FabTime Book Reviews

Author's Note: These book reviews were written by Frank Chance and Jennifer Robinson for the FabTime website, mostly during the 2000s. They refer to then-current editions of the books and may not always reflect current editions. We feel that there is still value in making these books available, as they were instrumental for us in creating and growing our software business, FabTime (now owned by INFICON Inc.) They cover business and management in general and cycle time management in particular.

Focus by Al Ries

This book without question changed the way we run our business. Although many of the ideas in the book run counter to common intuition, nearly everything we read resonated with us. Ries' essential message is that the marketplace is very crowded. Therefore, the best way for a company to succeed in the long term is to focus on one area and become the leader in that area.

According to Ries, two trends suggest the benefits of focus. One is that the bigger the market is, the more specialization is required for a company or product to stand out. For example, in a small town, a single general store might be sufficient. A large city, however, can support a variety of different kinds of stores. This suggests that as a company gets bigger, and goes after more global markets, they should offer fewer products, not more products. The second trend is that categories tend to sub-divide as products mature (e.g. there used to just be cars, now there are 12 categories of cars). The leader of the main category is rarely the leader of the later categories (especially not all of them). People tend to be more comfortable buying from the market leader in a particular area and will attribute certain benefits to that leader. However, if the leader blurs the issue, by also claiming to be the leader in some other area, people will often switch to using a different product. People trust specialists.

Focus! boils down to common sense. If a company is very good at one thing, and concentrates its efforts around that one thing, the company will be more successful than if it tried to do many different things. Ries fills the book with real-world examples that prove this point again and again. So convincing is this book that within a few months of reading ***Focus!*** we changed our company name and mission to reflect a singular focus on wafer fab cycle time reduction. We recommend that you read it too. It could change the way you look at the world.

The Goal by Eli Goldratt and Jeff Cox

This book has been widely read by semiconductor manufacturing personnel since it was first published in 1984. It accurately describes the behavior of manufacturing facilities, including such fundamental concepts as bottlenecks, constraints, and the impact of variability. One reason why it has been so broadly read is that it frames these concepts in the guise of a novel. This makes the ideas easy to read and digest.

The premise is that Alex, a factory manager, is given an ultimatum -- dramatically improve the performance of his factory in three months, or the facility will be shut down. Believing that traditional improvement strategies will never make enough difference in such a short time, Alex must resort to more desperate measures. He tracks down an old professor, now working as a consultant, and begs for advice. The advice of this consultant, Jonah, sets Alex and his team on a journey. Instead of just giving them the answers, Jonah asks them questions, and refuses to give

more help until each question has been answered. As Alex learns through this process, so does the reader.

Some of the lessons of the book include the following.

- When you are productive you are accomplishing something in terms of your goals. Every action that brings a company closer to its goal is productive. The goal of a manufacturing organization is to make money.
- Because of variability, a factory cannot be run at 100% of capacity. Or, as Jonah says, “the closer you come to a balanced plant, the closer you come to bankruptcy.”
- One of the biggest problems in improving your factory is collecting the right data. Alex eventually concludes that “we're going to have to accept the fact that we're not going to have perfect data to work with.”
- “An hour lost at the bottleneck is an hour lost for the entire system … The actual cost of a bottleneck is the total expense of the system, divided by the number of hours the bottleneck produces.” This suggests managing bottlenecks very closely. This idea has spawned numerous consulting and software firms since the book was published.
- Non-bottlenecks do not need to be regulated so closely and should not be operated to maximize utilization. Jonah says that “activating a non-bottleneck to its maximum is an act of maximum stupidity.”

We think that everyone who works in a manufacturing facility should read this book at least once. We re-read it at regular intervals, and always find it insightful

Factory Physics by Wallace Hopp and Mark Spearman

Factory Physics is the kind of textbook that people refer to while doing jobs out in the real world. It places a mathematical framework around the behavior of factories, describing the underlying relationships with clear, easy-to-read examples and summing them up into straightforward Factory Physics Laws.

The book is targeted toward factory operations. The first section is a review of the traditional production management curriculum, including inventory models, MRP and JIT. The second section contains what we find the most useful material in the book, with chapters on basic factory dynamics, variability, and pull production systems. The third section puts the principles from the second section to use, and includes chapters on total quality management, production planning, shop floor control, scheduling, capacity planning and inventory management.

This book is filled with simple numeric examples and charts but does not shrink from including queueing formulas where they are necessary. The Factory Physics Laws are easy to remember, but also important. For example, Law 11 states: “Pay me now or pay me later: If you cannot pay for variability reduction, you will pay in one or more of the following ways:

1. Long cycle times and high WIP levels.
2. Wasted capacity (low utilization of resources)
3. Lost throughput

We recommend this book for those who have some engineering background but want to better understand factory operations and planning systems. It's the text that we would use if we were teaching a course on the subject.

Crossing the Chasm by Geoffrey Moore

Moore's thesis is that when a new type of high-tech product is introduced, it follows a predictable path of marketplace adoption. First, a small group of visionaries buys the product. The visionaries look for products that have high strategic value. They are willing to live with software that is not quite polished. They like new things and are looking for breakthroughs. They are soon followed by early adopters, who are willing to take on high risk products in return for expected high rewards. Visionaries and early adopters, however, make up only a small portion of the market. The bulk of the market consists of the early majority and the late majority (also called pragmatists). A product cannot be successful in the long term without being adopted by this mainstream market.

The problem is that pragmatists are only comfortable buying products that require significant change if they know that other pragmatists are already using them. Pragmatists are more cautious, and only want to use products that are well-established and well-supported. They tend to distrust the opinions of visionaries and want to wait until their pragmatist counterparts begin using a product. This creates a chasm in the marketplace. Sales drop after the early market is saturated, while the pragmatists all wait for someone else to make a move.

Fortunately, Moore offers several concrete suggestions for crossing the chasm. Chief among these is the notion of targeting a niche market in which people can communicate more easily. Also important is the idea that to reach the mainstream market, vendors must provide the "whole product," including supporting services. For more details, we suggest that you read the book.

Microchip Fabrication by Peter Van Zant (Fourth Edition)

The subtitle of this book is "A Practical Guide to Semiconductor Processing," and we found this to be accurate. Though not exactly a book that will keep you up at night wondering what happens next, it is an excellent reference for anyone who wishes to know more about the technical side of semiconductor manufacturing. It is well-indexed, and the chapters are self-contained enough to make it useful for catching up on one topic at a time. We found the introductory material on the background of the semiconductor industry to be particularly interesting, as was a later chapter on the business side of the industry.

