you ask your CIO “Hey, this Leyva idiot says that architecture is important. What are we doing about it?” and they respond with “We have it covered boss, we’ve selected J2EE, or we’ve selected .NET.” Fire the silly motherfucker on the spot (see The Only Architecture That Matters below).

Looks like the CIO unemployment situation just got worse.

Chapter 6: Architecture

Components

Components are software building blocks that are reusable. They plug into, and require, a runtime commercial plumbing infrastructure such as .NET or J2EE. Individual components are compatible with one or the other (i.e. .NET or J2EE) but not both. There are a number of software companies that generate revenue based on providing reusable components and nothing more. There are now third party market makers that facilitate transactions between component buyers and sellers (e.g. Flashline at www.flashline.com). In short, the software component market is maturing, however despite a maturing marketplace, many (perhaps most) IT organizations and plenty of ISV's (sad to say) are woefully behind the component reusability curve. Why is that? Why does it matter?

Buy versus Build

Like so many other aspects of software development, the reasons that components and reusability have not taken off as dramatically as expected, despite the progress, are largely rooted in historical, organizational, and sociological (biological?) considerations. Developers and other technologists are natural-born tinkerers. They like to build stuff. They like to use the stuff they build. They have an innate proclivity to use their own stuff.

IT management, on the other hand, recognizes the importance of components in the abstract, but becomes totally caught up in the selection process. They become obsessed with finding the best components where, in the majority of cases, damn good ones will do. Either that or they refuse to give up the cold hard cash required to purchase them, the epitome of being penny wise and dollar foolish.

The change to a software component mindset is happening ever so slowly. Most organizations would benefit greatly from purchasing commodity component technology and moving up the abstraction food chain to design issues. The design domain is where significant differentiation is likely to occur going forward.

In the consumer electronics space there is a significant trend toward outsourced manufacturing. Microsoft designed the Xbox but Flextronics manufactures it. Microsoft rightly decided that it was not in the consumer electronics manufacturing business, and did not need to be.

This is not an argument for outsourcing your entire software manufacturing operations; in the near term commercial components provide a happy middle ground. In the longer term, variations on the software manufacturing outsourcing theme will have to be re-visited based on the same economic drivers as consumer electronics (i.e. that differentiation will become completely design centric).

Reducing the DIP

Component based development dramatically reduces outstanding decisions in progress. Once you buy or build a robust set of components you use them with confidence over and over. This strategy can provide up to 60% of any new product pre-built. Using pre-built components has revolutionized a number of industries including retail-housing construction. The exact same idea is now being applied to software development.

It is obvious that starting with a 60% pre-built solution yields significant benefits. But the upside to this strategy is more subtle. Not only do you get the benefit of using proven parts, you also win by not having to revisit these issues from a decision-making perspective. In short, a 60% pre-built solution probably yields more like a 70% reduction in product cycle time.

Chapter 6: Architecture

The Only Architecture That Matters

The graphic below is an over-simplified, yet representative view, of the emerging software stacks (i.e. layers) concept. The premise is that layers lower in the stack have no knowledge of layers above them. This approach embodies the “separation of concerns” abstraction. The top three layers represent where you add value and where you are likely to leverage enabling technologies for competitive advantage.

XYZ Plumbing

The only architecture that matters resides within this layer. It embodies the XYX architectural framework and represents how we do architecture around here. This layer should provide an architectural abstraction of your commercial plumbing layer, not because you will be replacing it anytime soon (you won't), but because it simplifies the layers above it. How? Largely by codifying decisions related to the base services that commercial plumbing provides:

and so on. In a sense this layer rationalizes the commercial plumbing layer in a manner that makes sense for your organization. If this layer is missing then your development teams will need to make thousands of design decisions every time you require a new wiz-bang app. The impact on the DIP should be clear by now. The more decisions you need to make the longer development will take, often by orders of magnitude. In addition, you will require a small army to support disparate implementations.

It should be obvious that commercial plumbing vendors, or anyone else, will not be able to provide this layer completely off the shelf. However, industrial strength component parts are available and should be leveraged. This layer is almost always ignored as corporations rush to build apps. The reasons for this are obvious. XYZ Plumbing will not result in any visible competitive advantage in the short run (i.e. the next two financial quarters). In fact, its implementation could slightly delay the delivery of XYZ's next killer app.

Look into the future five years and it will become apparent which organizations built this layer. It will largely determine the winners from the losers for all the reasons covered in the introduction to this chapter.

Chapter 6: Architecture

XYZ Applications

Assuming that you have the first four layers of the stack under control, killer apps will become more straightforward to build. Straightforward does not imply easy. We are a long ways from easy, however apps will become both more powerful and delivered in previously unheard of timeframes. The focus will shift to design issues and domain expertise. It will shift from construction to creativity.

Where will these apps come from? Look around your organization at things you already do well and look to leverage that expertise in exciting new directions. Your intellectual capital lives in the wetware of your employees, in your legacy systems, and in the myths and legends of your enterprise. You are not lacking in raw materials. What's missing is the know-how required to package and deliver these raw materials in creative ways.

XYZ Portal

By portal I mean the view that your info-spaces provide to users, both internal and external to the organization. The XYZ Portal layer must be designed in a manner that supports display device independence. This layer is the only layer that should interact with the user. Design is central to all three layers that you control but it is hypercritical to the portal layer. Usability, navigation and branding all converge here.

You cannot underestimate the importance of the portal layer. A wonderfully designed and aesthetically pleasing and scalable architecture will amount to nothing if your portal layer is lacking utility, quality copy, easy understandable navigation, pleasing graphics and a host of other features that consumers are now accustomed to vis-à-vis their experiences with some of the best sites on the web.

Building Bridges

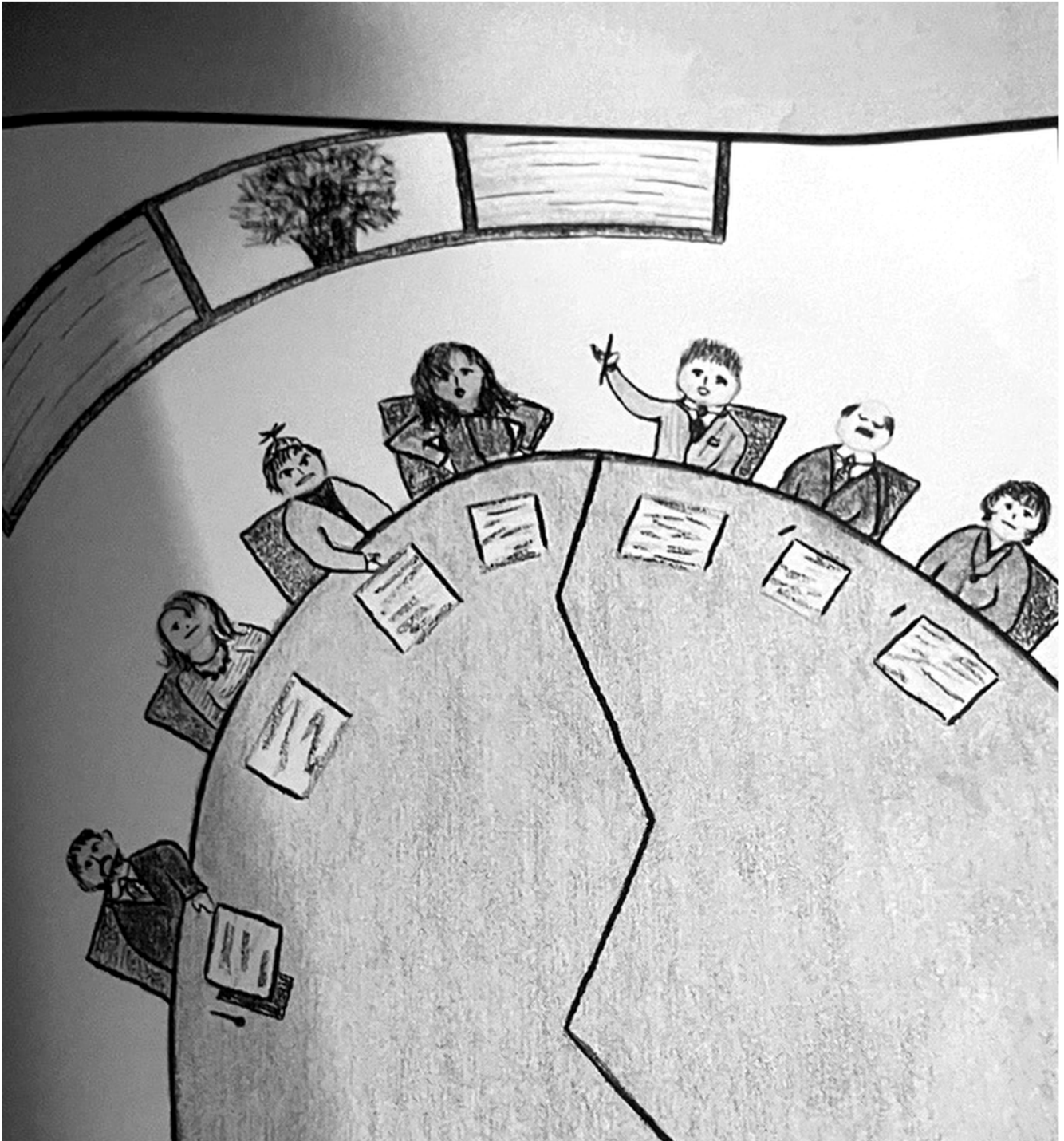
The new urban planning concerns itself with building bridges to the future. It is a monumental task equivalent to re-building a ship at sea. There are no easy answers and no silver bullets. If you hear the song of the wires then you have no choice other than collaboration with the builders. Many are the ships that will be lost at sea, dashed upon the rocks of limited vision. The new urban planning is all about the built world of internal spaces. Software architecture underpins and instruments constructs that we have previously imagined but have been unable to realize.

Collaboration is like a love affair, it veers between admiration, desire, rejection, anxiety, contempt, rage, powerlessness, misunderstanding, not getting what you want but maybe coming nearer to knowing what you thought that might be.

—Katherine Clarke, 1999

Success awaits the master collaborators, and only them. Software architecture requires a culture of collaboration. It requires conversations that incorporate and embody many disciplines and the

return of the agora. Everything converges on the centrality of human dialog. Components and commercial plumbing are building materials. In the end they will only serve to mock us if we cannot find our humanity in the songs of the wires.



