Project Management

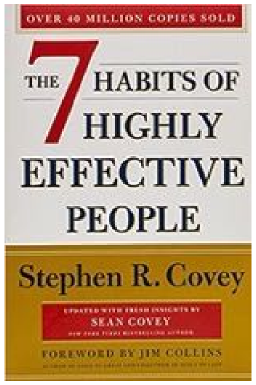
Main Ideas:

- Crystal clear understanding of the requirements. Involve all business stakeholders into the planning process. Make sure they agree on the plan.
- Crystal clear understanding of priorities: what is important – and what is "nice to have".
- Concentrate only on important features, sacrifice the rest.
- Simplifty, simplify, simplify.
- Split big job into small pieces, small steps.
- Grow project as a child from few features to full-featured product.
- Military hierarchical management is not effective. Respect workers, leverage their brains, give them freedom to be effective, to communicate and collaborate.
- Give people tasks that match their abilities.
- Create safe and engaging atmosphere.

Project Management

Table of Contents:

Begin With the End in Mind


OVER 40 MILLION COPIES SOLD
THE 7 HABITS OF HIGHLY EFFECTIVE PEOPLE
Stephen R. Covey
SEAN COVEY
FOREWORD BY JIM COLLINS

Habit 2: Begin With the End in Mind
from "The 7 Habits of Highly Effective People" by Stephen Covey

All 7 habits:

1. be proactive
2. **begin with the end in mind**
3. put first things first
4. think win-win
5. seek first to understand, then to be understood
6. synergize
7. sharpen the saw

THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Eric Ries

Ries has experienced **two startups failures**, and attribute the error in both cases as **"working forward from the technology instead of working backward from the business results you're trying to achieve."**

Main ideas in his book:

- One **must begin with the customers** in the form of **interviews and research discovery**
- Building an **MVP (Minimum viable product)** and then **testing and iterating quickly**
- Ries also recommends using a process called **the Five Whys**, (Why ..., Why ..., Why ..., Why ..., Why ...) a technique designed to reach the core of an issue

Splitting Big Problems Into Small Doable Steps

When I was a student (in Moscow, Russia, 1975), we were sent to suburbs to collect vegetables (carrots). The task was simple. We were separated in pairs, and each pair had a ~120 meters long row of carrots to collect in 1 day. There was 6 pairs starting side-by-side like runners at a sport competition. At the end of the day nobody finished even 50% of their norm. It was taking 2 days to finish the norm.

This continued for several days, and then our manager (who was a 1st year post-graduate student) made a very simple and very magical change. He simply split one long row into 6 short rows. Each pair of students now received a set of 6 rows 20 meter each.

```
All 6 pairs before:

111111111111111111111111111111111111111111111111111111111111111111111111
222222222222222222222222222222222222222222222222222222222222222222222222
333333333333333333333333333333333333333333333333333333333333333333333333
444444444444444444444444444444444444444444444444444444444444444444444444
555555555555555555555555555555555555555555555555555555555555555555555555
666666666666666666666666666666666666666666666666666666666666666666666666

All 6 pairs after

111111111111222222222222333333333333444444444444555555555555666666666666
111111111111222222222222333333333333444444444444555555555555666666666666
111111111111222222222222333333333333444444444444555555555555666666666666
111111111111222222222222333333333333444444444444555555555555666666666666
111111111111222222222222333333333333444444444444555555555555666666666666
111111111111222222222222333333333333444444444444555555555555666666666666
```

You can ask – what's the difference?
Well, this day the whole field was finished by 2 pm !

It took only 5 hours to do what before was taking 2 full days (16 hours) !
This is 3-times increase in productivity.
Without any incentives !

I can tell you how it felt.

You start working on one row - and pretty soon you see that you can reach its middle in a short time. So you decide to take a break at the middle.

But when you reach it - you could see that it is actually very easy to finish the row - and then take a break. And when you finished the 1st row - you see that now only 5 left. So you estimated how long it took you to finish the previous row - so you can tell when you will finish the next row. etc.

The work started to feel manageable. And everybody rather enjoyed the process. It was addictive.

Do you want to **increase effectiveness 3-fold**? Then plan your work as a set of simple manageable self-contained and self rewarding steps.

Split Project into "Buckets"

I remember a wonderful course by Steve Manning:
**"How To Write A Book On Anything
In 14 Days or Less… GUARANTEED!!"**

His approach was:

1. Create a title
2. Create a list of ~10 chapters
3. Create a list of ~10 topics (buckets) inside each chapter (each topic requires approximately ¾ of a page of text) – so you have total of ~100 "buckets".
4. Fill the buckets with some content.
   Do NOT edit. Do not return back.
   Just type something into those "buckets".
   Fill them in random order.
5. Edit buckets one at a time in reverse direction (start from the end of the book and move towards the beginning). This way each bucket could stay on its own.
6. Do more editing ...

You can use similar approach when creating any kind of documents (PowerPoints, Project Plans, etc.).

Similar approach is used by teachers in schools with "slow" kids.

If you ask a child to draw a picture of a ship, he will not be able to do it and will get upset.

But you can make a simple step-by-step plan for the child:
.. get a box of pencils
.. sharpen the pencils
.. get a sheet of paper
.. get a book with pictures of ships
.. select a picture you like
.. etc.

The child can now follow this plan.
Result – he successfully creates a picture.
And he is happy and proud.

Different "Bucket Sizes" for Different People:

Some team members can execute the high level command ("draw a ship") all by themselves. They can figure out the steps – and execute them.

But most of people need to receive a step-by-step plan.
Some people require a very detailed plan.

A good manager should **calibrate the size of the steps individually for each team member** to make them effective workers.

First Principles Thinking

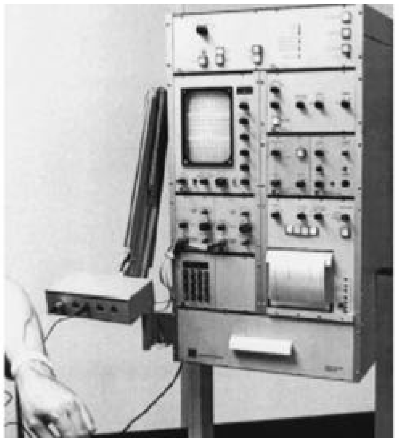
First principles thinking (FPT) is thinking from scratch, actively questioning every assumption. FPT means to reverse-engineer complicated problems, break them down into basic elements, and then reassemble them from the ground up.