Most of the book, however, is concerned with the process side of chip manufacturing. Van Zant begins with the atom and explains in an easy-to-read introduction just why it is that conductors work at all, and why silicon is used for most wafers. He also discusses other substrates, and how the wafers themselves are manufactured. He then moves on to an overview of the main processes in wafer fabrication, before moving in ever-increasing levels to detail to very specific wafer fabrication processes. He concludes with chapters on the various types of devices and integrated circuits in use today, and methods for packaging them. For anyone working in the semiconductor industry, and wanting to better understand its techniques, vocabulary, and history, this book has much to offer.

Our one complaint about the book is that in the version we read, the publisher apparently rushed to make the publication date, and as a result, the book is filled with small editing errors (incorrect punctuation, sentence fragments, inaccurate acronyms, and outright garbled sentences in some

cases). This makes the book harder to read. However, due to the book's wealth of technical content, we think you will still find it worthwhile.

The Effective Executive by Peter F. Drucker

To be effective, Drucker writes, is to do the right thing. This is in contrast to mere busyness—doing many unimportant things right. Effectiveness is a set of habits that can be practiced, and thus learned. Drucker breaks these habits into five categories:

1. Know Thy Time
2. Focus on Contribution
3. Make Strengths Productive
4. Do First Things First
5. Make Effective Decisions

Drucker begins with time, the fundamental constraint of the executive. He recommends keeping a detailed log of activities, to find out where time is being spent. Once this record has been established, it is possible to manage one's time and to cut back on unproductive usage. The discretionary time that results can then be consolidated into the largest possible blocks. Drucker argues that only with these large blocks of time can the executive tackle the most important tasks and hope to achieve results:

All one can think and do in a short time is to think what one already knows and to do as one has always done.

Within each category, Drucker proposes concrete practices that advance executive effectiveness. He illustrates these practices with examples from his substantial consulting experience. Some of these examples seem dated in this frenzied dot.com era. However, the dot.com executives may be providing us with an updated study on the difference between busyness and effectiveness. As Drucker points out:

> "The people who get nothing done often work a great deal harder. In the first place, they underestimate the time for any one task. They always expect that everything will go right. Yet, as every executive knows, nothing ever goes right. The unexpected always happens—the unexpected is indeed the only thing one can confidently expect. And almost never is it a pleasant surprise."

This book is a concise source of concrete ideas that can be immediately put into practice. For those joining management from the technical world, this book is especially helpful—while you were writing your thesis on superconducting RF cavities, your peers were studying Drucker in business school. If you don't have a copy, pick one up at the airport before your next flight. Odds are you will dog-ear at least one page from each chapter before you land.

World Class Manufacturing Casebook: Implementing JIT and TQC by Richard Schonberger

Schonberger is often credited as originator of the phrase "World Class Manufacturing". Since the early 1980's he has been publicly beating the Just-In-Time drum. Although the popularity of JIT in

general and Kanban systems in particular has waxed and waned over the years (remember Quality Circles?), the basic concepts are here to stay:

- Inventory hides problems and increases costs.
- Cutting inventory forces you to solve these problems.

A ruthless depletion of inventory logically leads to a manufacturing line with closely coupled stations, where each station begins work only when the downstream station requires it. Thus, inventory is pulled through the line rather than pushed through it.

In this casebook, Schonberger presents descriptions of 26 JIT implementations. Most of the sites are in the U.S., with the remainder in Asia. None of the cases are wafer fabs, although several assembly/test facilities are included. Each case includes discussion questions (for Schonberger's thoughts on the questions, see "Instructor's Manual to Accompany World Class Manufacturing Casebook: Implementing JIT and TQC"). The casebook is designed as a companion to Schonberger's "World Class Manufacturing: The Lessons of Simplicity Applied."

Kanban implementations have been attempted in a number of wafer fabs, but the results have been decidedly mixed. Considering this, it seems worthwhile to periodically review the original JIT literature, both for useful principles and for explanations as to the difficulty of wafer fab implementation.

In terms of useful principles, Schonberger has a section on "JIT ratio analysis" that is worth reviewing. In it, he discusses three ratios:

- Lead-time to work content: This measure is widely used in wafer fabs, and is commonly referred to as the "cycle time multiplier" or "X-factor" (the ratio of actual cycle time to raw process time).
- Process speed to sales rate: This measure is not often seen in wafer fabs (probably because it is most applicable to assembly line layouts). It refers to the ratio of production rate to consumption rate. For example, if an upstream tool is capable of 60 wafers per hour while the downstream tool is only capable of 10 wafers per hour, the process speed to sales rate ratio would be 6. The JIT goal is to drive this ratio to 1, e.g. to achieve a perfectly balanced line. Theory-of-Constraints advocates will necessarily disagree with this Goal.
- Number of pieces to number of workstations: This measure is not often seen in wafer fabs, but it does offer useful insight. It is like the "lead-time to work content" ratio, but viewed through a different window, namely counting inventory (in wafers) and dividing by the number of tools. A high ratio indicates excess inventory or numerous large batch tools (in either case, bad for cycle time).

In terms of JIT implementation difficulties in wafer fabs, a quick scan of the 26 cases reveals the following:

- Most of the factories surveyed were able to move equipment with relative ease. In wafer fabs, moving a single tool (let alone many tools) is often difficult, with problems ranging from utility support lines to regulatory permits.
- Most of the factories surveyed had relatively short manufacturing process flows, ranging in length from 5 steps up to 25 steps. Most wafer fab process flows involve hundreds of steps.

- Although these factors do not mean that implementing JIT/Kanbans in a wafer fab is impossible, they do indicate there will be hurdles unique to wafer fabs. And in any event, we may still profit from a thoughtful application of JIT principles and concepts.

If you are interested in the history of JIT's introduction to the US, this book is for you. If you are interested in the details of JIT implementation, and you don't mind the dated nature of the examples, this book is a useful reference. If you are interested in applying JIT to wafer fabs, you'd best keep looking. Personally, we would like to see a follow-up to these cases, a JIT reunion of sorts, to learn the fates of these 26 factories.