TALENT WARS, TEAMS & eCULTURE

Chapter 7

Talent Wars, E-Teams and E-Cultures

The guys in the black hats always come out when times are hard, this time to pontificate on why the Free Agent Nation and The New Economy are nothing more than myths invented by an army of pseudo-intellectuals who profit from promulgating this sort of happy horse shit. The irony is that, despite the dot-com implosion, the economy and the dynamics of labor markets have both changed permanently and irrevocably. The much-publicized chasm between the old industrial economy and the new knowledge-based economy is not only real, but will continue to grow in importance, and in a dramatic fashion, for the foreseeable future.

The Internet and the New Economy, while not one and the same, are certainly part of the same positive feedback loop that, along with globalization in its myriad manifestations, are pushing world economies into increasingly un-chartered waters. Bill Gates has been quoted saying that “People are overestimating the importance of the Internet in the next five years, and underestimating its importance in the next twenty.” This statement, taken out of context and on face value, is either clearly wrong, or disingenuous on Bill’s part.

Here’s why. In the next five years the Talent Wars are likely to increase in ferocity and winners will acquire the assets necessary to dominate their respective markets for the next fifty years. We are at an inflection point of immense proportions. This is the kind of discontinuous change that occurs once every thousand years or so. Of the three factors of production (land, labor and capital), labor, in all its incarnations, is about to emerge in a manner that will permanently establish its dominant position. Over time you will see smart labor leveraging the new medium in ways previously unimagined, all to the upside of both the top and bottom lines.

Just as smart software has invaded every nook and cranny of corporate business processes from the plant floor to the executive suite, smart labor will begin to make its presence felt as the key differentiator in industry after industry. So while this chapter focuses on eTeams and eCultures within the software space, it will soon become evident that its applicability is much broader in scope, as corporations struggle with the creative imperative in all aspects of their value chains. As Manuel Castells notes in *The Rise Of The Network Society*:

Toward the end of the second millennium of the Christian Era several events of historical significance have transformed the social landscape of human life. A technological revolution, centered around information technologies, is reshaping, at accelerated pace, the material basis of society. Economies throughout the world have become globally interdependent, introducing a new form of relationship between, state, and society, in a system of variable geometry.

Though Mr. Castells certainly does not share my views on the importance of labor, he does an eloquent job of describing, in vivid detail; the general social and economic conditions that place smart labor in a position of prominence in my worldview.

Chapter 7: Talent Wars, E-Teams and E-Cultures

The Mission

Almost everything, non-trivial creatively, that happens in the business world, and in many other walks of life, is the result of a project. Projects have a well-defined beginning, middle and end. They always require a project plan. The plan not only defines the stages of the project but also drives its execution. I prefer the word mission because it implies motion and carries a military connotation.

Signing Up

You usually either sign up for the mission or somehow, through economic pressure, or the threat of bodily harm, are volunteered. However, signing up, whether for a military mission or the creation of a commercial software offering, is, for me, a much deeper commitment than is apparent on the surface. It means that you will do almost anything within your power to achieve the end game.

In reference to a software project, signing up does not necessarily imply consistently working 60 hours a week for the duration of the mission. However, it does imply that you will go above and beyond the call of duty, which often entails some overtime, and/or other personal sacrifices at critical junctures. Signing up is a binary proposition, you're either on board or you're not. There is no acceptable middle ground. It is a contract that is signed with the rest of the team members. It is mutually binding and cannot be broken without putting the mission at risk.

Getting all team members signed up is the responsibility of the mission's Natural Leader (see Director Pattern in Chapter 3). On projects where I have assumed this role, I always look for opportunities to shoot on sight any team members caught sitting on the fence. They usually die a very slow and painful death, often from multiple wounds. I would rather reduce headcount than risk the mission by having to carry wounded combatants iteration through iteration.

No One Gets Left Behind (the WIIFM)

While Signing Up ensures that everyone on the project commits to everyone else by signing the same mutually binding contract, the Natural Leader also signs a personalized contract with each individual. This contract is a promise that individual, and specific, benefits will accrue to each (and every) team member in return for his or her effort during the mission. It answers the question "What's in it for me?" This contract is implicit between the Natural Leader and an individual team member and is not shared with other mission comrades.

Would you persuade, speak of interest not of reason.

—Benjamin Franklin

I recently had the pleasure of seeing one of the best war movies of all time, *Black Hawk Down*. Based on the true story of a week in the life of America's finest on a mission in Somalia. The movie's powerful punch is delivered not by outstanding performances from Hollywood's elite,

but by the graphic representation of the brutality of war, and the poignant relationships that are manifested in the middle of life and death situations. The motto of Delta Force is, “No One Gets Left Behind,” and I guarantee you, if the film is to be taken at face value, this is no gung ho marketing campaign. It is a mutually binding contract that is honored at all costs.

Notice that all this contract talk does not involve an organizational abstraction outside of the project team. It is widely known that soldiers in the heat of battle fight harder, and often until no one is left standing, not for God or country, but because of the personal relationships they have with those in nearby fox holes. It is the shared experience of the mission that brings out the best in us.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Natural Leaders

Natural Leaders own the mission. By definition there is only one Natural Leader for any given mission. They can coexist with a leader that derives authority from position power, but are obviously more effective when position power and natural leadership are aligned. Natural Leaders create team magic. Magic is not a by-product of serendipity. It is clear that renowned movie directors do not rely on serendipity to create the cast chemistry required for a great movie. The creation of cast chemistry is premeditated.

Good fellows are a dime a dozen, but an aggressive leader is priceless.

—Red Blaik

How do Natural Leaders do what they do? Does it require charm, superior social skills, above average intelligence, empathy and love of humanity? Well the truth is that no one knows for sure. Probably a Natural Leader possesses all the aforementioned qualities and still others that remain un-named. The nature versus nurture argument is relevant, and like so many other cases where this argument is raised, the only plausible answer is that natural leadership requires some combination of both. That said, it is clear that leadership skills can be improved with practice and grow with experience.

It must be noted that leadership skills are not immutable. Change the mission context in a dramatic fashion and/or within a different domain and you are unlikely to see the same Natural Leaders emerge. Leadership is often fleeting. It is extremely context sensitive and occurs within certain space and time boundaries. However, that said, within a given context and domain, leadership can certainly be maintained across a relatively large number of missions.

As previously mentioned, there are no scientific principles of leadership, and perhaps there never will be; yet there are some rules of thumb that have worked for me and I will share them with you. These always seem to apply, despite the composition of the current team and the context of the current mission.

Leading the Congregation

Every team member is an essential part of the team and brings unique qualities and experiences to bear in their assigned role. This is an axiom of team dynamics, however it is pure abstraction and is only relevant if each team member is made to feel unique. One of the Natural Leader's most important goals is to develop this sense of uniqueness through quiet praise and various other subtle techniques.

One not so subtle technique is to create forums wherein individual team members get to shine in front of their peers. I have used this technique with great success and go out of my way to look for opportunities that fit a particular individual. This is often most beneficial for journeyman and junior team members, who otherwise might not get an opportunity to lead the congregation. Can I get a witness?

Chapter 7: Talent Wars, E-Teams and E-Cultures

Screwing the Pooch - Whose Team is this Anyway?

High performance teams, jelled teams, killer teams or in short, great teams, require a considerable amount of energy to build and sustain. However, their complete and utter destruction can happen in a matter of hours, or less. This happens most often when outside forces, not understanding the team's dynamic, impose constraints or demands that make the contracts that took so long to negotiate, unenforceable.

The quickest way to ensure the team's destruction is to change the Natural Leader, either voluntarily or involuntarily, without any preparation or consideration for who will assume this role. Organizations must learn to have a better ear for team dynamics. In many ways the Natural Leader embodies the team. He or she created it, nourished it, and maintained and enforced the contracts. They often command more loyalty from the troops than some bullshit organizational abstraction that won't hesitate to downsize them out of existence at the first sign that the quarterly numbers are in jeopardy.

Fuck with the dynamics of a successful team, the one working on the latest iteration of your killer app revenue machine, and you have just screwed the pooch royally. Team implosions in the software industry are legend. One minute the magic is so palpable that you can breathe it like pure oxygen, the next minute the fire you started consumes it and you are left choking to death. Now you see it and now you don't.

I will pay more for the ability to deal with people than for any ability under the sun.

—John D. Rockefeller

Chapter 7: Talent Wars, E-Teams and E-Cultures

The Any Good Manager Fallacy

A good manager can manage anything right? Take for example Jack Welch (i.e. former CEO of General Electric), who is universally renown for his management skills. Can you conceive of a business or government project that Jack would not manage effectively? I can. Jack is a damn good manager, but he probably would have done a poor job of managing The Manhattan Project (responsible for building the first atomic bomb). Why? Jack is a world-class executive but not a world-class theoretical physicist. He would have had a hell of a time trying to discern a great atomic bomb design from a disastrous one. And even if this feat were remotely possible, we certainly would have lost the war by the time Jack made a reasonable determination.

If you are going to manage, let alone lead, any non-trivial software development project (and most trivial ones for that matter) you better be an extremely knowledgeable and respected member of the team. Otherwise the geeks are going to chew you up and spit you out. Design sessions will turn into popularity or techno-pissing contests, neither of which is likely to produce great results.

No, Jack Welch, being a world-class chief executive, would have recognized immediately that he was not the right man for the job, and would have, in short order, gone out and hired the best available talent that money could buy. It seems obvious in this example, but over and over again I have seen individuals put into positions of management or leadership, where they have, at best, a rudimentary understanding of the underlying technologies and/or domain. As if charm, good looks, or social graces were enough to get the job done.

This is not to say that being a technically competent individual with good managerial skills is enough either, since I have already expounded at length on the importance of leadership. But if you have these skills, at least you are qualified to be in the game!

Chapter 7: Talent Wars, E-Teams and E-Cultures

For the Love of the Game

Most people, at the top of their profession, derive some intrinsic value from their work. It is probably one of the reasons they excel in their chosen vocation. Such a calling usually satisfies significantly more than their financial needs. They derive an extreme sense of satisfaction, and at times, inspiration from the work itself. Ernest Newman noted:

The great composer does not set to work because he is inspired, but becomes inspired because he is working. Beethoven, Wagner, Bach and Mozart settled down day after day to the job in hand with as much regularity as an accountant settles down each day to his figures. They didn't waste time waiting for inspiration.