FPT requires courage and going against rules (fresh thinking):

- "Never be afraid of doing tasks you are not familiar with.
  Noah's Ark was built by an amateur. Professionals have built the Titanic."
- "The safest way to go on strike and achieve nothing
  is to do everything by existing rules."

THINK LIKE MUSK
How Elon Musk Uses First Principles Thinking to Invent the Future (And How You Can Too)

The first time I've heard about FPT was from an interview with Elon Musk. He uses this type of thinking to design a cheap rocket from scratch, to re-engineer electric cars, to everything he does.



In 1989-1991 I participated in designing a first in Russia portable myograph. As you see, the device was very small and integrated with a laptop computer. Compare it with a prototype which was the size of a table (b/w photo on the left).

We managed to make it small by thinking from scratch and moving as much functionality as possible from hardware to software.

We didn't have a lot of resources and couldn't do things right. But **we knew the key requirements**, and all other pieces eventually fell into place.

Our competitors having literally 100 times more resources (money, people, etc.) had never finished the project. Because they wanted to make things right. So they drew a plan, distributed responsibilities, debated and agreed on a 5-year budget - and **they have never (NEVER) delivered a final product**.

Simplicity

"There are two ways of constructing a software design; one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult." - C. A. R. Hoare

"Make everything as simple as possible, but not simpler. " - Albert Einstein (1879-1955)

"Thank you, Lord, for making all necessary things simple, and all complicated things unnecessary" - H. S. Skovoroda (1722-1794)

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius -- and a lot of courage -- to move in the opposite direction." - Albert Einstein (1879-1955)

KISS principle - Keep It Simple, Stupid - is largely used by military, and is attributed to Clarence Leonard 'Kelly' Johnson (1910-1990), who worked for Lockheed Martin

"You can see that [with a simpler toolkit] the amount of extension programming goes up considerably. What you don't see is that the total implementation effort may be much lower because the underlying toolkit is much simpler. There the programmers need spend much less time reading documentation, fitting their new software into the old, etc. Sometimes less is more. " - Philip Greenspun, https://philip.greenspun.com

Prioritization is the key

Warren Buffett 5/25 rule:
Write down 25 tasks you think are important.
Choose just the top five – and do only them.

Your success depends on being focused on the most important tasks/features.
Your success depends on cutting off all tasks/features which are secondary.

Common reason of failure is that people spend time on doing things which shouldn't have been done at all in the first place.

Many projects failed because their architects failed to make things simple. They tried to make things **right**. As a result they have built systems which were never quite operational and couldn't survive change.

Find simple ways to do things.

Try to find a way to complete project in two weeks. If you can't see that you can complete the project in 2 weeks - don't do it immediately. Think first.

Because if you think it takes 2 weeks - it will take 2 months.
And if you think it will take 2 months - it will take a year.
And in the middle you or your boss will realize that it should be done differently - thus you will never finish it.

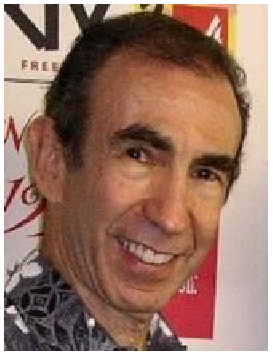
F**k it, we don't need it

Simplify the design, sacrifice some features

"**Do we really need it ?**" Often 10% of features takes 90% of your programming time. By sacrificing them you may make your project **10 times simpler**.

Same principle can be applied to the features of programming architecture. Trying to make perfect code may make your project **10 times more complex**.

The great marketer **Joe Sugarman** has a little formula for judging any project he gets involved in. He calls it "**ELF**" (**Easy, Lucrative, and Fun**). Make your project an **ELF** project.


Joe Sugarman

Unix – Example of Simplicity of Design

Unix is the dominating OS (~2 Bln android devices, ~1 Bln iOS devices, ~1 Bln servers in clouds, 100 Mln Macs, etc.).

Ken Thompson has created Unix OS in 1969.
Nowadays he works at Google as one of creators of the "Go" language:
- https://www.youtube.com/watch?v=sln-gJaURzk -

In this article (http://www.linfo.org/thompson.html) you will learn how **Ken Thompson** has developed the UNIX operating System in assembly language (to assist himself in playing and creating computer games) in ~1969, then created the B-language, and later rewrote it's kernel in C-language developed by **Dennis Ritchie** in ~1972
First UNIX was running on computer with only 4K of memory!

Then **Thompson** returned to UCB (University California Berkeley) and while being there in 1975-76, he introduced people there to Unix, which started the UCB clone of Unix (BSD = Berkeley Software Distribution).
This later became the foundation of Mach OS for **Steve Job's** NeXT Station, which was later acquired by Apple, and is the heart of modern MacOS and iOS.

Linux is an independent open source POSIX implementation of Unix.
Android OS is a Google's version of Linux (android phones & tablets).
1980s - **Richard Stallman** creates a free software movement (GNU project) which led to development of numerous software tools.
1991 - **Linus Torvalds**, young student at the University of Helsinki in Finland, releases (posts on the Internet under GNU license) the first version of Linux's kernel (Ver. 0.02), which he developed as a hobby.

Ken Thompson
(born 1943)

Linus Torvalds

Unix – a toolbox vs finished application

Unix was designed as a minimalistic set of small basic utilities and ways you can combine them to make more complex systems. Simplicity allowed it to be very reliable, extensible and maintainable. And eventually to win against all other operating systems.

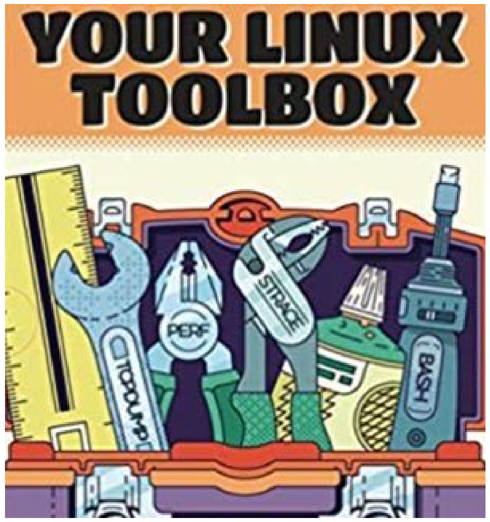
You have to decide very carefully whether you really need to deliver a finished application, or if the client will be much happier with a toolbox (set of tools and modules) giving him functionality he needs.

It is ~100 times more difficult to deliver a finished application, because you have to test it on all supported platforms, provide documentation and customer service, work out all small details. On the other side, finished application is not flexible and may not fit customer needs. The bigger the client - the more flexibility he may require.