The Theory of Constraints and Its Implications for Management Accounting by Eric Noreen, Debra Smith, and James MacKey

The Goal (1984, Goldratt and Cox) paints a dim view of cost accounting. The disagreement is fundamentally about performance measures. Under full-absorption product-costing, there is a tremendous incentive to build inventory and keep machines busy. Under the throughput-accounting methodology advocated by the theory of constraints, the emphasis is on producing only those products that can be sold. Switching a factory to operate by the theory of constraints often causes full-absorption measures to move sharply in the wrong direction, even as the factory improves bottom-line measures such as revenue and profit.

This criticism of full-absorption costing is not new, however. Opponents of absorption costing have long argued that it leads to excess inventory. What Goldratt has done is to dramatically display the debilitating effects of excess inventory, and to promote a methodology (the theory of constraints) that leads to a sharply different set of measurements:

1. Throughput - the rate at which the factory generates money through actual customer sales;
2. Inventory - money the factory has invested in things which it intends to sell; and
3. Operating Expenses - money the factory spends to turn Inventory into Throughput

These performance measures are not entirely revolutionary, however. Variable costing is a widely taught alternative to full-absorption costing; throughput accounting is to a great degree an extreme version of variable costing. Contribution margin analysis is also taught in management accounting classes and embodies many of the ideas of throughput-accounting.

Several questions remain, however:

1. How exactly does throughput-accounting differ from tools of management accounting such as variable costing and contribution margin analysis?
2. How are companies using throughput-accounting in practice? Have they thrown out their existing full-absorption reports and replaced them with throughput-accounting reports? Or are they using a mixture?

In this book, Noreen et al. present the results of their study on these questions. They begin with a presentation of the theory of constraints, and a comparison of throughput-accounting with the newer management accounting methods. This section of the book is brief but thorough and does a good job comparing and contrasting the methodologies.

Next, the authors present case study summaries of factories (no wafer fabs, unfortunately) that have implemented the theory of constraints. Although not a statistically random sample — the authors solicited volunteers at conferences sponsored by the Avraham Goldratt institute -- the case studies still provide useful insight into how a company can use the theory of constraints in practice. As an added benefit, the authors report on the extent to which the more abstract thinking processes (a generalization of the theory of constraints presented in later Goldratt books) are used at the study sites.

The book concludes with the authors' observations and speculations regarding the theory of constraints and its implications for factories. Although the authors seem a little too ready to stand in awe of the thinking processes, they do a good job of outlining when and how these techniques could be used in practice.

It's certainly not a page-turner — how many accounting books are? -- but this book achieves its stated purpose and makes a handy bookshelf companion to ***The Goal*** and ***It's Not Luck*** (Goldratt 1994). If you are interested in the comparison of throughput accounting and variable costing / contribution margin analysis, it is especially valuable.

The Tipping Point by Malcolm Gladwell

The Tipping Point is the name given by epidemiologists for the dramatic moment in an epidemic when everything can change all at once. The flu, for example, can be held in check for a long time without being an epidemic. But suddenly, once some threshold is crossed in terms of number of people infected, things get much worse very quickly. Gladwell's premise is that in addition to applying to viruses, this type of pattern is observed in many other situations. The book is filled with far-reaching examples, from the resurgence of Hush Puppies shoes to the popularity of Sesame Street to an epidemic of teen suicides in Micronesia.

Perhaps the most well-known example described is the rapid fall in crime levels in New York City in the mid-1990s. Murder rates fell by 64.3% in a five-year period, with other types of violent crimes dropping by 50%. This happened after years of steady increase. Gladwell argues that the factors conventionally cited as causing the improvement (improved policing, declining crack use, and aging of the population) are not sufficient to explain the suddenness of the change. All three factors included gradual shifts in behavior, and yet the drop in crime occurred very rapidly. Gladwell makes a convincing argument that the police in New York put into place certain conditions that suddenly "tipped" the crime epidemic, sending crime rates into a decline.

Gladwell believes that by understanding how such tipping points are reached, we can deliberately use them to market products, or push for social changes, or just understand ourselves better. He describes three types of people who disproportionately affect social and behavioral epidemics and explains how their behavior starts the ball rolling. For example, Connectors are people who know a lot of other people and can spread an idea through multiple communities. Their presence makes an idea contagious. Next Gladwell outlines the concept of "stickiness" that makes people who hear about a new idea remember it, and in some way do something about it. He suggests that little changes in the presentation of ideas, designed to make the ideas stickier, can have big effects. Finally, he discusses the necessity of context for a real tipping point to be reached.

Many of the discussions illustrating contagiousness, stickiness, and context go off onto tangents regarding human behavior, nature vs. nurture, and various sociology experiments. This, perhaps, stems from the author's background as a writer of science-based magazine articles. He is unable to

resist a good story, even if it doesn't directly relate to his main premise. However, most of these side excursions are interesting, and all are well-researched and documented.

One section of the book that we found especially interesting was a chapter in which the author draws a parallel between the tipping point and the chasm described in Geoffrey Moore's ***Crossing the Chasm*** (see our review above). The premise of the chasm is that during the introduction of discontinuous technological innovations, a chasm exists because most users are unwilling to accept the recommendation of very early users (innovators). Gladwell proposes that the chasm can be crossed by the same mechanics followed in reaching the tipping point. The tipping people that he has already identified move things across the chasm by translating them from what innovators want to what mainstream users want. He gives an example in which a marketing company deliberately courts these tipping people to move a sneaker from highly adventurous skateboarders to the mainstream market.

We also found it interesting that a tipping point-like behavior was observed in Jennifer Robinson's Ph.D. dissertation on time-constrained processing for wafer fabs. Time constrained processing is when some process step must be completed within a particular time window of a previous step. If the time window elapses before the wafer is processed at the later step, it must return to the earlier step for reprocessing. In simulations, Jennifer found that for many systems, some small amount of reprocessing could be handled without any difficulty. However, once the level of reprocessing exceeded some threshold (different for each system), behavior would rapidly degrade into instability. This threshold is, apparently, a tipping point for time constrained systems.

Connections to our other areas of research aside, we recommend this book because it is a quick, interesting read that might give you a new perspective on human behavior.