All the truly exceptional developers that I have had the pleasure of working with displayed an intense passion for their craft. They got off on the work. Just like exceptional musicians get off on their music. Passion for your craft is a pre-requisite for greatness. If you want to hire the best of the best, learn to look for a certain fire in their bellies and sparkle in their eyes.

Chapter 7: Talent Wars, E-Teams and E-Cultures

The Economic Unit of Value Creation

Successful projects drive competitive advantage. The creativity that impacts activities of the value chain yielding marketplace advantages is almost always unleashed within the context of a project. Therefore, it follows that you want to staff your mission critical projects with the best available talent you can find, wherever you can find it.

Mission critical projects should define and guide your employment practices. Much of the talent you would love to hire would probably want to work on your mission critical project, given that you are willing to pay them what they're worth, however, they may not be as interested in a stewardship role for the next five years, once the project's objectives have been met.

In other words, there are people who love to continuously create, and there are those that create and then love to nurse their creations. One is not necessarily better than the other, but they are different. In order to be successful you must learn to recognize the individual differences and then hire world-class players for each category. Mission critical projects will only yield sustainable results if you are astute enough to hire the right combination of both.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Don't forget the "e"

The title of this chapter refers to “eTeams” and “eCultures” and the “e” in this context is not superfluous pseudo-branding on my part. The “e” represents the power of the Internet as a medium to transform business, as we know it. It is not eCommerce (i.e. the ability to do transactions) of whatever flavor (e.g. B2B, B2C, B2G, etc.) that will drive the transformation.

No, what will drive transformation is the way in which the medium will improve conversations that take place between human beings. Improved human dialog is the change agent. Everything else is a distant secondary consideration. eTeams and eCultures are organizational abstractions that understand the marketplace as an asynchronous and multi-threaded series of conversations, and include conversational mechanisms in their products and services.

A few thousand years ago there was a marketplace. Never mind where. Traders returned from far seas with spices, silks, and precious, magical stones. Caravans arrived across burning deserts bringing dates and figs, snakes, parrots, monkeys, strange music, stranger tales. The marketplace was the heart of the city, the kernel, the hub, the omphalos. Like past and future, it stood at the crossroads. People woke early and went there for coffee and vegetables, eggs and wine, for pots and carpets, rings and necklaces, for toys and sweets, for love, for rope, for soap, for wagons and carts, for bleating goats and evil-tempered camels. They went there to look and listen and to marvel, to buy and be amused. But mostly they went to meet each other. And to talk.

Yes, markets are conversations. The Cluetrain Manifesto has propelled this meme on the world’s stage with astounding velocity. However, very few people actually grok the implications, and fewer still are leveraging the idea as a way of doing business. Enormous opportunities await any organization that finds its voice, and through it learns to have an ongoing conversation with its customers, employees, and other stakeholders.

Chapter 7: Talent Wars, E-Teams and E-Cultures

The Environment

During the dot.com craze, companies thought they could instantly manufacture a corporate software culture by having a fully stocked kitchen, game rooms, casual dress code, and mandatory corporate fun activities. All of these culture creation props, for most of these companies, were as empty as their bank accounts.

These perks are obviously nice to have, including a physical work environment that is conducive to creativity and personal productivity, but they are no substitute for a sincere corporate culture. The burden of consistently working 60-80 hour weeks for months at a time, and spending precious little time with those you love, will not be lessened by ten free diet cokes a day.

Shared Values

It is clear that a sustainable corporate culture must be built on a foundation of open communications and a shared set of core values, which include: honesty, integrity, respect, diversity, and profitability. Your shared belief system must permeate the entire organization and manifest itself in the day-to-day operations of the business. This shared belief system should be self-evident and should contribute to your employees' pursuit of life, liberty, and happiness (Jose can you see?).

As corny as a reference to the Declaration of Independence may sound, I believe in it wholeheartedly. One of the problems I have with most organizations is that they refuse, or are incapable of, answering the call to greatness. Their mission statements all seem to reference the same bullshit clichés that make most employees cringe. If you believe in something strongly then state it without equivocation.

Talking to Customers

Whose responsibility is it to talk to customers? Are these conversations solely the purview of sales, marketing, and senior management? Can your rank and file pick up the phone and have a conversation with your biggest customer's CEO? Why not? Conversations like these are taking place all the time online and there is nothing you can do to prevent them. Forget the Nazi mentality of attempting to control the corporate propaganda machine; it is totally out of control and becoming more so every day.

The very thought that marketplace conversations should be controlled is based on a premise of fear and deception, of talking at your customers instead of with them. Let the conversations flow and you will be amazed by how much value can be derived from having all ears listening to the dialog.

OK I see most of the sales and marketing executives squirming, and perhaps I am stepping out on a limb here. You can all no doubt recall examples wherein an employee, not authorized to speak on behalf of the organization, said something that was clearly inappropriate and you quickly

entered damage control mode. The probability of this sort of thing happening is guaranteed to increase were you to follow my advice.

So? If you are having an honest conversation with the marketplace, these things will take care of themselves. The marketplace intelligence gathered will increase by orders of magnitude. Who said there would not be bumps in the road? Get over it. Shit happens and will continue to happen, the magic lives in how you respond.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Us Versus Them

The business press easily killed a good-sized forest with all the print space that has been dedicated to the importance of cross-functional teams and communications. Yet, despite this, organizational silos exist everywhere you look. Why do these walls remain when there is universal agreement that they significantly contribute to organizational dysfunction? Obviously, as previously mentioned, the walls remain because certain stakeholders have a perceived vested interest in maintaining them.

One way to begin tearing down the walls is to eliminate physical barriers that exist between various departments. Have engineers and sales people intermixed in adjacent cubes or offices. While you are at it, throw in a person from purchasing and human resources as well. The daily communal contact will enhance cross-functional communications. Without encouragement from anyone, neighbors will begin to break bread, and perhaps drink a beer together after work.

Another, perhaps better, way to improve cross-functional communications is to encourage Natural Leaders, and individual team members, to open dialogs with all business functions required to accomplish the mission. Oftentimes line executives prevent cross-functional communication because of their own prejudices, fears, and insecurities. Learn to maneuver around these idiots and watch as the communications process improves dramatically.

Chapter 7: Talent Wars, E-Teams and E-Cultures

The Furniture Police

This sub-title is borrowed from the authors that made it famous, Tom DeMarco and Timothy Lister, in their still classic book entitled *Peopleware: Productive Projects and Teams*. There is not a better description anywhere of the basics required for a healthy software office environment. This book has become the Bible of workspace design for some of the world's leading commercial software providers.

Suppose that in addition to your present duties, you were made responsible for space and services for your people. You would have to decide on the kind of workplace for each person, and the amount of space and expense to be allocated. How would you go about it? You'd probably want to study the ways in which people use their space, the amount of table space required, and the numbers of hours in a day spent working alone, working with one other person, and so forth. You'd also study the impact of noise on people's effectiveness. After all, your folks are intellect workers—they need to have their brains in gear to do their work, and noise does affect their ability to concentrate.

Not only do DeMarco and Lister get at the essence of the problem, they provide solutions backed up by hard research data. Despite the fact that this book was first published circa 1987, I continue to be astounded by how little progress corporate America has made toward providing healthy workspaces.

You cannot create an information age company with an industrial age mind.

If you have been tasked with workspace design, rush out and get a copy of this book for all stakeholders participating in the decision making process.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Talent Connoisseurs

Our people are our greatest asset. We should all agree to abolish this statement from the business lexicon, because for 99.9% of the companies that I have consulted with (unfortunately) this statement is empty rhetoric and corporate bullshit of the worst kind. Most companies do not treat their employees as if they were the corporate crown jewels, and everybody knows it. Cut the crap unless you can deliver the goods.

Everything else being equal (and of course they never are), the companies with the best teams will win more often than not. It follows then that you require damn good coaches and talent scouts to sustain an advantage. You better pay these folks well because they will be (and are) in high demand. Recognizing quality talent is tricky business, and if you rely on bullshit Ivy League pedigree nonsense as your principal criteria, one of your competitors is likely to wind up with all the golden needles in the haystack.

The Recruiting Buzz

When I am brought in to help an organization build a team, the first question I ask is “What is your recruiting buzz?” Of course, I always get the expected response, blank stares followed by “What are you talking about?” In a labor market where quality talent is scarce, they are in the driver’s seat, and are usually very selective when choosing new assignments—not unlike the best actors and directors in Hollywood.

Why would the best talent want to work for your company, and on this specific project? Your recruiting buzz answers this question. It is the story told to candidates in order to get them excited about your mission. Oftentimes companies are engaged in very cool and exciting projects and do not know how to milk this for all it is worth during the recruitment process. Don’t get me wrong; this is definitely not about selling a story that will vaporize as soon as the newly signed up candidate walks through the door. It is about presenting what your project has to offer in the best possible light. Without a great recruiting buzz you are unlikely to win your fair share of the best candidates. Build it. Practice it. And use the shit out of it.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Preventing Burnout

Technology workers, because of the long hours, the relentless push of new technologies, the constant pressure to hit milestones, and the chaotic nature of the technology labor market are highly susceptible to burnout. I have argued throughout this text that software development is a creative process, similar to making movies and other art forms. Do the best actors and directors make movie after movie without taking time off in between to recharge their creative batteries? Usually not without paying a significant price. Creativity is a high-energy work. It is next to impossible to create when you are too tired to see straight.

Burnout is reaching epidemic proportions among North American workers today. It's not so much that something has gone wrong with us but rather that there have been fundamental changes in the workplace and the nature of our jobs. The workplace today is a cold, hostile, demanding environment, both economically and psychologically. People are emotionally, physically, and spiritually exhausted. The daily demands of the job, the family, and everything in between erode their energy and enthusiasm. The joy of success and the thrill of achievement are more and more difficult to attain.

Making developers work on one mission critical project after another, without the distraction of other productive, but less demanding activities in between, is almost guaranteed to produce burnout in a dangerously high number of individuals. I have known developers that leave companies; even ones that they like for the most part, just to break the monotony of endless creativity on demand cycles.

Breaking this cycle does not always mean time off from work. There are numerous extremely productive economic activities that will achieve this end, including: researching emerging technologies, attending conferences to learn more about the industry and key competitors, writing the article for Wired Magazine that they have been telling you about, and getting the training they have been neglecting, etc.