Thus it makes no sense to spend too much time on making a turn-key application for a big client. Instead you may design a set of modules which allow to easily construct custom applications for the client. Working with many clients you will add more modules to the system making it more valuable to your market.

Each individual module can be small, manageable and reliable. The custom application as a whole will be probably built by the client, so you don't have to provide customer support on it's quality.

Thus by shifting your goal from making a compiled product to making a toolbox, you made your work 10..100 times simpler and easier - and you made yourself 10 times more valuable to the marketplace.



Dinosaurs

Big complex systems remind me of dinosaurs. Do you remember what happened to them? And who is now dominating the Earth?

Distributed Systems

In big organizations you simply can't get all departments to use the same programming language, the same systems, the same versions of software.

All you can do is establish some pretty liberal **general guidelines** and **rules of communication**.

Simple vs "right"

Today (as always) we face 2 conflicting requirements:
- systems get larger and more complex
- systems need to change more often - and changes should be made faster

When things change faster and faster - **simplicity becomes a "live or die" requirement**.

In many situations you literally don't have enough time to make things "right" or complex. Or even "completely finished".

What do you prefer - a simple system which does the job - or a system which is architectured "correctly", - but never quite works, and can't be adapted to your requirements fast enough?

So don't try to make things right.
Make things simple instead.

Simplicity vs Politics

Be carefull when offering simple solutions to managers of big organizations.

An individual developer usually will select a simple solution.

But a manager at a big organization will, on the contrary, almost always select the political solution, which is usually more complex and expensive.

Please note, that managers are NOT stupid. They are doing the **right political move** in order to survive and grow in the organization, get more people under them, get promoted, increase visibility and weight of their department, etc. Yes, they are doing the **right thing** for them !

This **right thing** may not be the best thing for the organization or the product, but managers will never admit that. And they usually will get promoted despite losing millions of dollars and failing to build effective systems.

19 General Manager
18 2 Years | Deputy General Manager
17 2 Years | Asstt General Manager
16 3 Years | Sr. Manager
15 4 Years | Manager
14 4 Years | Deputy Manager
13 4 Years | Sr. Engineer / Officer
12 3 Years | Engineer / Officer

Growing a Project as a Child

Let's say you started a web site with just one page.
Then you added some more.
You working one page at a time.
Each page is a finished product.
Together pages make a system - your web site.

This system growth is stable.
You always succeed, because each step is simple enough and rewarding enough.

This is example of evolutionary development.
You start from a simple bare bones single function utility.
Then you write another one. And another.
You add features. You combine them together - and finally you have a product.

You allow the program to grow as a child.
When the child is born - you don't know exactly what it will grow into.

May be the product itself will not be a success - but one of the components will.

The art is to structure the development process into a set of easy and rewarding small steps.

The art is to structure the development process so that the product starts to be useful at early stages. And then it grows and improves with the feedback from users.

If software is simple and useful - it can start with a 2-week project - and then grow as a child.

If software is not simple or not useful - it will be very difficult for it to grow or even survive, regardless of how much time and money were originally invested, or how "right" was its design.

I wrote this - and then found that I was not the first who formulated this. See for example www.dreamsongs.com/WorseIsBetter.html - "Worse Is Better" by Richard P. Gabriel.

Hacker vs Architect

Example. Let's say you need a report. So you asked two programmers (John Hacker and Bob Architect) to create it.

John Hacker hacked a script in one hour and emailed you the report.

You looked at the result - and realized that your original requirements were not correct. So you asked to make changes, which John emailed you in the next 10 min.

After going back and forth 5-10 times you were completely satisfied.

You could customize the reports providing some simple parameters on the command line or in a short text file. John sent you all the scripts along with a short README instructions text file.

He also put the scripts on the web server and made a simple web page for you to run the report in the browser – and either download it – or receive via email.

Bob Architect, on the other hand, is a true believer that if you do things right - it will save you effort in the long run. (**wrong**)

So he started with drawing UML diagrams using Rational Rose or TogetherJ to make proper Object Oriented design of the "report engine". He also decided to make a proper GUI interface.

At the time when John Hacker went through 10 revisions, Bob Architect was still busy thinking about distributing responsibilities between his classes and interfaces, and was very proudly telling you about his elegant design of "request manager", "cache manager" and "report manager".

Finally he made the application work - but **it took him 10 times longer**. And when he finally finished, you already needed a different report. So you asked him to change the reports. He said that it will take him long time because your new requirements don't fit well into his object model, so he will have to start from scratch.

Does all this sound familiar?

In this scenario, first programmer sacrificed some features (GUI, OO design), which allowed him to complete the job much faster, and then to use user's feedback to make multiple changes to make the "product" good and easily maintainable.

Create "Clickable Dummy"

Projects have three basic components:
cost, schedule, scope

But figuring them out (specifying what to do, the actual scope, cost, and schedule) may take 30-50% of the project.

Jesse Erlbaum recommends to agree with clients on a two-phase approach:

- 1st phase - 30-50% of cost/effort
  preparing specifications and estimate
  (create clickable "dummy" of an app.)
- 2nd phase - 70-50% of cost/effort
  actually implementing

Jesse Erlbaum
Erlbaum Group,
New York
http://www.erlbaum.net

From Waterfall to Agile

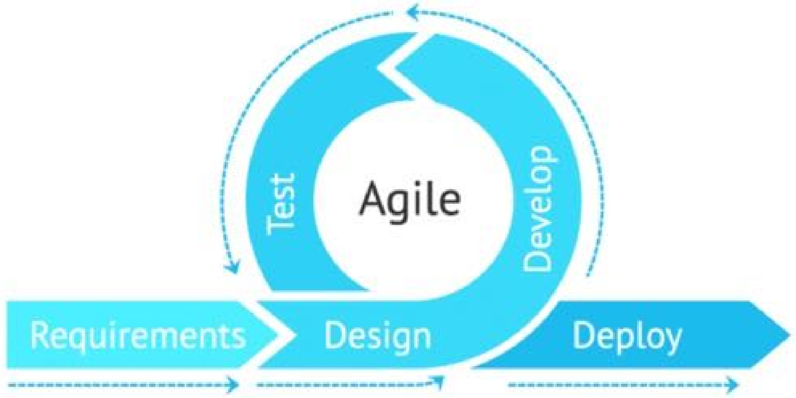
Waterfall

```
Analysis
  Requirements
    Design
      Development
        Testing & Integration
          Deployment
            Maintenance
```

The Waterfall methodology has its origins within the manufacturing and construction industries, ... however, the term “Waterfall” wasn't used.