The Non-Designer's Design Book by Robin Williams

As small business owners, we spend a lot of time preparing materials for presentation. Marketing materials, web pages, business cards, letterhead, and of course our software itself. Even if you don't operate a business yourself, you probably present written information to other people, in the form of letters or reports or charts, on a very regular basis. This month's featured book is one that can help you to present this information better. ***The Non-Designer's Design Book*** outlines a set of principles for design and typography that we try to follow every day. The four basic principles are Contrast, Repetition, Alignment, and Proximity.

The idea behind Contrast is to avoid using elements that are like one another. Things should be either the same, or very different, to provide contrast. For example, you wouldn't want to have your title be just slightly larger than your regular text, in a very slightly different font. This just makes the reader see that something is off, without knowing what it is, and makes your document look unprofessional. Instead, you should make the title much bigger, or in a very different, bolder, font. The book describes in detail how to choose fonts and styles for contrast. If you start applying this principle, you will see positive effects in your work immediately.

The premise behind Repetition is that you should repeat certain visual elements throughout for your work, such as colors and shapes and line thicknesses. This makes your work look more unified and gives the reader a sense of familiarity. It can also make the page look more interesting, so that people will be more likely to read it. Williams recommends that you think of repetition as being consistent, and then push that idea further to find places to deliberately add repetition. For example, you might use small squares as bullets, and use the same bullets to mark the end of text in your headers and footers.

The third principle, Alignment, is straightforward. The idea is that everything in your document should be in some way aligned with other things on the page, either vertically or horizontally. Nothing should be arbitrary. This gives the reader clear lines to follow, and makes your work look more sophisticated. Under the principle of alignment, Williams cautions strongly against centering things, because she argues that left or right justified alignments form a much stronger line for the reader's eyes to follow (with centered text you have two ragged lines instead). We found this surprising at first, because like most people we were taught in school to center headings. However, after reading Williams's justification, and looking at her examples, we have become believers.

The final principle is Proximity. Proximity states that elements that are logically related to one another should be placed near one another on the page. Similarly, adding white space between elements that are not logically related helps the reader to visually distinguish between them. The most obvious application of this principle lies in the design of business cards, which are easiest to use when related information is clustered together.

The book is filled with examples to drive home these four principles and help you to understand why Williams thinks that they make sense. The book also includes puzzles and exercises to reinforce these points, all provided in a fun, easy-to-read style. The book is also a helpful reference for understanding certain typography elements like types of fonts (serif or sans-serif, thick/thin transitions, etc.). All in all, if you want to make your reports and presentations and charts look better, or learn a little bit about typography, this is the book that you should buy.

Further reading:

The Non-Designer's Web Book by Robin Williams and John Tollett. This sequel to ***The Non-Designer's Design Book*** includes a review of the principles established in the first book, applied in detail to web page design. There are also some helpful tips related to meta-tags and other administrative features of web design. We refer to it often.

The PC Is Not A Typewriter by Robin Williams. This little book outlines various typing conventions that started in the era of proportional-font typewriters, and now make your work look amateurish (e.g. typing two spaces after the end of each sentence).

Root Cause Analysis by Robert Latino and Kenneth Latino

To appreciate this book, it is best to understand that the authors are in sales-mode for most of the text. In the first two-thirds, they are selling the concept of root cause analysis, and more particularly their methodology for doing so. In the latter one-third, they are selling their company's software (PROACT). This software automates the process of performing root cause analysis. Both authors work for the Reliability Center, Inc., a for-profit consulting firm that specializes in industrial reliability consulting. The Reliability Center was founded in 1985, when the authors' father retired from Allied Chemical and went into the consulting business (he had established the reliability engineering department at Allied in 1972). While this book is obviously not a textbook on root-cause analysis, you can still glean useful insights from it.

Root cause analysis is the orderly process of searching backwards from an undesirable event or outcome to its causes. Here the authors make a useful distinction between types of causes:

- Physical root causes — these are tangible causes likely to be found first in the search. For example, if the outcome is "tool down", the physical root cause could be "electricity

failure". The authors point out that you should not stop working at this point, because the most interesting and useful root causes lie much deeper.

- Human root causes — these are decision errors that usually result in physical root causes. In our example, the human root cause could be "maintenance flipped a breaker off to work on a different tool, not knowing that it would affect our tool". When you find human root causes, the authors say, you should starting asking the question "why?"
- Latent root causes — these are policies, procedures, and practices that people use to make decisions. Continuing our example, the latent root cause could be "breaker-box diagrams are not updated on a regular schedule". The authors argue that latent root causes are the true culprits.

The authors also make several interesting points when discussing the focus of root cause analysis projects. They argue that many root cause analysis projects fail due to lack of focus, e.g. a propensity to work on every undesirable outcome, rather than the few most important events. They draw a distinction between sporadic events (dramatic impact, but limited frequency) and chronic events (not so dramatic, but happen all the time). The authors believe that chronic events, while not costly on an individual basis, are costly on a cumulative basis, and are the best focus for root cause analysis.

The book includes individual chapters on the steps of root cause analysis (putting together the team, performing the analysis, communicating results, tracking implementation results). Again, you sometimes have the feeling that you are reading a consulting project proposal, e.g. "here is what we would do if you hired us to come and perform a root cause analysis at your facility". But if you were to perform a project like this on your own, the authors' big-picture blueprint would be quite helpful.

In summary, the sales nature of the book is off-putting at times. However, the authors have been in the reliability consulting business for many years, and you have to respect their experience. If you are planning a structured attempt at root cause analysis in your facility, you will pick up enough tips from this book to easily justify its cost.

Faster - The Acceleration of Just About Everything by James Gleick

January 1st, 2001 seems to be an appropriate date to review James Gleick's book ***Faster***, since the book is about the continual (and especially the recent) acceleration of many aspects of our lives. ***Faster*** also seemed appropriate for a FabTime book review, because it is by and large about time. ***Faster*** received numerous positive book reviews from newspapers and magazines when it was first released. The book is well-written, and well-researched, with a slew of references and anecdotes. It is written in a detached, but frequently amusing style, and covers a wide range of sub-topics concerning time.

The main idea behind the book is that as our technologies speed up, so do other aspects of our lives, such as our expectations for entertainment, and doing multiple things at the same time. Gleick feels that "if we don't understand time, we become its victims" and so the book analyzes our relationship with time. Particularly interesting tidbits include the fact that the door close button on most elevators is disabled, but remains, in part, to give people a sense of having more control over the process. Also, the fact that television and radio interviews are routinely

compressed slightly, to remove the annoying pauses and delays of ordinary speech. Rush Limbaugh was apparently quite surprised when he found that his producers were able to save six minutes out of each hour-long program this way (and fill them with advertising, of course). We also found interesting Gleick's examples of how large corporations save their own time at the expense of their customers' time, by using things like complex automated voicemail systems.