If you want to build and sustain high performance teams then be attuned to the emotional, psychological, and spiritual well being of every team member, not the least of which is you. Many people in our industry experience severe cases of burnout from time to time but are reluctant to discuss it. Even with the widespread acceptance of anti-depressant medications, openly discussing burnout is still viewed by many as an admission of weakness.

One way to remedy this situation is for the elders of the industry to begin openly sharing their personal experiences. Let us be clear on one point, for all the perceived glamour of software careers, they can be, and almost always are, brutally demanding. Organizations that learn to manage these issues in sincere and creative ways are likely to have employees that are their greatest asset, and are treated as such.

Chapter 7: Talent Wars, E-Teams and E-Cultures

Human Resources as a Competitive Advantage

Historically HR has the dubious honor of being the butt of more departmental jokes than any other business function. HR was perceived as being too bureaucratic and controlling. They are criticized for an obsession with paperwork and viewing their principal role as employee lawsuit prevention. In other words, they are perceived to be a necessary evil, but essentially reactionary and powerless.

In enlightened organizations this is all about to change. Since much of the drudgery of HR functions is being outsourced or automated, smart HR organizations within fast companies will begin to deliver strategic value with a focus on organizational competitiveness and the bottom line. They are moving just as fast as they can to avoid the fate of the Irish elk.

Through many generations, the Irish elk became the victim of an evolutionary quirk that led to its demise. Each year its antlers grew larger, eventually becoming so heavy that it was unable to lift its head and forage for food. It became weaker and weaker and eventually starved to death. The antlers that had helped the elk survive in the first place became exaggerated over time, leading to its extinction.

If you accept the premise that the outcome of talent wars will determine the winners and the losers going forward, then it follows, just as surely as night follows day, that HR is likely to rise in prominence. However this will not be the HR that we have all joked about, but a transformed HR that will have everything to do with competitiveness and very little to do with administration. It will be an HR made of crazies in search of their own kind.

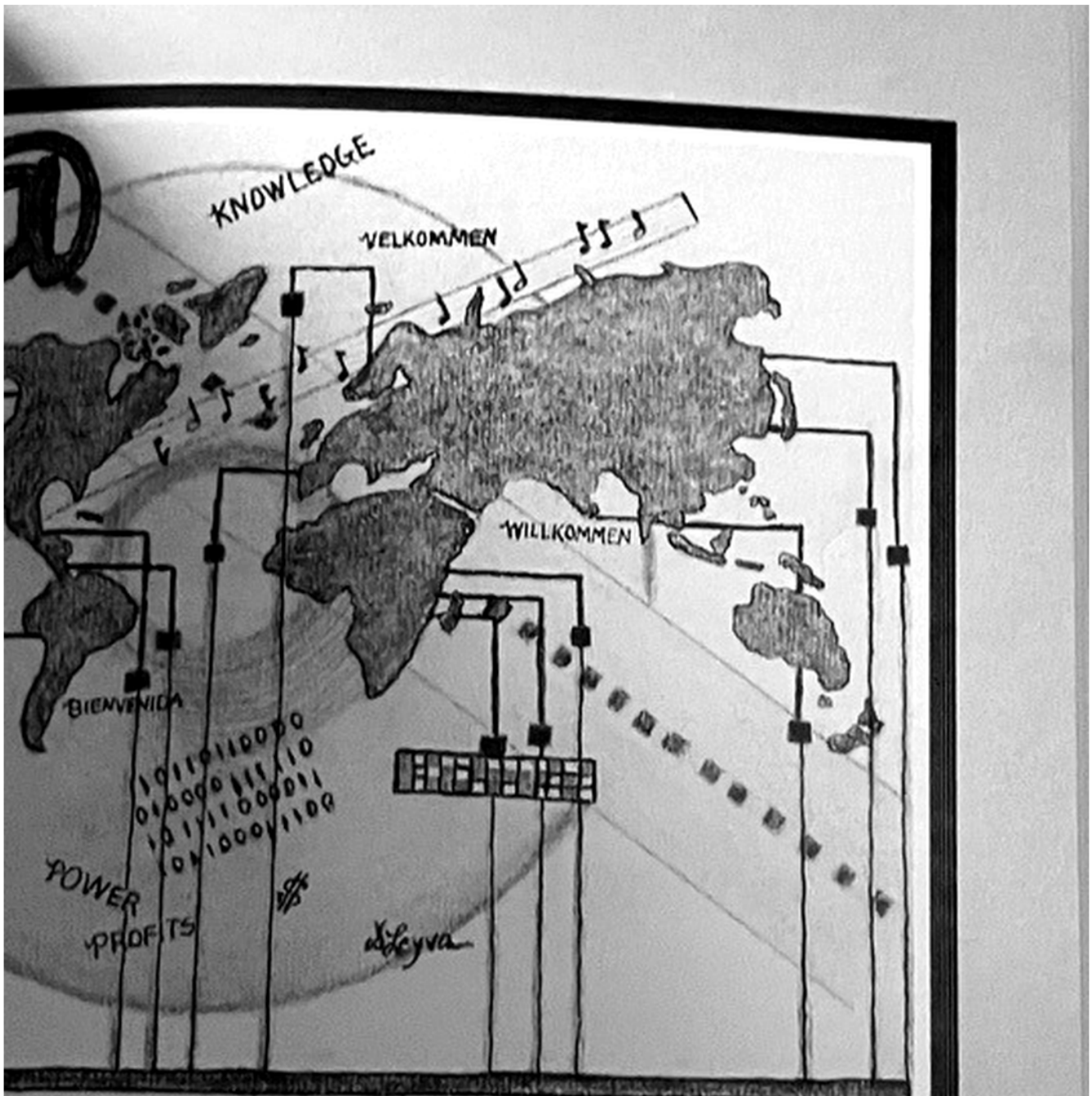
You say you don't want emotional, volatile, and unpredictable, just imaginative? Sorry they only come in a package...I can offer you a dedicated, loyal, honest, realistic, knowledgeable package, but the imagination bit will be rather limited.

—Patricia Pitcher, *The Drama of Leadership*

Hired any crazies lately? Why not? Steve Jobs probably has. Steve experienced both great success and spectacular failure in the marketplace, but always brought creativity, intelligence, and imagination to the table. Legend has it that Steve often hires artists, poets and historians for his product development teams. Is he completely nuts or is he onto to something important? The magic according to Steve is that:

They have exposed themselves to the best things that humans have done and then brought those things into their projects .

Do a good job at creating the right culture and team environment and, at best, all the competition can do is steal some of your talent, they can't steal the engine that knows how to create and nurture it. HR must become the engine that creates and nurtures a highly creative, hyper-productive, workplace environment.



Knowledge Management
and Infotainment

Chapter 8

Knowledge Management & Infotainment

An invasion of armies can be resisted, but not the idea whose time has come.

—Victor Hugo

Forget big bang, top-down, big brother approaches to knowledge management (KM). The people peddling this stuff are selling pure snake oil, at very steep prices. It is the height of intellectual arrogance for anyone to pretend to have answers and deep insights in this emerging discipline. We are all babes in the woods, barely scratching the surface, and not very successfully at that. So don't expect to find answers in this chapter! The most I can hope to accomplish is to share a few insights and to comment on research that appears quite interesting.

I can't tell you what KM is, but I can tell you, with some degree of certainty, what it is not. KM is not discussion databases, document management systems, data warehousing, data mining, email on steroids or any other existing software category that, at one time or another, has been proposed as part of a KM solution. These technologies have their place, and when used appropriately can produce significant returns on investment; however, they should not be confused with KM.

What is infotainment? The obvious answer is the combination of information and entertainment. It is one of those words that the digerati invented to mean whatever they (we) wanted it to. I will take artistic license (again) and provide yet another usage for this word. Infotainment describes the way marketing delivers stories in a world of rich electronic content. Although the Internet is full of experiments, businesses, on the whole, have been reluctant to adopt rich multi-media in order to turbo charge their online marketing efforts. Despite this, infotainment will become the knowledge conduit of choice between buyers and sellers. The more interactive it is, the more effective it will be.

One of the principal arguments for this reluctance has been the issue of insufficient bandwidth. Rich content is bandwidth intensive, and the only thing that consumers' hate more than a poorly designed web site, is a slow one. Bandwidth remains a real issue. Despite the fact that DSL and Cable Modems have improved the state of the art, we still have some distance to go before fast universal access is available to the masses. Even access from within many corporate firewalls remains pathetically slow. Many employees have faster access at home.

Knowledge workers require fast access to the largest information repository known to man. The Internet, as a legitimate work-centric resource, is so prevalent and so important, that it perplexes the shit out of me why corporations do not make high speed access for their employees a number one priority. These same organizations preach the virtues of automation while at the same time slowing down everyone's productivity.

So fucking what if employees use the Internet to check their stocks and keep up with the news of the world? Misuse is an altogether different problem, not unlike spending hours on the phone with friends and family. Perhaps you should install the cheapest phone system you can buy, one

that guarantees static, as a way to cut down on personal phone use.

Another, more insidious reason, why corporations have been slow to adopt rich content as a marketing tool, is our puritanical social history. Rich content feels too much like Hollywood, an idea that we are all attracted to but often for all the wrong (read sinful) reasons. Be careful though, cling to staid corporate images at your own risk. You will be victimized by nostalgia for a business world that was never real to begin with. Even the most conservative of corporations (e.g. IBM) are now going out of their way to appear hip.

KM, Infotainment and eLearning are all converging into alternative story-telling mechanisms. These ideas are indelibly linked in my mind since there is so much conceptual overlap that blurs any clear lines of demarcation. In each case the power resides in the story, although multi-media technologies certainly enhance it.

Chapter 8: Knowledge Management & Infotainment

Telling Stories

Why are we obsessed with automating that which is impossible to automate? Better to focus on the creation of knowledge and its transfer, then to control it. For thousands of years man has been transferring knowledge, from generation to generation, via storytelling, without any need for computers. To the degree that software can enhance our story telling capabilities, it has the potential of improving, instead of impeding, our ability to transfer and retain knowledge.