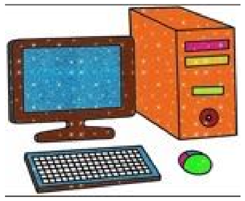
A paper written in 1976 by T.E. Bell and T.A. Thayer is when the term may have been first used.

Agile



In 2001, a team of 17 visionary software developers held a meeting in Utah to discuss industry problems and possible solutions.

They later created what is known throughout the industry as the **Agile Manifesto**.

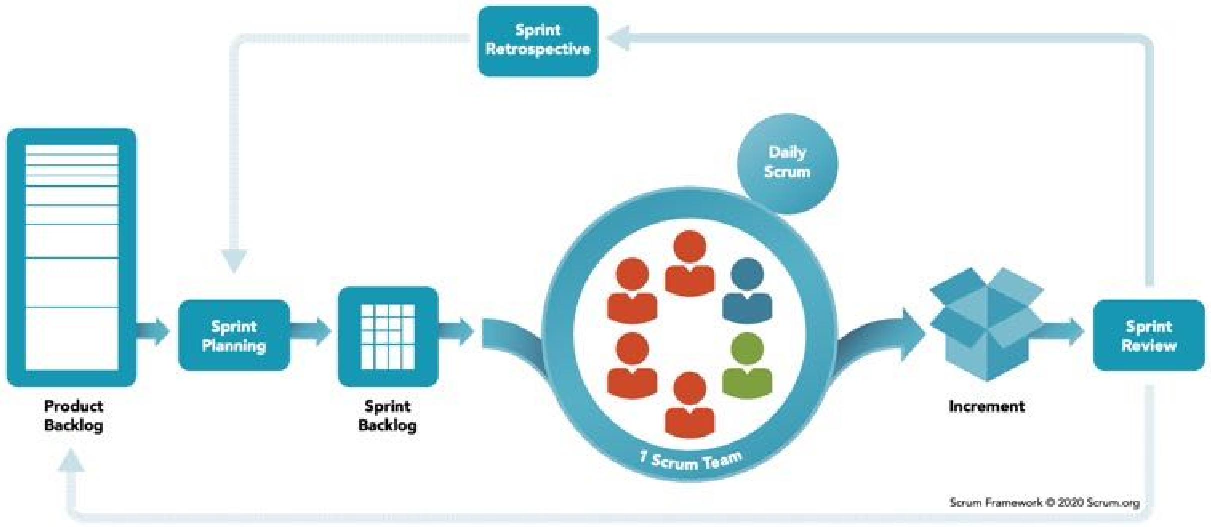


Scrum

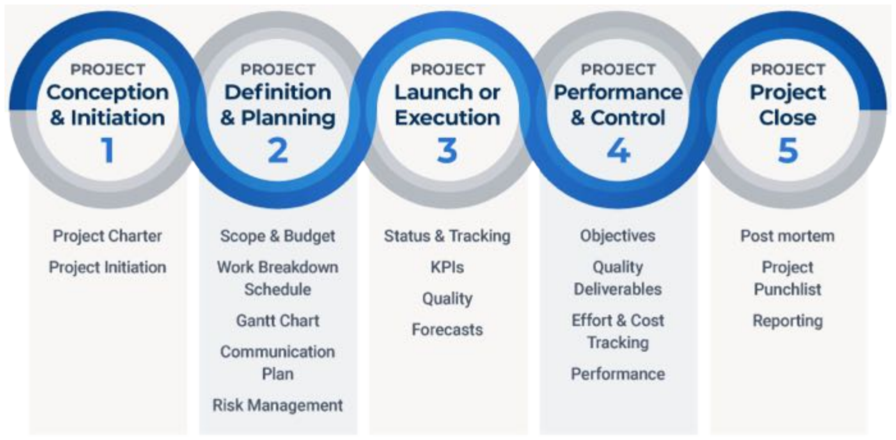
A scrum (short for scrummage) is a method of **restarting** play in rugby football that involves players packing closely together with their heads down and attempting to gain possession of the ball.

Scrum is a framework for developing, delivering, and sustaining products in a complex environment.
It is simple and effective.
Importan components are:

- product backlog
- planning sprints (sprint backlog)
- daily scrum meetings
- sprint review and retrospective



Project Management Process



Project Management:

- Formal project management structure (processes, procedures, and tools)
- Invested and engaged project sponsor(s) (executives promoting the project)
- Clear and objective goals and outcomes
- Documented roles and responsibilities
- Strong change management (protect from "scope creep")
- Risk management
- Mature value delivery capabilities (project tools, processes, and procedures)
- Performance management baseline (for cost, schedule, and scope)
- Communication plan

DevOps

What is DevOps

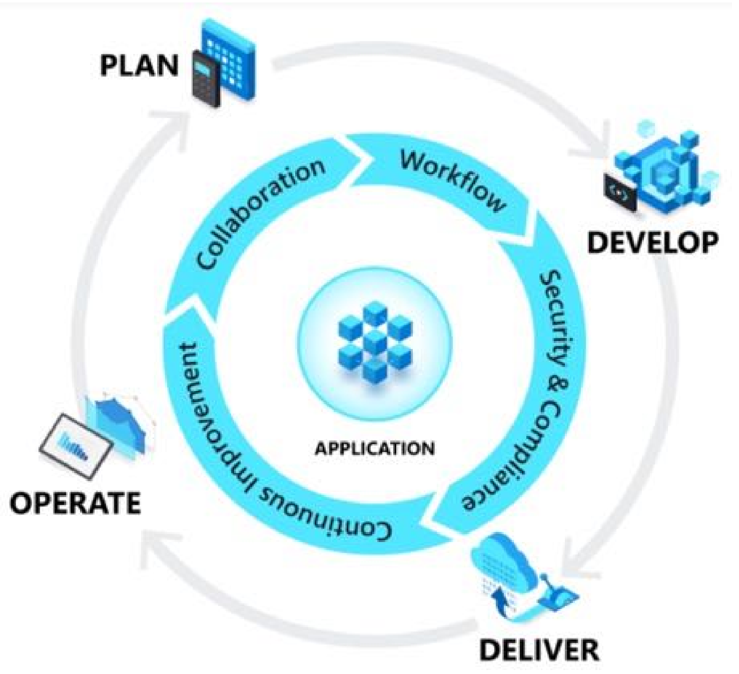
DevOps = Development (Dev) + Operations (Ops)

DevOps is the union of people, process, and technology.