We thought that the chapter most relevant to semiconductor manufacturing was one entitled "The Paradox of Efficiency". Here Gleick explains how increasingly efficient systems are more negatively affected by unexpected events than systems that include more slack. The example used in the book is commercial flight scheduling. The airlines have become so good at scheduling flights to minimize idle airplanes that, when there is a problem with a plane, there are no extra planes around to take over. Similarly, with tightly scheduled hubs, weather delays cause cascading series of problems. Gleick refers to systems without much spare capacity as "tightly coupled", where perturbations are felt throughout the system. This is analogous to a wafer fab in which all tool groups are highly utilized. A problem at one tool can lead to WIP bubbles throughout the fab, because the tools have insufficient slack to absorb the extra variability.

Gleick seems to have a slightly negative attitude towards this acceleration of our culture, and the things this acceleration is taking away from us. However, it is difficult to say for certain, because he refrains from drawing many conclusions. We thought that the book would have benefited from some firmer conclusions, some further discussion of the implications of the acceleration on our future. However, Gleick writes more as a scientist, dissecting a phenomenon in which he has interest. And, of course, nothing he writes is going to slow down society's "overdrive", so perhaps there is no point in his making empty recommendations. Overall, if you have taken note of the ever-decreasing amounts of quiet time, the chronic increase in multi-tasking, or just the general way in which time seems to be going by faster and faster, you will likely find Faster an engaging read.

Necessary But Not Sufficient by Eli Goldratt, Eli Schragenheim, and Carol Ptak

Necessary But Not Sufficient is the fourth in Goldratt's series of "Theory of Constraints Business Novels". As in the other books, Goldratt makes his points through the format of a novel - the reader learns what Goldratt is trying to teach through the discussions and experiences of the characters. As in all of these books, neither the characters nor the plot are well-developed enough for the book to stand on its own as a novel. However, the technique works, because the novel format is both more entertaining and more accessible than a standard business book. At its best, this format encourages the reader to put down the book for a minute and try to think through to the logical conclusion of whatever problem the main character is currently facing.

Necessary But Not Sufficient focuses on the fortunes of a large ERP (Enterprise Resource Planning) software vendor. The company is doing well in its industry, growing 40% each year, but faces a series of difficulties. The company president and sales director are confronted with the approaching market saturation in their target industry (very large corporations), a saturation which will in turn prohibit future 40%/year growth, and eventually kill the company's stock prices. The company's technical guru and the head of their implementation arm are struggling with the ever-increasing complexity of the software, which is dramatically slowing the response to customer-reported bugs, and customer-requested enhancements. Even as they are dealing with these issues,

the management team is confronted by the needs of a loyal customer who has been asked to demonstrate the bottom-line value of the ERP software by his board of directors.

They spend quite a bit of time on this last problem, and conclude that to realize bottom-line value, the technology in the software is necessary, but not sufficient. To provide everything that is needed to extract the full value from the system, the company must do things that software companies do not ordinarily do. In particular, they must identify the hidden assumptions in how people were doing their jobs prior to implementing the software and find out which of those assumptions no longer holds true with the software in place. Re-configuring such rules is necessary to extract value from the new system. For example, if your ERP system enables you to run financial reports once a day, instead of once a month, but you still only run them once a month, because that's the process that's in place, you won't achieve the benefit that you could be getting.

As they work with this client, the management team comes to realize that the bottom-line value question can help solve their other problems, too. At this point, they discover the Theory of Constraints methodology and begin expanding their ERP system to use TOC production, project management, distribution, and engineering modules. This evolves into a bit more of a sales pitch for Theory of Constraints and Drum-Buffer-Rope than we would have liked. Basically, everyone in the book who is introduced to TOC finds it "refreshing" or "perfect" or something to be "crazy" about, and it leads to tremendous improvements. While we have nothing against TOC as a philosophy - there are many good things about it - it made the book seem too much like a vehicle for selling TOC consulting. Despite this problem, the book is an engaging read and offers considerable food for thought in regard to improving both software sales and manufacturing performance.

In summary, this book proposes that when you are trying to justify a software purchase, or justify a new feature in a software program, you should focus on the bottom-line value of the system. Traditional software benefits like "better visibility into operations" are not useful, unless they can be translated into dollars. When implementing a large-scale software system, it is necessary to modify operating rules to fully leverage the potential of the software. According to Goldratt, software vendors and the companies that implement their products will be successful if they follow this prescription.

Why Systems Fail: And How to Make Sure Yours Doesn't by David Turbide

This is a book about MRP II installations. More generally, it is a book about complex system implementations. Many of these implementations fail. Many more fail to achieve the lofty goals originally promised. Turbide sets out to explain why so many fail, and how to maximize the likelihood that yours won't.

MRP II (Manufacturing Resource Planning) includes the original MRP (Material Requirements Planning) plus a host of linked modules (production planning, customer service, and financials). The term MRP has fallen out of vogue, replaced by ERP (Enterprise Resource Planning), but the core functionality remains the same, as have the implementation challenges:

- Extended implementation schedules (1-2 years). The long schedule makes it difficult to keep up the enthusiasm required of everyone in the company to perform not only their regular job, but also the extra work required to bring the new system online. And, it means

that whoever you assign to lead the project full-time will need a new job upon completion of the project, since their regular assignment could not go unfilled for such a long time.

- Extensive business process changes. These large systems touch every aspect of a business, and consequently require widespread retraining. Many times, employees have to adapt to fit the system, rather than the system adapting to fit the employees. These changes are stressful and can lead to resentment and disuse of the new system.

For those not interested in the details of MRP II systems, Turbide does a good job of distilling his experience into a few concrete recommendations, and then backing these recommendations with examples from his consulting practice. Some of these recommendations will be very intuitive to you:

- Successful implementations have strong and visible executive support from start to finish.
- Successful implementations avoid mission creep and schedule slippage.