Learning requires interactivity, a living and breathing relationship between teacher and student, master and apprentice, producer and consumer. What's more, these roles are transient. Today's master on a given topic is tomorrow's apprentice on another. Consumers no longer accept information that is shoved down their throats. They want and need the ability to talk back. It is the height of ignorance to attempt to automate human interaction out of the equation. We have become so enamored of our tools that they often blind us and rob us of our common sense. All learning occurs in a social context through conversation:

We view conversation as essential. We use it as a medium for decision-making. It is through conversation that we create, develop, validate, and share knowledge...Conversation is more than simply an intellectual endeavor: it is fundamentally a social process. Conversation is social in two ways. First, people speak to an audience. Speakers notice how their audience is reacting and steer their remarks appropriately: nods and eye contact convey one message; questions and furrowed brows another; yawns and fidgeting still another. Second conversation is social in that people portray themselves through conversation. They advance their personal agendas, project their personal style, take credit, share blame, and accomplish other social ends through their talk, often with a great deal of subtlety. The social nature of talk is not an undesirable side effect, but rather the heart of it.

Knowledge transfer requires intimate personal relationships and master teachers comfortable with their craft. How is it that we can fill volumes with KM related content and provide scant mention of teaching and education?

I think we risk becoming the best informed society that has ever died of ignorance.

—Reuben Blades

Chapter 8: Knowledge Management & Infotainment

Reward the Teachers

We all remember our favorite teachers, or at least we should. We owe them a debt of gratitude that can only be repaid, I think, by participating in their profession in small ways on a daily basis. There is no shortage of opportunities to practice this vocation: home, church, work, etc. If the captains of industry want to accelerate the pace of knowledge transfer within their organizations, and between their organization and external stakeholders (e.g. customers, suppliers, partners, and the public at large) then they should begin to reward the natural teachers that live within their walls.

How should you go about doing this? Pay the usual suspects a few million dollars to institute a formal reward program? I don't think so! I have argued that projects are the economic unit of value creation; they are also the mechanism by which rewards should be distributed. Are there not great teachers on the team that created your latest million-dollar revenue machine? Pay them for the value they brought to the table and you will ensure that they will continue to practice their craft at your For-Profit University.

Creators of knowledge should be rewarded in a similar fashion, but teachers more so. Why is that? Teachers, in whatever discipline they happen to find themselves (e.g. marketing, engineering, software development, etc.), are often both creators and propagators. Creation and propagation, in the business world, are synergistic and self-reinforcing. Rarely, if ever, are they mutually exclusive. Developers are not capable of teaching if they are not respected creators. Without the ability to create they lack the necessary credentials to teach.

Doers are rare. Thinkers are rarer. Thinker-doers are rarest.

—Unknown

Natural teachers fall into the third category and are rarer still. They are masters of collaboration and the true leaders of the new economy. They represent the neuron cells of any organization's knowledge transfer engine because they create, propagate, and teach.

Chapter 8: Knowledge Management & Infotainment

Small is Beautiful

Most KM practitioners insist on dealing with the organization in its entirety as the key abstraction for exploring and measuring KM initiatives. Why this obsession with bigness? It is either a fallacy of the highest order, or a convenient way of generating more revenue per consulting engagement. If we want to understand something as complex as KM then we need to focus on a more granular abstraction (i.e. the project team). Physicists seeking a better understanding of the universe decided to explore the smallest unit of matter they could find and were amazed by what they discovered.

Knowledge transfer always seems to happen most effectively within small communities sharing a common social context, whether among members of a high performance team inhabiting the same physical work environment, or a closely-knit virtual community working on similar problems. Social interaction, constant experimentation and feedback power the learning process.

If you want a return on your KM investment then you should focus on empowering these communities with the tools and resources that they deem to be KM enablers. Imposing a one size fits all solution is a complete waste of time and money. Only the insiders know what will work within their learning ecosystems. That said, KM czars certainly have an important consultative role to play. They can make suggestions related to emerging technologies and / or anecdotal evidence of what has worked elsewhere.

Once a community decides on an approach, financial resources and warm bodies are critical to the prototyping and implementation effort. It is OK to think global as long as customized solutions are driven locally, and by individuals that are the most likely to be impacted (i.e. teachers and students). Mission critical projects rarely get this kind of support. World-class knowledge workers always search for tools and techniques that will improve both personal and group productivity. They never lack ideas that would enhance both, if only they had the luxury of pursuing them.

Do not be conned into believing that you can engineer your way to KM nirvana. That is utter nonsense. First of all, most corporate cultures have a complete disregard for the humanities in the workplace and tend to treat all forms of labor as replaceable parts of the machinery. As if the automatons that you created are going to magically turn into renaissance men and women. There are many things that need to change before corporations begin to resemble institutions for higher learning.

Chapter 8: Knowledge Management & Infotainment

Information and Entertainment

Most people enjoy being entertained, even beyond what is reasonable for the health of our psyches. Witness how billions around the globe have been seduced by television. If Karl Marx were living today he would surely insist that television is the opiate of the masses. Despite the Internet's rising popularity, television remains king of the hill.

The convergence of text, graphics, full-motion video, and sound into powerful web-based story-telling technologies will transform the business world long before it changes the viewing habits of television's captive audience. In the B2B universe, where sales cycles are longer, and revenue per transaction greater, their impact is likely to be the most dramatic.

Marketing has always been about telling stories. And now technology has taken story telling to an entirely new level. Despite the dot-com implosion, where the marketing machines lost all sense of reality, multi-media significantly enhances the spinning of a good (credible) yarn, and technology will continue to lead the way. However, it is the clever use of these technologies by old economy corporations that will signal the dawn of a new era.

The Changing Nature of Brands

When I search the web looking for products I am not interested first and foremost in brand. I am interested in acquiring the knowledge necessary to make an informed decision. I read product data sheets, review white papers and case studies, and expect to be sold, and yes, entertained. How so? If your content is boring, if it is comprised of pages and pages of pure text, you are going to lose my interest real fucking quick. I will be at one of your competitor's sites faster than you can blink.

We know a lot about how brands have worked in the past. But now more than ever, there is evidence that our knowledge of brands isn't enough for these times.

—Young & Rubicam Executive, Advertising Age International, September 19, 1994.

Regis McKenna, Silicon Valley marketing guru, makes the following observation about the changing nature of brands:

Brands had the effect of “reducing or eliminating the need to find out about a product before buying it,” as one marketing primer put it. But all that is now history. Branding of a new and entirely different kind is being born—brand as an encapsulation of actual, experienced value. The nature of that experience is increasingly determined through customer preferences expressed in dialog with producers or service providers—an exchange made possible by technology, and one in which the consumer has the upper hand.

Chapter 8: Knowledge Management & Infotainment

Screwing the Pooch Revisited

Industrial Age marketing organizations have been slow to rid themselves of their mass marketing addiction, even though they have readily embraced enterprise CRM (Customer Relationship Management) “solutions”. I am no CRM expert, but I am always suspicious of product categories that include the words enterprise and management in their sales pitch. In the past, they always had a big-bang stench surrounding them, proving to be anything but the promised panacea; often creating more problems than they solve. Big Business is enamored with Big Solutions, even though they are often anathema to the mission and completely miss their mark.

Perhaps by now the reader has the impression that I have an axe to grind against enterprise implementations of any packaged software (I can’t imagine why), and by implication the usual suspects that rake in millions from these consulting engagements. I have first hand experience with these death-march projects and they tend to create long lasting negative work related memories. In a recent report by the Meta Group they had the following to say about the relative success of ERP implementations:

The average ERP implementation takes 23 months, has a total cost of ownership of \$15 million and rewards (so to speak) the business with an average negative net present value of \$1.5 million. And the news gets worse.

The anecdotal evidence about CRM implementations that I read about in the trade press indicates that they are suffering the same fate. What is the dynamic at work here? Rarely is it that software does not work as advertised. Almost always the failures are the result of people and process problems. But how can this be? The usual suspects are hired because of their experience and the long list of stellar reference accounts that backs them up.

Clearly there are systemic root causes for these problems since the same patterns emerge over and over. It is not surprising that the majority of implementations grossly exceed the allocated budget. There are well-understood hidden costs that neither the executives who are often trying to camouflage a serious organizational issue, nor the usual suspects that are trying to win a deal, are interested in discussing. Since the magnitude of the hidden costs tend to be quite large, the ROI calculations of the project’s value are exaggerated, to put it mildly.

Getting with the Program

News organizations adopted enabling web technologies faster than any industry I have observed, including technology. For example: all the leading news sites are heavy users of video as a means of enhancing their product offerings. Why don’t we see the same level of multi-media use with respect to marketing of old economy products and services? Labeling marketing departments as luddites or (when kind) laggards, and ascribing to them sinister motives related to defending their turf is too simplistic to be useful.

The best and brightest in these organizations are aware of the power and potential of new media, but have been deprived of much needed organizational support. What sort of support might that be? Philosophic and strategic endorsement of new media by the corporate power structure as well as technology support and training required to drive down the costs of production. These setbacks are temporary. Soon we will begin to see online marketing efforts take on a completely different look within successful leading edge companies.

The aforementioned companies are likely to be the ones that discover that better results are achieved through a cultural transformation based on radically reoriented thinking than from the implementation of a canned solution. Skunk works projects, and lots of 'em, will get you where you need to be faster than any big-bang project designed primarily to treat symptoms of the disease, while further destroying the patient's health in the process.

Chapter 8: Knowledge Management & Infotainment

Interactivity

Knowledge is like power. It is rarely transferred. It is almost always taken. The acquisition of knowledge requires intense interaction with ideas. Only by interacting with ideas do students internalize “know how”. It is widely known in the technology industry that most formal training is a waste of time and financial resources when it is not applied immediately after the course work is complete. Training companies are not in the knowledge transfer business; they are in the information presentation business.

Real training occurs when students immediately apply information and relate it to things they already know. It is the application of information that leads to internalization and the creation of this thing we call knowledge. The best developers rarely require formal training. They know they are in the knowledge business and are extremely adept at seeking out the information required as a starting point for their learning processes. Once they have information they begin to experiment and interact with it. They research and experiment and then research some more. They realize intuitively that learning is an iterative process, one that they are instinctively familiar with and enjoy.

Constructivist theories and active learning theories have helped educators understand the way learners actively create meaning by exploring, experimenting, testing, and applying knowledge in self-directed and collaborative fashions (rather than in a predetermined course of study).