DevOps enables formerly siloed roles
- development
- IT operations
- quality engineering
- security

to coordinate and collaborate to produce better, more reliable products.

By adopting a DevOps culture, practices and tools,
teams gain the ability to better respond to customer needs,
increase confidence in the applications they build, and achieve business goals faster



Azure DevOps - a Microsoft product that provides version control, reporting, requirements management, project management, automated builds, testing and release management capabilities. It covers the entire application lifecycle, and enables DevOps capabilities

DevOps and AWS
Tooling and infrastructure resources for DevOps practitioners

AWS

Google

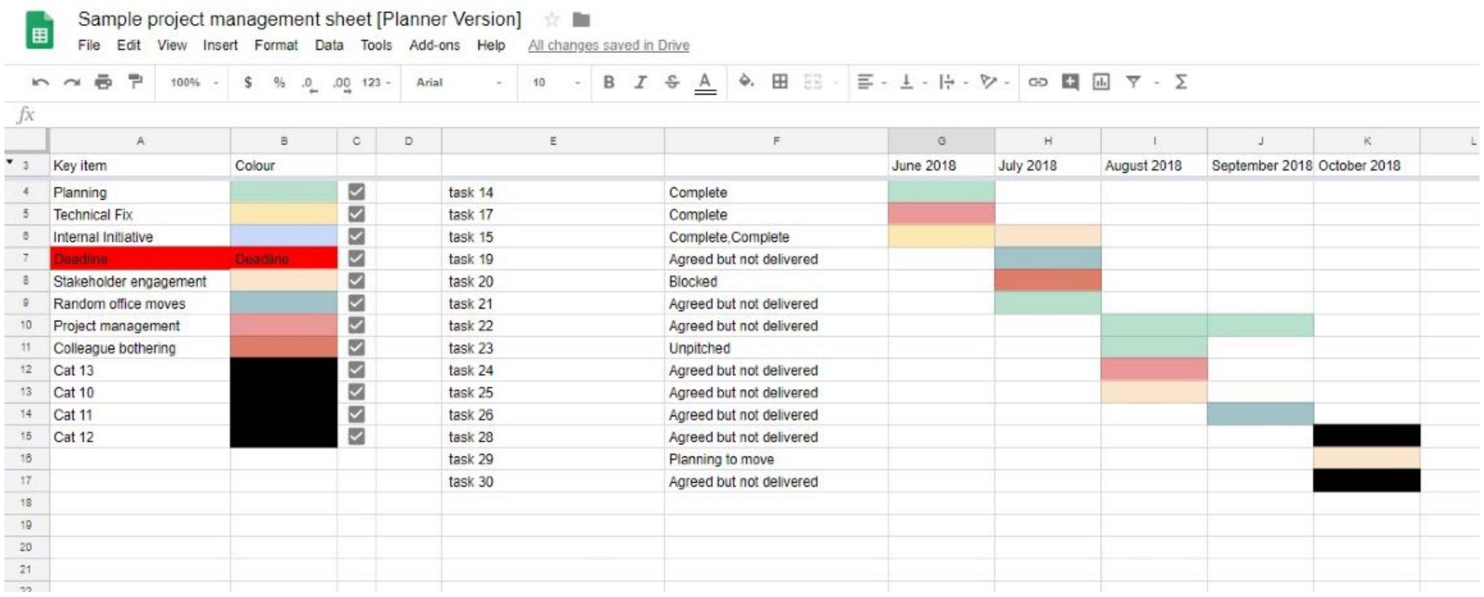
Simple PM Tools

There are many DevOps tools and bug-tracking tools,
for example, Jira (https://www.atlassian.com/software/jira)
or Azure DevOps (https://dev.azure.com).
See long (~40) list here:
- https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

Note that for small scale projects people successfully
use Google Docs and Google Sheets:
- https://moz.com/blog/visualising-time-using-google-sheets

Some software similar to JIRA:

- ClickUp
- Binfire
- Basecamp
- Pivotal Tracker
- Asana
- Clubhouse
- Trello
- ProofHub
- Kanbanize
- Notion
- Wrike
- Bitrix24


Sample project management sheet [Planner Version]
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive
100% $ % .0 .00 123 Arial 10 B I S A
fx

	A	B	C	D	E	F	G	H	I	J	K	L
3	Key item	Colour					June 2018	July 2018	August 2018	September 2018	October 2018	
4	Planning	🟩	✓		task 14	Complete	🟩					
5	Technical Fix	🟨	✓		task 17	Complete	🟥					
6	Internal Initiative	🟦	✓		task 15	Complete,Complete	🟨	🟧				
7	Deadline	Deadline	✓		task 19	Agreed but not delivered		🟦				
8	Stakeholder engagement	🟧	✓		task 20	Blocked		🟥				
9	Random office moves	🟦	✓		task 21	Agreed but not delivered		🟩				
10	Project management	🟥	✓		task 22	Agreed but not delivered			🟩	🟩		
11	Colleague bothering	🟥	✓		task 23	Unpitched			🟩			
12	Cat 13	⬛	✓		task 24	Agreed but not delivered			🟥			
13	Cat 10	⬛	✓		task 25	Agreed but not delivered			🟧			
14	Cat 11	⬛	✓		task 26	Agreed but not delivered				🟦		
15	Cat 12	⬛	✓		task 28	Agreed but not delivered					⬛	
16					task 29	Planning to move					🟧	
17					task 30	Agreed but not delivered					⬛	
18												
19												
20												
21												

Vision and Enthusiasm

Successful people tend to have an attitude which can be expressed as following: **"We will build it – and it will be great !"**

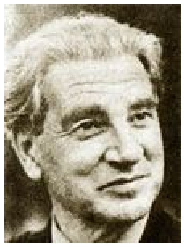
In 1951 Nikolai Pertsov was appointed as a Director of **White Sea Biological Station of Moscow State University**. When he arrived at the place designated as Belomorskaya Biostation MSU, there was nothing there except for a small shed – a "cabin" and several tents.

Nikolai Pertsov launched intense activity. **He didn't have the necessary funds**. **But he was good at captivating people with enthusiasm**. He has built a laboratory and homes, got hold of the present fleet of several boats, laid electric and telephone lines through miles of "taiga" (forest), provided necessary scientific and educational equipment.

Nowadays hundreds of students and staff are comming to the bio-station every year. Dozens of books, thousands of scientific articles, hundreds of PhD degrees, thousands of trained professionals - this is the result of the activity led by Nikolai Pertsov's bio-station.