Other recommendations may not be so intuitive:

- Successful implementations have a team leader that is from a user department (e.g. production, customer service, or accounting), not from MIS. Turbide specifically recommends that MIS be involved in the system selection process, but only serve in an advisory role during the implementation. And he recommends that the team leader always be a company employee rather than a consultant.
- Successful implementations focus a disproportionate amount of time and energy on education and training. Turbide recommends that you make a reasonable budget for education and training, then double or triple it. With sufficient training, he argues that any shortcomings in your software (and there will be areas where the package does not fit your needs) can be overcome.
- Successful implementations use packaged software with minimal aftermarket customization. Turbide believes that to do otherwise is to invite long-term system support headaches. His thesis is that several systems will fill 80% to 90% of your requirements, and the choice among these systems is not nearly so important as how you implement the chosen system. Modifying your system during the installation makes it much harder to upgrade, and harder for your vendor to support you.

For wafer fabs, the closest parallel to MRP/ERP is your choice of MES. If you are thinking of replacing your MES, you should read this book first. It could make a big impact on the way you view the implementation process and save you significant headaches in the process.

Leading with the Heart: Coach K's Successful Strategies for Basketball, Business, and Life by Mike Krzyzewski and Donald Phillips

This book review was written by Jennifer Robinson

Let me start out by disclosing that I received my undergraduate degree from Duke in 1989. This is essentially the same as saying that I'm a Duke basketball fan. You already know this if you know anyone at all who graduated from Duke during the Coach K years. So, when I tell you that I loved this book, you'll have to understand that I'm not objective. But I can safely say that if you're a Duke

basketball fan, and/or a Coach K fan, you'll enjoy this book, and you might find it inspirational. If you're not a Duke fan, well, you can't argue with success. Coach K just won his third NCAA Championship, only the fourth coach ever to win three titles. And he did this with a program that nearly always graduates all its players (did in fact graduate them all until very recently).

This book is about leadership. It traces Coach K's development as a leader, from the organizer of high school pick-up games, through his tenure at West Point, his assistant coaching days, and finally, his time at Duke. He draws lessons from his experiences, and from the important people in his life, and returns to the same lessons time and time again throughout the book. Discipline, persistence, trust, leadership, honesty, and of course, how to win.

The book uses a basketball season as a framework, and moves from pre-season (groundwork, organization, core values, leadership) to regular season (teamwork, training, rolling with the punches) to post-season (crisis management, focus, celebration), and finally to a few all-season priorities (friendship, character, basics). Many of the principles described can be applied to any type of team, not merely sports teams. Each chapter concludes with a one-page list of "Coach K's tips". If you don't have much time, you could just read through these tip pages and find them worth the price of the book. Most of the tips are common sense. For example:

- Goals should be realistic, attainable, and shared among all members of the team.
- Having fun helps reduce pressure.
- Cultivate relationships with people who support your organization.
- Business, like basketball, is a game of adjustments. So be ready to adjust.

Some of the tips sound a little hokey, like "The only way you can lose is if you don't try your best" and "Touch people's hearts with sincerity and excellence." And yet, if you read the whole book, the thing that comes through is that Coach K believes in everything he says. And his methods obviously work. This is what makes the book inspirational for me, the fact that he clearly practices what he preaches.

What makes the book fun to read are the examples, nearly all drawn from basketball games and teams over the past 15 years. For example, what did Coach K say to his team during the time out right before the fantastic overtime finish of the Duke-Kentucky game in 1992? Was it luck, as many people say? Or was it an example of leadership? How did Coach K react when Elton Brand, Corey Maggette, William Avery, and Chris Burgess all left early in 1999? What did he say to his team after the crushing loss by 30 points to UNLV in the 1990 Championship game? If you've ever been curious about any of these things, this is the book for you.

The book isn't all that well-written, in a classic sense. Some of the sentences run on or are terse. It reads, in fact, like the author is talking to you. Some people will find this annoying; others will find that it reinforces the message. The book is well-organized, and a quick, easy read. It may leave you with some ideas for team building, or with some additional motivation to be a good leader. Or, at the very least, it will leave you with some Duke basketball trivia to share with your friends.

How We Know What Isn't So: The Fallibility of Human Reason in Everyday Life by Thomas Gilovich

If you watch sports at any level, you're bound to hear variants of these phrases:

She's on a roll!"

“He’s on quite a run!”

“She’s unstoppable!”

“What an awful streak!”

“He’s in the zone!”

The implication being that streaks are a natural part of athletics. But what, exactly, constitutes a streak? We each have an intuitive feel for hot streaks — it’s that time when you’re hitting every pitch, timing your overhand smashes perfectly, and draining every putt. And similarly for cold streaks — the pitcher has your number, the net is a 6-foot chain-link fence, and you’re hitting every tee shot onto the fairway... for the next hole.

Are streaks real? They must be — we experience them all the time!

But let’s look a little deeper. Take basketball’s Kobe Bryant for instance. What does it mean for Kobe to experience streaks? If we look at his pattern of hits and misses, how can we quantify the presence or absence of streaks? One reasonable hypothesis is that once Kobe hits a couple shots in row, then he’s in the zone, thereby increasing his chance of hitting his next shot. Or if he misses a couple shots in a row, he loses confidence, thereby decreasing his chance of hitting his next shot. This hypothesis implies that his shots are not independent rolls of the dice, e.g. the outcome of one shot influences the outcome of other shots.

Having taken our share of stats classes, we can test this hypothesis. What we do, in fact, is to assume the opposite:

- Null hypothesis: Kobe’s shots are independent (no streaks).
- And then we see if we can reject this null hypothesis, based on the data.
- One way to test this hypothesis is to perform a runs test. A runs test makes no distributional assumptions about the underlying data -- it merely looks at the number of runs in the data and compares this to what would be expected when drawing truly independent random numbers.

In the spirit of scientific investigation, we recorded the outcome of each of Kobe’s field-goal shots for the 1st game of the western conference finals between Los Angeles and San Antonio (May 19, 2001). Here is the data (O Missed shot, X = Hit shot):

O O O X X X X X X O O X O O O (halftime) X X O X O O X X O X X O O O X O X O X X O

[Sports Illustrated lists Kobe as hitting 19/35 shots from the field for this game -- this could be due to a different treatment of shot attempts where a foul occurs. Our analysis is based on the sequence shown above.]

Looking at the first half, Kobe came out cold, then got red hot, then cooled off again before halftime. And in fact, here is a sampling of the announcer’s first-half comments:

“What an awful start” (after he missed three in a row)

“The basket is looking mighty big right now” (after he hit 5 in a row)

“This is an unstoppable run!” (after he hit the 7th in a row, and coincidentally right before the run did, in fact, stop).