—Warren Longmire, *A Primer on Learning Objects*

Communities of practice (e.g. high performance development teams) are important because they provide additional opportunities for quality interaction and reflection. Increased interaction and reflection leads to more experimentation in a virtuous feedback loop. A healthy social environment within communities of practice is conducive to enhanced learning experiences since peers are usually quite attuned to their comrade’s abilities and points of reference. Through story telling and examples they are often able to answer the un-asked question.

The goal for web-based interactivity should be to create micro-worlds that an individual (e.g. customer, student, etc.) actively participates in. Active participation implies that the individual implicitly or explicitly assumes a role in this virtual universe. The micro-world represents a new reality that provides the context for effective transfer of information. The degree to which this experience allows for two-way dialog increases the probability that the individual will move from information, to conversation, to experimentation, to the acquisition of new knowledge and ultimately to quality decisions. The challenge for marketers and virtual instructors alike is the crafting of micro-worlds that embody information in a succinct, entertaining, conversational and interactive manner.

Chapter 8: Knowledge Management & Infotainment

Experiencing the Brand

Circa 1985 there was a nascent software marketing strategy called shareware. It was a “try before you buy” concept and there was serious debate over how much nagging of the user should occur once the evaluation period had expired to encourage the purchase and registration of the product. In 1987 the Association of Shareware Professionals was formed with the following mission in mind:

to strengthen the future of shareware (user-supported software) as an alternative to commercial software. Its members, all of whom subscribe to a code of ethics, are committed to the concept of shareware as a method of marketing.

—The Association of Shareware Professionals Website

The association is alive and well, you can visit them at www.asp-shareware.org.

I imagine that many software providers still make a good buck using shareware as their principal marketing strategy, however, you don’t hear the term much anymore since nowadays almost all software distributed via the Net has evaluation (eval) copies available for download. Software vendors that do not provide eval copies are usually out of business in short order. There is also a fair amount of entry-level free software available whose goal is to allow the user to own a fully functional, though limited, product (e.g. standard editions versus enterprise editions). The hope is that the user will find a need to migrate to a paid edition.

Eval copies are a way to experience both the product and the brand. Only if you’re impressed by what you see are you likely to take the necessary steps that will lead to a purchase. Many vendors force you to fill out information forms before allowing you to download, and some will even follow-up with an email or a phone call, imagine that. We take all this for granted now, but back in the 80’s this was brand new territory.

As necessary as this practice is in the software space, most software marketing is still woefully behind what is currently possible. Product data sheets, white papers, presentations and the like are clearly more than what old economy companies tend to provide, but since almost everyone in software provides these extras, there is very little differentiation. Multi-media has a role to play here, but it has been lagging for all of the reasons previously discussed.

The try before you buy concept is applicable to all soft goods that can be distributed electronically via the Net. I rarely buy a book online without reading material from the excerpt and checking out the reviews. This is all expected and appropriate behavior for soft goods but what about products made of atoms? Good question. The answer is that these products are in dire need of effective multi-media micro-worlds if the provider wants to stand out from the crowd (name one that doesn’t). If it is impossible for the consumer to physically experience the product you have to do the next best thing. Create a simulated world where they can experience both the product and

the brand, virtually.

Chapter 8: Knowledge Management & Infotainment

Context-Sensitive Rich Content

It is widely known that searching, with respect to a buying decision, is a transaction cost that must be considered as an important part of any purchase process. This is true whether we are in the market for a pair of shoes, an automobile, or a million dollar piece of machinery. The more expensive the potential purchase the more willing (and likely) we are to incur additional search based costs.

Why do we search? Obviously, we search in order to gather the information necessary to make a prudent and knowledgeable decision. At some point we stop searching and rationalize the information that we have gathered. The stopping point occurs when we have found all the pertinent information (rarely), when the search-based transaction costs begin to exceed the relative value of the purchase (sometimes), or when we get tired and bored and want to get on with life (almost always).

The same principle applies when we search for information related to decisions we need to make in order to complete our work assignments. What would happen if we could, within certain decision domains, drive the search-based transaction costs close to zero? Obviously, we would make better decisions faster, at the same time dramatically improving productivity. This is essentially the Holy Grail of KM: get the right information, to the right people, at the right time.

Our crude attempts at solving this problem to date has centered on building bigger and better repositories (e.g. specialized databases) and improving full text-based search techniques. There is widespread agreement that these improvements have been beneficial. In the aggregate, two technologies, search engines and email, have dramatically accelerated the velocity of information worldwide and have been the prime movers in the creation of the largest library that the world has ever known.

As important as web-based technologies are and will continue to be, they do not constitute a breakthrough in KM. Searching for information, irrespective of how fast we can make it, will always take longer than we would like it to. Context sensitive content provides a step in the right direction. The software development process has been improved significantly by the inclusion of rich context sensitive help systems within state of the art Integrated Development Environments (IDE's). While the availability of specific information related to our current context does not constitute knowledge transfer, it does serve as an effective starting point.

If you accept the premise that knowledge transfer requires human dialog then by definition improved search technologies will never quite get us where we need to go. I present some plausible scenarios, implementable with existing technologies, which represent the KM future according to Leyva. The scenarios include context sensitive rich content as the starting point that leads to on-demand real-time conversations when required.

Chapter 8: Knowledge Management & Infotainment

Big Automation Company

After the dot.com implosion I have spent a significant amount of my time working with Big Automation Company on their next generation simulation product. They have a 15-year track record of providing high fidelity training simulators to some of the largest oil refineries on the planet. The Distributed Control Systems (DCS's) that run these refineries are 24/7 real-time systems with NASA like high availability requirements (i.e. they almost never go down). Anytime these systems are down for even short periods of time (i.e. minutes) the refineries lose millions.

The fact that these systems run 24/7 is great for refinery productivity but makes the training of refinery operators a difficult task. To address this problem the refineries have been using high fidelity simulation based training software since the advent of the DCS's. The simulation software contains graphical representations (i.e. models) of sections of the refinery that instructors use to develop training scenarios. These scenarios include: plant startup and shutdown, equipment malfunctions, DCS alarms, etc. Since the simulation models contain accurate virtual representations of plant processes and equipment they provide an excellent platform for the development of context sensitive rich content.

To my knowledge none of the simulator-based training systems, including those from Big Automation Company, have the functionality I describe in the following text, but they could. The technologies required to implement these ideas are all commercially available today. As mentioned, the plant models used in simulator training systems for refineries contain graphical representations of real world plant objects such as pumps, valves, distillation towers, etc. In addition, the models contain a representation of the plant topology. That is, how all the pieces of equipment are “laid out” and connected to each other.

Imagine clicking on a pump icon and selecting a menu option that describes the maintenance history of that specific pump. In addition, you can select another menu option that plays an audio/video clip describing the procedure required to maintain this pump to the specifications proposed by the manufacturer. The man in the video is one of the senior operators from this section of the plant.

Need additional information regarding this pump? Select another menu option that takes you to the manufacturer's web site where you can talk via real time chat with a manufacturer's representative about recent problems that you experienced with this piece of equipment. Sounds far fetched? Perhaps it is, but only because we are unaccustomed to thinking of computing systems along these lines. This idea is much too obvious not to find its way into leading edge products within the next few years. It is a killer app in the making!

I can hear the geek wheels turning. Leyva, you dumb shit, who could build such a system? The automation vendors that supply the simulation software would need to collaborate with perhaps hundreds, if not thousands of manufactures in order to pull this off. And what will it cost to

produce maintenance videos for every piece of plant equipment? Well I never said that it would be trivial or inexpensive (big assed grin)!

Bare with me a moment and lets explore the possibilities. Web services are, as we speak, standardizing app-to-app communication over the Net; so communicating with a large number of manufacturers will soon be plausible (notice I didn't say easy). Producing audio and video clips for every piece of maintenance equipment would be time consuming, but the technology to produce it and persist it is available today, and getting cheaper all the time. Besides, how expensive will it be when blue-collar boomers retire and take all their tacit knowledge out of the plant with them?

Lets assume, for the sake of argument, that all this is not only doable, but cost effective as well. Why would it be important vis-à-vis KM? Because if context sensitive rich content can lead to real time conversations then one of the missing pieces has just been added to the KM puzzle, people. Human beings conversing with other human beings lead to relationships and the building of trust. Once trust is established human beings begin to engage in one of the things they do best, telling stories. Stories lead to the effective transfer of information, which leads to experimentation, which leads (possibly) to the accumulation of new knowledge.

Chapter 8: Knowledge Management & Infotainment

War Games

The U.S. military has a long investment history in simulation-based training and all signs indicate that this investment is on the rise. Consider the monumental training needs facing all branches of the Armed Forces: a never-ending stream of new recruits, skills maintenance for enlisted men and women in a constantly evolving techno-driven military universe, and training the officer corps in a broad range of strategic enabling technologies. Despite the economic horror stories of the Pentagon paying \$1,500.00 USD for a hammer, when it comes to purchasing state of the art video game based training simulation software, the U.S. Armed Forces are at the top of the class, their knowledge base far exceeding that of even the most advanced private sector corporations.

Business people are slowly “getting it”. Schools get it here and there. But the U.S. military gets it big time. The military has embraced digital game-based learning with all the fervor of true believers. Why? Because it works for them. And trust me, the guys in charge of training at the Pentagon are a very sharp group. They have seen and evaluated everything.

Of course the military is capable of creating micro-worlds that are unheard of (yet) within the private sector, including the use of sights and smells:

To enhance the sense of reality, smells including burned charcoal can be pumped into the room. Participants can gesture and touch objects and elicit responses in the simulator. The machine also uses voice-recognition technology and different languages to allow participants to converse with the characters they encounter.

The intent of simulation-based training is not to replace field training but to effectively complement it. In other situations, such as large-scale nuclear and chemical accidents, simulation-based training is the only viable alternative. Given our recent successes in a number of regional conflicts around the world, one would have to conclude that the boys in the Pentagon are doing something right.

Chapter 8: Knowledge Management & Infotainment

Smart Appliances

Smart household appliances such as coffee makers, electric blankets, smoke detectors, mixers, bathroom scales, and an assortment of health monitors are poised to make the Jetson home of the future a reality today. These products utilize protocols that leverage existing electrical wiring and phone lines to talk to each other, all controlled by Mama Jetson from a central console. The implication is that all of these contrivances will make life easier, but that remains to be seen. However, one thing is certain, billions of dollars will be invested by an odd combination of bedfellows in order to make the Jetson household a reality.