The really interesting fact about the history of the station is that **almost all construction work was done by volunteers**. Nikolai Pertsov had an amazing skill of making people enthusiastic about the project and making them wilingly work on it.

- https://en.wikipedia.org/wiki/White_Sea_Biological_Station
- https://persona.rin.ru/eng/view/f/0/21543/pertsov-nikolai-andreevich


Nikolai Pertsov
(1924–1987)

THE NIKOLAY PERTSOV
WHITE SEA BIOLOGICAL STATION
Dining Hall
Boat Station
Doctor
Bathhouse
Pier
Buildings

Unique Abilities

Unique Abilities:

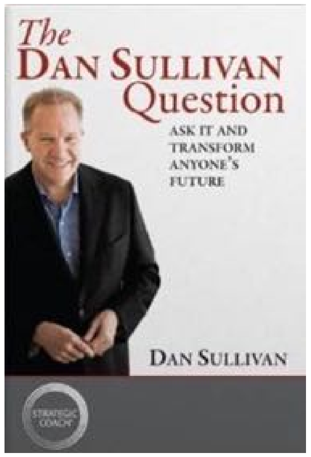
It is very important to align the roles of team members with their personal skills, preferences, and career goals.

Some people like to work with software, others like to do system configurations and networking/security. Some people like math and data analysis, others like graphical design. Some people enjoy creating architectures and making PowerPoint presentations, others are great at managing people.

It is a good idea to ask a "Dan Sullivan Question":

"If we were having this discussion three years from today, and you were looking back over those three years, what has to have happened in your life, both personally and professionally, for you to feel happy with your progress?"

Answer to this question will help you to "match" a person with the roles in the team. This will make team members more effective and happier.

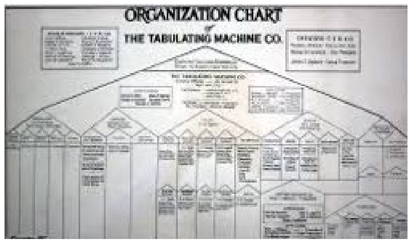

The Dan Sullivan Question
Ask it and transform anyone's future
Dan Sullivan
Strategic Coach

The Dan Sullivan Question
by Dan Sullivan, 2009

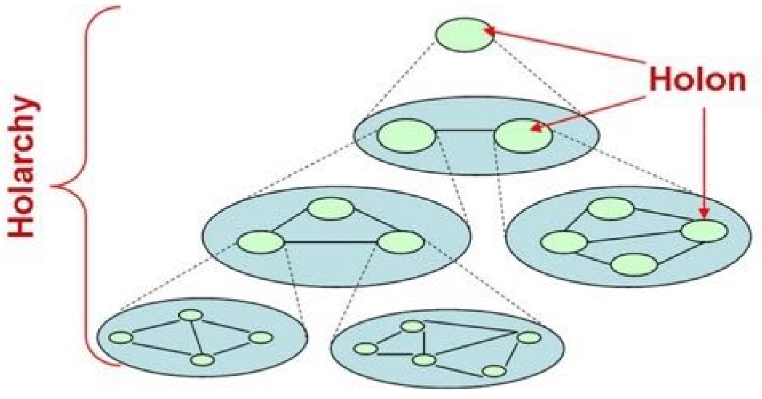
If a person is asked to do a task which he can't do or doesnt' like, it will be bad both for him and for the task.

Self Management, Holacracy

Silos


ORGANIZATION CHART of THE TABULATING MACHINE CO.

Hierarchical Pyramid



- **Holacracy** is a method of decentralized management and organizational governance.
- It claims to distribute authority and decision-making through a holarchy of self-organizing teams rather than being vested in a management hierarchy.
  - https://www.holacracy.org
- **Holacracy** is opposed to **old hierarhical organization** developed for manufacturing and construction businesses. Nowadays every industry today involves complex knowledge work. But IT and Data work doesn't fit well into hierarchy. Even modern factory workers need to contribute in ways that prior generations couldn’t imagine.
- **Holacracy** can be seen as a greater movement within organisational design to cope with increasing complex social environments, that promises a greater degree of transparency, effectiveness and agility.
- **GlassFrog** is the official software to support and advance your Holacracy practice. Made by HolacracyOne, it's the most robust tool available.
  - https://www.glassfrog.com/

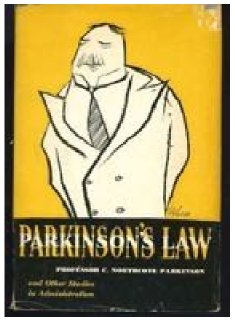
A **holarchy** is a connection between **holons**, where a **holon** is both a part and a whole.

The term was coined in Arthur Koestler's 1967 book "The Ghost in the Machine".

Holarchy is commonly referred to as a form of hierarchy; however, unlike hierarchy, it doesn't have an absolute top and bottom.

Note:
Many companies use non-hierarchical approach. For example, Facebook has hierarchy, but also uses product teams, which span multiple silos (multiple departments).

Parkinson Law



Book "Parkinson's Law"
by C. Northcote Parkinson
Classic - witty, brilliant.

Professor Northcote Parkinson (1909-1993) - British historian, author, and formulator of "Parkinson's Law," the satiric dictum that "**Work expands to fill the time available for its completion**."

"Administrators make work for each other, he said, so that they can multiply the number of their subordinates and enhance their prestige."

His second law was intended as a jibe at government functionaries, who he thought were inclined to expand their own ranks indefinitely, so long as taxes could be raised.

Written in a deadpan but mercilessly funny style, Parkinson's Economist essays were issued in book form in Parkinson's Law, or The Pursuit of Progress (1958). Apart from the books that made him famous, Parkinson wrote numerous historical works, including the critically acclaimed The Evolution of Political Thought (1958).s.

Here is the essense of some of Parkinson's main laws:

- Work expands so as to fill the time available for its completion. This law has many consequences. For example, it shows why the British Colonial Office has grown in number of employees as the actual number of colonies declined - so that it employed more people when the number of colonies had been reduced to zero than when they were at their highest number.

- Expenditure rises to meet income. (Unit costs of public services tend to increase to consume the available funding. Data expands to fill the space available for storage. Network traffic expands to fill the available bandwidth, etc. Unfortunately this law is not applicable to our wallet nor to its contents. Pity.)

- Expansion means complexity, and complexity - decay.