Faced with this overwhelming evidence, we must conclude that streaks exist, right? Well... we’ll perform our runs test, just to confirm our intuition.

Using Minitab:

- Total Observations: 37
- Field-goal percentage: 51.35% (19 shots made out of 37 total)
- Observed number of runs: 19
- Expected number of runs (under null hypothesis of independence): 19.4865
- Expected number of runs (under null hypothesis of independence): 19.4865
- p-value: 0.8710 Cannot reject at alpha = 0.05!

Not only do we not reject the null hypothesis, but we're also not even close to rejecting it. The number of runs is frightfully close to what we would expect, assuming that Kobe's shots are completely independent. So, we are at a loss — our intuition told us that Kobe's game was filled with streaks, but the statistics said no: there were exactly as many streaks as one would expect from totally independent shots.

Now one example does not a life-changing experience make, but it's enough to make one stop and think. Perhaps if we look at other games for Kobe, we will get a different result. Or maybe our error was in the technical definition of a streak. And that's where we turn to Dr. Gilovich. Gilovich argues that no, if we look at more games, we'll spend a lot of time watching basketball (which might be enjoyable), but we won't prove the presence of streaks. And we can twist the definition of streak every which way, but we won't find a reasonable definition that results in statistical evidence of streaks.

If we admit the possibility that streaks don't exist, this raises a host of interesting questions:

1. Why do we believe in streaks in the first place?
2. Why do we believe so strongly in streaks?
3. Is our belief in streaks irrational?
4. If we are presented with data opposing the existence of streaks, will we change our minds?

Gilovich explores each of these questions in depth in this book. He covers sports, but also a host of other areas in everyday life where the things that we know turn out to not be so. Taken together, it's an eye-opening experience to see how our intuition can lead us astray, particularly in the presence of large quantities of data. As wafer fabs generate millions of statistical data points every day, our intuition has plenty of opportunities for mischief. In our effort to coax information from this vast sea of data, the distinction between belief (an untested hypothesis) and fact is an important one. Gilovich's book provides useful insight into how easily we are fooled. Pick up a copy — it's a worthwhile read.

Clockspeed: Winning Industry Control in the Age of Temporary Advantage by Charles H. Fine

Although it's primarily a book about strategic supply-chain design, Charles Fine's ***Clockspeed*** offers several useful insights for those in the wafer fab world. Fine begins the book with a discussion of what he calls "fruit-fly" or "fast-clockspeed" industries. These are industries where the natural lifecycle of a product is short, new product introductions come at a fast and furious pace, and

companies must constantly innovate or be pushed aside. For example, movie studios and retail electronics are fruit-fly industries. Automobiles and airplanes, by contrast, are slow-clockspeed industries. He argues that by studying the fruit flies, we can see evolutionary patterns that would take decades to appear in the slow-clockspeed industries -- much as scientists study multiple generations of fruit flies. And, if the clockspeed of all industries is naturally increasing (a reasonable hypothesis), then companies in slow-clockspeed industries can use the experiences of the fruit flies to guide them as the industry clockspeed rises.

From his study of fruit-fly industries, Fine argues that:

- The natural evolution for companies and products is to cycle between a vertical phase, where a few major players offer packages of highly integrated products, and a horizontal phase, where a host of niche players offer modular products. The faster the industry clockspeed, the faster the cycle between vertical and horizontal phases. It's important that you know where your industry is along this cycle and use this information to your advantage when putting together your supply chain.
- The faster the clockspeed of your industry, the more temporary your competitive advantage. The only sustainable competitive advantage is the ability to transition from one temporary advantage to the next.
- The closer you are to the final customer, the faster your clockspeed. For example, Dell works in a high-clockspeed industry, supplying computers directly to consumers and businesses (new product offerings every few months). Dell purchases microprocessors from Intel (new microprocessor generations every few years). Intel buys equipment from Applied Materials (new equipment platforms every three to six years). While the upstream companies (Applied) work in slower-clockspeed industries, giving them more time to respond to competitive pressures, they pay a price for this in volatility (see next point).
- The farther you are from the final customer, the more volatile your business cycle. Small ripples in demand at the consumer level are magnified many times over as you move upstream in the supply chain. At the time of this review, we're seeing graphic evidence of this in the bookings for equipment suppliers. When you are examining your supply chain, you need to consider the health of your upstream suppliers and whether they can withstand this bullwhip effect.

Applied to wafer fabs, one implication of *Clockspeed* is clear. Since fabs naturally operate several layers removed from the end-consumer, we should expect the good times to be great, but the bad times to be harsh. For fabs more distant from the end-consumer, the bullwhip effect will be even greater (E.g. telecommunications device fabs, which sell to network equipment manufacturers, which sell to telecommunications infrastructure companies, which sell to telecommunications providers, which sell to end-consumers). Fabs need to make money during the good times and try not to get burned badly by the downturns (e.g. writing off inventory that becomes obsolete).

You may only read part I of ***Clockspeed*** (parts II and III deal with supply chain design and strategic decision-making), but you'll emerge with a better understanding of how your company fits into its extended supply chain, and how the behavior of this chain impacts your company and your future.

The Mythical Man-Month by Frederick Brooks, Jr.

This book review was written by Frank Chance

Introduction

Published in 1975, ***The Mythical Man-Month*** is a classic text on software development. A 1995 edition includes several new chapters of interest, but the original essays remain the heart and soul of the book. In this book, Brooks tackles the question of how to organize and manage large-scale programming projects. These are projects that require hundreds or thousands of programmers, and result in millions of lines of code (think SAP, or Oracle's database engine, or Windows 2000). The book is organized as a series of concise essays. In this review I'll discuss the opening essay -- one of my favorites.

The Tar Pit

Brooks opens his first essay with a comparison between the tar pits of prehistory and large-system programming:

> "In the minds' eye one sees dinosaurs, mammoths, and sabertoothed tigers struggling against the grip of the tar. The fiercer the struggle, the more entangling the tar, and no beast is so strong or so skillful but that he ultimately sinks. Large-system programming has over the past decade been such a tar pit, and many great and powerful beasts have thrashed violently in it. Most have emerged with running systems--few have met goals, schedules, and budgets. Large and small, massive or wiry, team after team has become entangled in the tar. No one thing seems to cause the difficulty — any particular paw can be pulled away. But the accumulation of simultaneous and interacting factors brings slower and slower motion. Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it."