Some unusual partnerships are forming in the business world today, alliances between leaders of high technology and—of all things—companies that manufacture ordinary household appliances. Microsoft has teamed up with Maytag, Sun Microsystems with Whirlpool, and Sunbeam, working with Zilog and emWare, has formed a new spin-off called Thalia. The purpose behind all this collaboration sounds like the stuff of fiction, since these companies plan to provide us with smart appliances and intelligent homes, all based on embedded chips and linked to the Internet for sending and receiving information.

I have serious doubts concerning how quickly this market will develop, let alone mature. If the present is any barometer of the future, then aspiring average Jetsons will just have to wait! Currently the Leyva household has four remotes for controlling our stereo and television. It pisses me off to no end when I can't remember that one arcane option that I need, when I desperately need it. What pisses me off even more is that I have to ask my wife how the fucking gizmo works again (for the hundredth time)!

Despite making a living in high-tech, I am not a gadget lover. Why? I spend my working life climbing learning curves and reading manuals. The last thing I want to do when I come home is to read yet another fucking manual to figure out how to program my digital thermostats. Chinese water torture would be preferable. In fact, I had some digital thermostats and I wound up ripping the fuckers out after losing the manual. I couldn't figure how to re-program the blasted things!

Only a high degree of usability and content sensitive rich content (as previously described) will work for me. I don't want smart appliances that know how to talk to each other. I want smart appliances that allow me to talk to human beings when I need them! The inmates have been running the technology asylum for a long time now and I will be damned if I let them loose in my house anytime soon.

Chapter 8: Knowledge Management & Infotainment

eLearning

A lifetime of learning is the idea that forms the foundation of the eLearning movement. This is no passing fad, although specific delivery technologies might be quite transient, the eLearning phenomenon itself is quickly becoming one of the cornerstones of our social order. There is widespread agreement within and across various communities (e.g. business, government, education, etc.) that continuous learning initiatives are central to maintaining our competitiveness and standard of living. The goal is clear, even though there may be ongoing and widespread disagreements concerning where and how the appropriate investments should be made. That said; let us not delude ourselves into believing that because of our technical prowess we will naturally or easily dominate the global knowledge based economy.

The European Community (EC) is on a Mission

At the European Council in Lisbon in March 2000, Heads of State set an ambitious target for Europe to become within ten years “the most competitive and dynamic knowledge-based economy in the world, capable of sustainable economic growth with more and better jobs and greater social cohesion”. The EC appears to be making significant progress:

These efforts are producing encouraging results. The eEurope benchmarking for the Report to Barcelona Summit shows that the initial eEurope target of connecting all schools to the Internet is all but achieved, and that attention must therefore shift to better connections and wider educational use.

All signs indicate that the EC has indeed moved on from basic concerns of connectivity and infrastructure:

At school we are seeing greater emphasis being placed on the quality of e-learning products and services, and in the pedagogical context for their use. We are moving beyond questions of connectivity and infrastructure, to ones associated with content, teacher training and organizational implications, including new social interactions inside and beyond schools. In many cases ICT fosters the opening of the school to other sources of learning, such as multimedia libraries, museums, local community resources, research centers, and transnational co-operation. It may also foster new relationships and new roles for pupils acting as researchers, creators, designers, etc.

I am in no position to comment regarding whether or not the EC has attained a position of leadership vis-à-vis the U.S., but they certainly appear to be doing interesting work and achieving measurable results. They are focused and committed and apparently have a strong desire to control their own destiny.

This is an arena where the U.S. government should take a lead role. The building of the necessary educational infrastructure and connectivity should become a national priority similar to the

federally funded construction of the interstate highway system. The amount of tax dollars spent by the U.S. military in simulation and eLearning R&D should find its way into our universities and into the private sector. If we get caught napping we are certain to get our collective asses kicked!

Chapter 8: Knowledge Management & Infotainment

Individual Responsibility

Even though we can and should invest enormously in the social capital required to ensure that our citizenry has access to world-class educational opportunities, we will never be in a position to guarantee individual results. It is the responsibility of the individual to avail themselves of the best that the society has to offer. Self-reliance, stamina, dogged persistence, and desire are the educational pillars that enduring careers will be built of.

The ultimate goal of the educational system is to shift to the individual the burden of pursuing his own education. This will not be a widely shared pursuit until we get over our odd conviction that education is what goes on in school buildings and nowhere else. Not only does education continue when schooling ends, but it is not confined to what may be studied in adult education courses. The world is an incomparable classroom, and life is a memorable teacher for those who aren't afraid of her.

Jethro Bodine (of Beverly Hillbillies fame) and I, for many years, had one thing in common (in addition to our good looks), our eighth grade education. Leyva you are full shit, didn't you make it a point to mention in Chapter 1 that you graduated from the University of New Orleans with honors? Well yeah, but only after dropping out of high school in the tenth grade and subsequently getting my GED at the ripe old age of twenty. Well, your math is as good as Jethro's since you apparently had him beat by one plus years. Not really.

In the summer, after graduating the eighth grade from Central Junior High, in Atlantic City, New Jersey, we moved to New Orleans. I made all A's in ninth grade without cracking a book or learning a damn thing. I love New Orleans dearly but the LA public school system (at the time) left a lot to be desired. Despite this, it was not the New Orleans school system that caused me to drop out of school in the tenth grade but a true desire to get on with my education (of sorts).

I read Shakespeare and the Bible, and I can shoot dice. That's what I call a liberal education.

—Reuben Blades

Many of my high-tech friends scoff at my street urchin past but its true (mostly) and I have the scars and the stories to prove it, but I digress. I was happy that at the age of twenty, after having worked more odd jobs that I care to remember (including summers at a Green Giant factory in southern Minnesota), I was not locked out of the educational opportunities that I then desperately desired. The reason that I was able to take advantage of the opportunity was the wonderful K-8 education I received in Atlantic City's public school system.

I am a big believer in public education. It has been the advancement vehicle of choice for lower and middle class Americans for the last hundred years, and will remain so going forward. I owe a special debt of gratitude to one Mrs. Theresa Marsden who was my homeroom and English teacher in junior high. Our class was made up of some of the smartest students in the city: Blacks,

Hispanics, Whites, Jews, Catholics and we all learned more from her then we could ever repay. She grilled us on the fine points of English grammar and the importance of language in our lives. Her influence has carried me all these years. I believe that I only took one additional English course in college.

The society gave me an opportunity and I worked my ass off in order to take advantage of it. I could not have asked for more.

Chapter 8: Knowledge Management & Infotainment

Emerging Technologies and Research

I have argued consistently, if perhaps not convincingly, in this chapter and throughout this text, that human communications is critical to making progress within the software development process and across a broad array of emerging technologies, including KM. The following text explores examples of technologies that have the lofty goal of incorporating the human factor as a principle component of the solution. These technologies are in early stages of development and are not, to my knowledge, available commercially at this time. That said they do provide us a glimpse of the possibilities.

Both examples happen to come from research being conducted at IBM, although research along similar lines is being conducted at other major U.S. corporations (e.g. Microsoft).

The Tower of “Babble”

Babble is an online conversation environment that provides a forum for capturing and using knowledge in a manner that makes visible the actors involved in the process. The goal is to capture knowledge in a discourse base, as opposed to a database, while at the same time capturing the social context that guides, informs, and underpins the knowledge store.

Imagine a knowledge management system that was designed from a social perspective, a system predicated on the assumption that knowledge is rooted in a social context. Such a system would assume that knowledge is produced within, and dispersed among, a network of people; that only a small proportion of knowledge is captured in concrete form; that knowledge sharing involves social factors like relationships, trust, obligation, and reputation.

Babble captures both synchronous (i.e. real time chat) and asynchronous conversations. In addition “social proxies” capture the manner in which individuals interact within a specific conversational topic. The system has been deployed by a number of groups within IBM and the subsequent analysis of group behavior has led to some interesting findings including improved trust and better relationships (work and otherwise) among group members.

Through our work on Babble we have begun to create an infrastructure that can support rich forms of social interaction. We have found that social proxies are a promising development, and we continue to be impressed with the power of plain text as a means of supporting interactions that are both complex and subtle.

Babble is an example of social computing that contains many seeds that are likely to yield significant fruit as we progress toward a more human centric computing universe. I have done the authors a gross injustice with such an abbreviated description of their work. I would encourage the interested reader to review the IBM Systems Journal, as documented in the End Notes, for a more complete treatment of the subject.