- Policies designed to increase production increase employment; policies designed to increase employment do everything but.

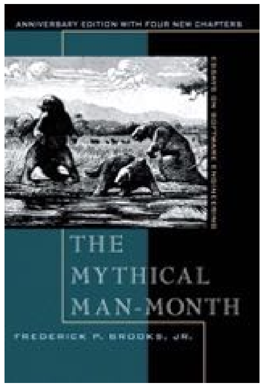
- Democracy equals inflation (people vote for higher pay rather than for increased production)

- Delay is the deadliest form of denial.

- The matters most debated tend to be the minor ones where everybody understands the issues.

- Committees become less effective after they grow larger than 8 members.

Brooks Law



The Mythical Man-Month - by Frederick Brooks

Classics - based on experience from 1960s at IBM.
A required reading in most SE (Software Engineering) and CS (Computer Science) courses.
This book has become a bible for software developers.
Brooks received the Turing Prize in 2000 (often called "the Nobel Prize of Computing").

One of the main ideas (also called "**The Brooks's Law**"):

"Adding manpower to a late software project makes it later. Artists cannot be rushed, and too many cooks will surely spoil the broth."

See good summary here:
https://en.wikipedia.org/wiki/The_Mythical_Man-Month

Main ideas in the book:

- The mythical man-month (adding more people will make the project later)
- No silver bullet
- The second-system effect (tend to make it too complex)
- The tendency towards irreducible number of errors
- Progress tracking (incremental slippages)
- Conceptual integrity (separate architecture from implementation, keep things simple)
- The manual (document external specs)
- The pilot system (throw-away system)
- Formal documents (needed)
- Project estimation (work will take 3-times more than required for programming)
- Communication (continuous clarifications with the architect and other groups)
- The surgical team (very small team creats the main components)
- Code freeze and system versioning
- Specialized tools (good to share by team)

Brooks Law and open software projects:

Jamie Zawinski: Apache project, Linux Kernel, and other large software projects, the bulk of the work is done by a few dedicated members or a core team -- what Brooks calls a "surgical team." "In most open source projects," says Zawinski, "there is a small group who do the majority of the work, and the other contributors are definitely at a secondary level, meaning that they don't behave as bottlenecks." ... "Most of the larger open source projects are also fairly modular, meaning that they are really dozens of different, smaller projects. So when you claim that there are ten zillion people working on the Gnome project, you're lumping together a lot of people who never need to talk to each other, and thus, aren't getting in each others' way."

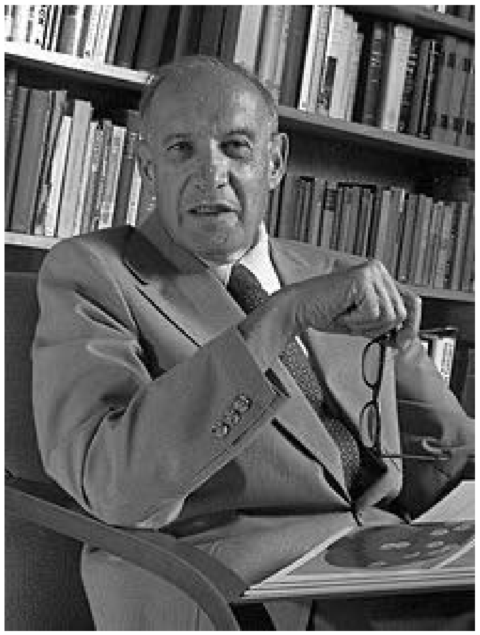
Brian Behlendorf (Apache & Collab.net): "We don't consciously think about it, but I think that the philosophy of keeping things simple and pushing out almost anything extraneous or nonessential to external modules has been followed fairly carefully in Apache. We've also been fairly successful (I think) in 'federalizing' the Apache process to sister projects."

Peter Drucker

Peter Ferdinand Drucker was a management consultant, educator, and author, named "**the founder of modern management**".
 - https://en.wikipedia.org/wiki/Peter_Drucker

Key Ideas:

- **Decentralization and simplification** is better than "command and control".
- The prediction of the **decline of the "blue color" worker**.
- The concept of "**outsourcing**" ("front room" and "back room").
- The importance of the nonprofit sector (third sector).
- **Respect for the worker**. People are an organization's most valuable resource. A manager's job is both to prepare people to perform and to give them freedom to do so.
- The need for **"planned abandonment" of old methods**.
- A belief that taking action without thinking is the cause of every failure.
- **The importance of volunteering**
- The need to manage business by **balancing a variety of needs and goals**, rather than subordinating an institution to a single value.
- A company's primary responsibility is to serve its customers. Profit is not the primary goal, but rather an essential condition.
- **"Do what you do best and outsource the rest"** is a business tagline first "coined and developed" in the 1990s by Drucker.

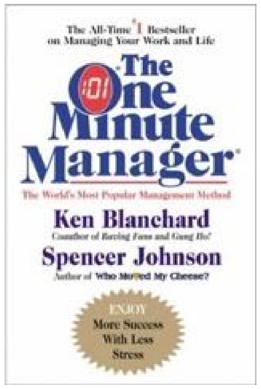


Peter Ferdinand Drucker
(1909-2005)

Peter Drucker on marketing:

"Because the purpose of business is to create a customer, the business enterprise has two - and only two - basic functions: **marketing and innovation**. **Marketing and innovation produce results; all the rest are costs**. Marketing is the distinguishing, unique function of the business."

One Minute Manager


The All-Time #1 Bestseller on Managing Your Work and Life
The One Minute Manager
The World's Most Popular Management Method
Ken Blanchard
Spencer Johnson
Author of Who Moved My Cheese?
ENJOY More Success With Less Stress

Book "The One Minute Manager"
by Spencer Johnson & Kenneth H. Blanchard

Classical bestseller.
How to achieve incremental improvement using 3 tools:
1. One minute goals (on single sheet of paper)
2. One minute praise
3. One minute reprimand

1. Set one minute goals

- plan the goals together, have people write them on a single page, with due dates.
- ask them to review their most important goals each day
- ask them to periodically check if their behavior matches their goals. If not - rethink their behavior or goals.