Remember, these words were written in 1975. Do they still apply today? Consider Windows NT 5.0. First scheduled to appear in 1997, it slipped to early 1998, to late 1998, then to 1999 (whence it was renamed Windows 2000). Here are some public estimates:

- 5,000 programmers.
- 35,000,000 lines of code.

Clearly, NT 5.0 qualifies as a large-system programming project. And just as clearly, Brooks' tar pit is as prevalent today as in 1975!

Let's carry on with the NT 5.0 example. Assume the worst case, that all 35,000,000 lines are new code. It's reasonable to assume that the development started in roughly 1994. So we have:

- 5,000 programmers x 5 years = 25,000 programmer-years.
- 35,000,000 lines of code / 25,000 programmer-years = 1,400 lines per programmer-year.

If you are a programmer, or have ever taken a programming class, this number (1,400 lines per year) seems amazingly low. Most of us have hacked together a bit of code that approaches one thousand lines in just a day or two. How could it possibly take a Microsoft programmer an entire year to complete 1,400 lines?

Two possibilities spring to mind:

Microsoft hired 5,000 incompetent programmers to develop NT 5.0.

Or it's a lot harder to write a large-scale programming systems product than to hack together a single program.

Brooks would argue that the latter answer is the correct one. He starts by defining terms:

(1) A program

An individual program is the result of our two-day programming binge. It is ready to run by itself, on the machine where we wrote the code. If we add documentation, generalize the code, write test cases, and make the code maintainable by a general programming audience, we have:

(2) A programming product

Alternatively, if we take our program and completely define its interfaces according to a predefined specification, and test its interaction with a large number of other components, we have:

(3) A programming system component

And if we do both (add documentation, generalize the code, write test cases, make the code maintainable, define its interface, test its interactions), we have:

(4) A programming systems product component

Brooks uses a 3x rule of thumb for the work required in taking each of these steps:

(2) = 3 times the effort of (1)

(3) = 3 times the effort of (1)

(4) = 9 times the effort of (1)

Or, in words, development of a standalone program requires only 1/9th the effort required to develop a component in a programming system.

Returning to our Microsoft example, if we apply this 9x factor to the 1,400 lines per programmer-year productivity measurement, we get 12,600 lines per programmer-year (e.g. if we took each of those programmers and set them to work in isolation, hacking away on a single program). In a separate essay, Brooks quotes a manager who found that, on average, his programmers only spent half of their time developing -- the rest was consumed by paperwork, meetings, and various other tasks. Factoring this into the Microsoft example, we arrive at 25,200 lines per programmer-year. And the Microsoft programmers begin to look more respectable.

In another measure of just how little has changed since 1975, Brooks quotes an estimate of 1,000 lines per programmer-year. If the 1,400 lines per programmer-year quoted above is accurate, it represents a productivity gain of just 1.75% per year for the 20 years between 1975 and 1995. This result confirms another of Brooks' hypotheses — that productivity of programmers is relatively constant, no matter the language used for development. Therefore, the real productivity gain comes from moving to a higher-level language, where each line represents more actual work.

Although aimed at large-system projects, Brooks' commentary often applies more generally. For example, this first essay closes with sections titled "The Joys of the Craft" and "The Woes of the Craft". Under woes, he discusses the problem of obsolescence:

> "... the product over which one has labored so long appears to be obsolete upon (or before) completion. Already colleagues and competitors are in hot pursuit of new and better ideas. Already the displacement of one's thought-child is not only conceived, but scheduled. This always seems worse than it really is. The new and better is generally not available when one completes his own; it is only talked about. It, too, will require months of development. The real tiger is never a match for the paper one, unless actual use is wanted. Then the virtues of reality have a satisfaction all their own."

Summary

Brooks' essays cover a variety of the challenges inherent to large-system programming, but are useful reading for anyone involved in software development. The name-sake essay (***The Mythical Man-Month***) discusses the indivisibility of many programming tasks, and why this makes the addition of manpower to a software project a futile effort. My other favorites are "Aristocracy, Democracy, and System Design" (a discussion of conceptual integrity) and "Plan to Throw One Away" (on the benefits of explicitly planning for multiple releases before shipment). Some questions have been made obsolete by technological advances, for example the discussion of how to distribute type-written documentation among a large team. You will be surprised, however, at how many of the problems that Brooks faced still frustrate us today. As an added benefit, the brevity and clarity of Brooks' writing make it a delightful read. If you are a programmer, if you work with programmers, or if you manage programmers, you should read this book.

For a quite different view of systems development, see ***The Cathedral and the Bazaar***.

Next - The Future Just Happened by Michael Lewis

In this book, Michael Lewis explores how the Internet has encouraged changes in the way people live their lives. While he doesn't view the Internet as causing revolutionary change, he does see it as a tool that facilitates certain trends already in progress. These trends include the flattening of hierarchies, the development of closer relationships between insiders and outsiders, and the increased influence of younger and younger individuals in technological areas.

Lewis spends considerable time describing three teenagers who represent different instances of trends exaggerated by the availability of Internet access. The first is Jonathan Lebed, a New Jersey teenager who was fined by the SEC for "manipulating" the stock market. The second is Marcus Arnold, a Southern California teenager who became the top-rated legal advisor on a website called AskMe.com, consulted by thousands of adults. The third is Daniel Sheldon, a British teenager who became a leader in the peer-to-peer computing movement and advocates the notion that all intellectual property should be free. Lewis spends considerable time interviewing these youths, with a goal of understanding how they became revolutionaries, and what trends they exemplify that were already in place in society (e.g. the democratization of the stock market, the fall of lawyers and other professional from their elite pedestals, and the loss of copyright protection that follows from digital media). For all three teenagers, the Internet allowed them to use masks to reinvent themselves, and challenge figures of central authority. A British rock band, Marrillion, is another example.

The book is a bit fragmented. The two last sections are about how the Internet is changing the traditional models of television advertising and polling, and about the backlash against technology by certain technology pioneers, while interesting, are very distinct from the first two sections. Overall, however, the use of stories about and interviews with real people makes this book a fun and interesting read. Lewis has a knack for making the reader think about the larger issues, without laying down the law about what the reader "should" be thinking.