Chapter 8: Knowledge Management & Infotainment

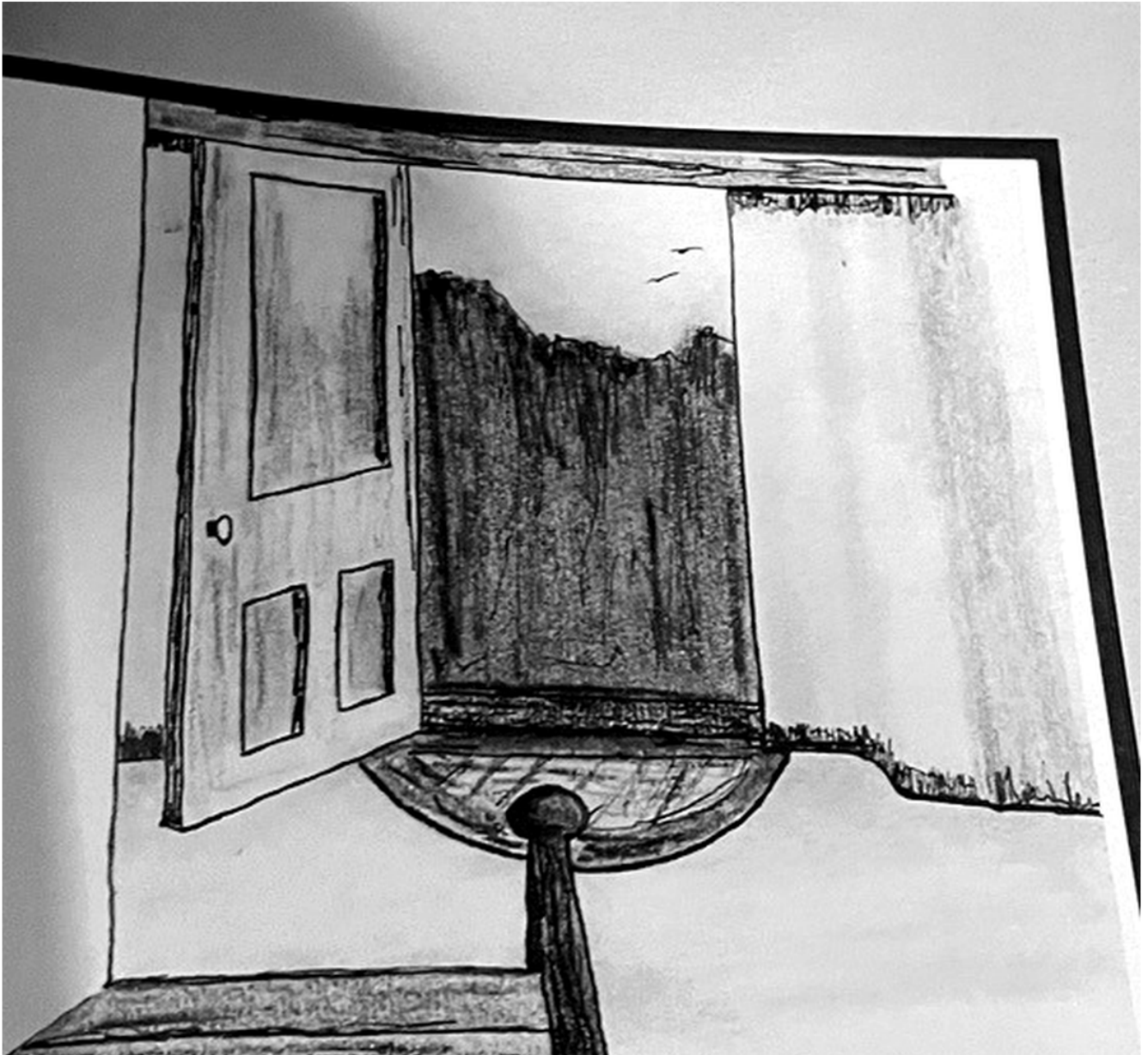
Telling Stories Revisited

Researchers at IBM are in the process of developing a markup language for capturing the essence of stories. The name of the markup language is StoryML. They propose the following conceptual framework for describing a story:

According to this research stories are important because they are useful in creating, capturing, disseminating, and internalizing knowledge and that it accomplishes all of these simultaneously and not sequentially. This research supports the anecdotal evidence that I have presented in this text regarding the importance of human dialog within KM. Stories are extremely useful and powerful communications constructs within and across disciplines.

As instantiations of a type of knowledge that can be used in so many business processes, stories also have the advantage that they can help knowledge flow through the organization. Not only are stories capable of being used by many different business functions (marketing, design, management), they are also capable of being understood by various professions. Thus stories can serve not only to support communities of practice with a common vocabulary; they can also serve an important coordinating role within a team whose members come from different communities of practice.

In short, story telling is the communication construct of choice for collaborators working within and across disciplines on the increasingly complex task of bringing products and services to market. Not only does story telling contribute significantly to the creation of these offerings, but likewise in the articulation of their benefits to the marketplace.



What Comes After
What Comes Next?

Chapter 8: Knowledge Management & Infotainment

Epilogue

It is no surprise that industrial age organizations are dysfunctional with respect to the processes and skills required for effective story telling. Within these organizations story telling was (is) the purview of the few and the privileged, thus they lack both a cultural and intellectual foundation upon which to build. Rectifying this dilemma is one of the many challenges in attempting to jump the curve from an industrial age mentality to an age where information and knowledge rule.

KM, infotainment, and eLearning are all different manifestations of the same problem space. We are on a mission to discover effective approaches for telling stories digitally. Context sensitive rich content, powered by compelling and feature rich multi-media technologies is an important enabler, but not the principle driver. The convergence of story telling and technology is the killer app of the next millennium. Central to this process is the elevation of the human voice (and spirit) to the fountainhead from which all value springs.

Chapter 9: What Comes After What Come Next?

The Soft Stuff is the Hard Stuff

However, if there were an overarching theme within this text, a take away of sorts, it would be this: human communications is key to advances that we seek within science and technology. This meme is often summarized in the business press with the following phrase: the soft stuff is the hard stuff. The soft stuff is the antithesis of what drove the industrial revolution, and therefore, it should come as no surprise that our cherished institutions are having a difficult time coping. Evidence abounds supporting the fact that the soft stuff (interpreted in whatever fashion you like) produces hard profits, so struggle on.

I have made a crude attempt of describing how the soft stuff relates to the software development process. I have also alluded to the fact that in many ways, little understood at present, the business of business will be dominated by software for the foreseeable future. Hence, in my mind at least, this text is more of a business book than a technology tome. Now I am about to jump into the deep end and discuss things, which I am not qualified (or eminently qualified?), to speak of. You are welcome to jump into these perilous waters with me, but be forewarned, there are no lifeguards on duty. I'm gone.

Two tears in a bucket, motherfuckit...

Chapter 9: What Comes After What Come Next?

Prisons or Schools

There is not another investment in social capital that I can think of (granted I am no expert on social policy but I am willing to listen) that will level the playing field of economic opportunity for all our citizens, and ensure the future of the nation, than an investment in public education at all levels, but especially K-12. In order to guarantee the possibility of a chicken in every pot, let us find a way to provide a laptop in every student's backpack.

People will harbor a hopeful view of what they can achieve only if their societies do, in fact, offer some scope for individual accomplishment. If their societies provide them with the opportunity to grow as individuals and to have an impact on their environment, their attitudes will reflect those realities.

I am sick of the demagogues of both the right and the left that cannot find a way to improve the current state of affairs with respect to public education (funded by whatever creative means our best minds can devise). Perhaps public education should become the poster child of The Radical Center so that, at least on this one issue, we can move beyond political labels and the politics of personal destruction, however entertaining and amusing we might, from time to time, find these devices. It is not at all amusing that we jeopardize our children's (and through them the nation's) future because of a fucked up, broken and outmoded way of thinking.

Let us within the technology industry assume our share of civic responsibility and take a lead role in the public education space. I challenge the dominant corporations to collaborate in creating the necessary processes, and seed the program with the necessary resources, in order to provide a framework for individuals to leverage when donating their time and energies. Let us take our experience, humble as it may be at the moment, related to KM, infotainment and eLearning and apply it where it is likely to do significant good. Consider it an investment in the great developers and visionary leaders of the future, precisely the ones that will lead our corporations to even greater triumphs tomorrow.

Make your educational laws strict and your criminal ones can be gentle; but if you leave youth its liberty you will have to dig dungeons for ages.

—Michel de Montaigne

He who opens a school door, closes a prison.

—Victor Hugo

Chapter 9: What Comes After What Come Next?

The Good Old Days

I love America, warts and all, but I cringe whenever I hear politicians whining about the good old days. America has never been better, however her finest hours lie ahead. Do you want to return to the good old days of the 1950's? Black Americans without basic civil rights in the South, Mexican Americans treated as indentured servants, women as second-class citizens? Fuck that! I too hear America singing, singing her endless songs of rebirth. I am an American and a patriot. My ancestors have roamed these continents for thousands of years. I am a son of immigrants and a singer of the immigrant's song, prepared to defend America to the death, if need be.

Johnny get your gun, and kiss your momma bye

The time has finally come, are you prepared to die?

For that which you believe in, for you family and your friends

That sweet old cause is calling us, and upon us it depends

To make a contribution, to take a clear cut stand

The music is playing all around us Johnny

Do you wanna join the band?

—War, The Aztec Kings

Yeah who wants to join the band? America, despite her faults, is the greatest nation on Earth. Those who would belittle her would do better to clean up their own houses before throwing stones. Clean 'em up or we may have to clean up the fuckers for you! The American renaissance will not be led by the weak kneed. Challenge us if you like, but be prepared to get your asses kicked. If you fear our Armed Forces, just wait until our technology and culture are hitting on all cylinders.

Leyva, you are a complete lunatic! Why all this talk of a renaissance, of war, and of patriotism? Because, my dear friend, the answers lie in our people, in art, faith, and hope—not in technology.

About the Author

Through reckless, risk, taking, and dumb luck, I have managed to ride wave after wave of this roller coaster called the technology industry. I have had the good fortune to wear many techno-hats: from developer to Chief Architect, from Professional Services Manager to Director of Applications Development. Mostly I have earned a living as a top-gun consultant, running just as fast as I can to keep up with the young-turks on the latest technologies and the business gurus on emerging macro trends.

Throughout my technology career, I have been an avid (my wife would say obsessed) student business and the business use of emerging technologies. I consider myself a bona fide geek, even though I rarely geeked just for the hell of it. I was always more interested in how technology could solve real world problems and in connecting the dots between business, technology, economics and pop culture.

A few years ago, Debbie and I started the B2Bdepot, Inc, with the vision of creating a boutique internet consultancy that focused primarily, but not exclusively, on architecture. Why architecture? Because architecture appeared to encompass so many of our professional interests and passions: branding, software development and all the eBilities (scaleability, availability, reusability, etc.) Despite the bottom falling out on the market (ouch), we have had a blast and manage to make a few bucks at the same time.

I am looking forward with anticipation to catching the next wave. It is coming just as surely as night follows day. See ya on the beach!

About the Illustrator

Deborah L. Leyva

The technology business has been very good to me for many years now. Mine is one of those Cinderella stories that happens over and over in this wonderful and crazy industry of ours from: Secretary, to Engineering Analyst to Developer to Director of applications in the IT organization of a major financial services firm, where I lead a department of fifty. I left that job to start the theB2Bdepot, Inc.

I was never quite the geek that Carlos (hubby) is; however, that did not prevent him from dragging me (sometimes kicking and screaming) along his many techno-journeys: from C to C++, from PowerBuilder the Delphi, from Unix the Client Server, and now onto internet technologies. It has been an amazing and at times, scary ride.

I have recently discovered that over half of my brain (can't remember if it is the right or left) that has led me to painting and other artistic pursuits. I am a Web Diva turned Artist. I had a wonderful time doing the artwork and illustrations for Silicon Stories even though Carlos can be overly demanding and an all around pain in the ass to work sometimes (sorry honey.)

You can view my artistic *stuff* on our website at www.theb2bdepot.com. Better yet, if you want to buy one. Send me an email.

Carlos might be looking forward to catching the next technology wave, but frankly I would be happy just to spend more time contemplating the universe with a couple of margaritas in the sun and hanging out with Dallas (our two year old Sheltie.)