2. Give one minute praisings

- Praise people as soon as possible.
- Be specific about what they did right
- Tell people how good you feel about what they did right, and how it helps
- Pause for a moment to allow people time to feel good about what they've done
- Encourage them to do more of the same
- Make it clear you have confidence in them and support their success

3. Give one minute reprimands (re-directs) to address mistakes

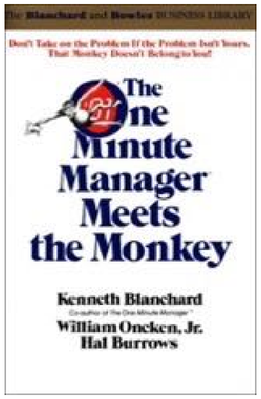
First half-minute – reprimand

- Do it as soon as possible
- Confirm the facts first
- Review the mistake together, be specific
- Express how you feel about the mistake and its impact on results
- Pause... Be quiet for a moment to allow people time to feel concerned about what they've done.

Second half-minute – re-connect

- Let them know that they're better than their mistake, and that you think well of them as a person
- Remind them that you trust in them, and support their success
- Realize that when the reprimand is over, it's over

One Minute Manager Meets the Monkey


The Blanchard and Bowles BUSINESS LIBRARY

Don't Take on the Problem If the Problem Isn't Yours. That Monkey Doesn't Belong to You!

The One Minute Manager Meets the Monkey

Kenneth Blanchard
Co-author of The One Minute Manager
William Oncken, Jr.
Hal Burrows

Book "The One Minute Manager Meets the Monkey"
by William, Jr. Oncken, et al

Humorous and highly effective way to learn how to start delegating business tasks.

Monkey is the next move ...

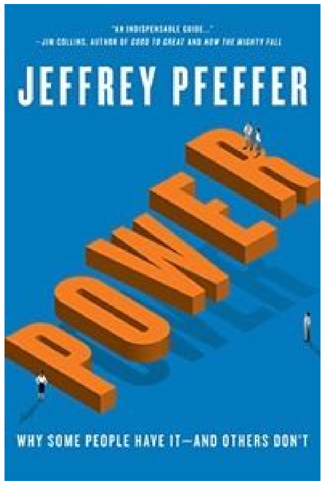
Who owns the monkey?
The book describes many situations when subordinates put monkeys on their boss's back. And teaches tips and tricks on how managers can avoid these leaping monkeys and put them back on the suborinates backs.

Example, a person goes to the boss and says "Boss, we have a problem".
This is a dangerous situation, because a monkey representing this problem may leap from subordinate back to boss's back – if the boss agrees to do something about this problem.

How to avoid those leaping monkeys?

Here is an example:
"Stop right there!
We can never have a problem.
It is either you have a problem – and I will advise you what to do.
Or I have a problem – then I will tell you what you will need to do for me.
In any case you will be the one doing all the work.
So, what's the problem?"

Power & Growth Mindset



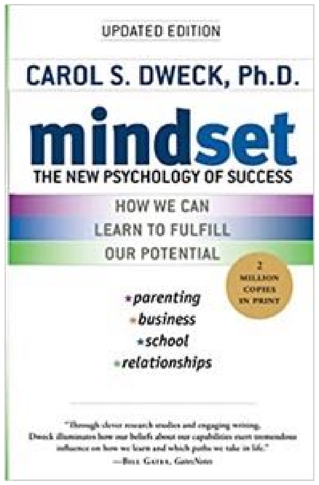
Power: Why Some People Have It - and Others Don't
by Jeffrey Pfeffer

How to become more powerful.

Jeffrey Pfeffer, professor of organizational behavior at Stanford University, states that **intelligence, performance, and likeability alone are not the key to moving up in an organization**.

He asserts that vital factors are:

- self promotion
- building relationships
- cultivating a reputation for control and authority
- perfecting a powerful demeanor


UPDATED EDITION
CAROL S. DWECK, Ph.D.
mindset
THE NEW PSYCHOLOGY OF SUCCESS
HOW WE CAN LEARN TO FULFILL OUR POTENTIAL
2 MILLION COPIES IN PRINT
parenting
business
school
relationships

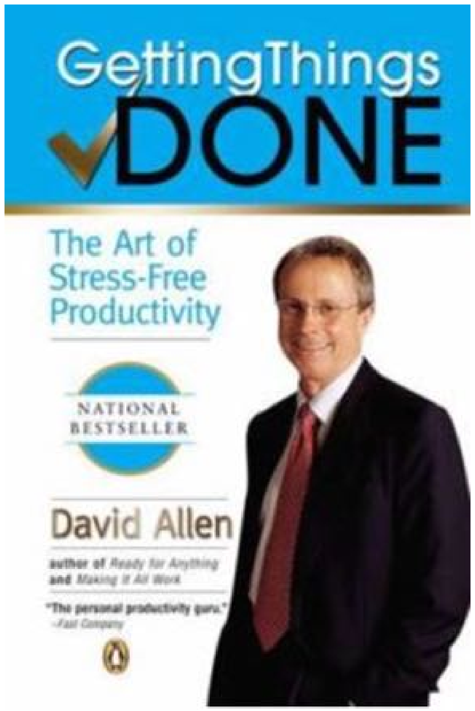
Mindset: The New Psychology of Success
by Carol S. Dweck

Carol Dweck has coined the term **"Growth Mindset"**

If you reward correct results, people will learn to avoid risk and failure.

If you reward efforts, attempts, invention, and do not punish for failure - people will become much more effective and in the end achieve better results.

"Getting Things Done" by David Allen


Getting Things DONE
The Art of Stress-Free Productivity
NATIONAL BESTSELLER
David Allen
author of Ready for Anything and Making It All Work
"The personal productivity guru."
—Fast Company

"You mind is for having ideas, not for holding them" – David Allen

GTD is a manual for stress-free productivity.
It teaches you to set up a system of lists, reminders and weekly reviews, in order **to free your mind** from having to remember tasks and to-dos and instead let it work at full focus on the task at hand.

The book was published in 2001 (20 years ago).
Nowadays 20-years later you can easily spot the similarity between the system proposed in the book with modern practices of software development (Agile/Scrum methods).

Here is a good short summary of the book:
- https://fourminutebooks.com/getting-things-done-summary/ -

David Allen proposes a system based on two principles:
- define the next executable action for every theme
- log your actions in a system which can be trusted.

Main principles:
- Use a "collection bucket" to store things outside your mind.
- Create a "next actions" list for all your projects.
- Do a weekly review of everything, or else!

Specifically the system consists of 5 steps:
- collecting
- processing
- organizing
- reviewing
- executing

Quotes

From book "HBR's 10 Must Reads on Managing People"

Daniel Goleman: There are six leadership styles - coercive, authoritative, affiliative, democratic, pacesetting and coaching. The most effective leaders are able to change between these styles when appropriate.

Frederick Herzberg: Punishments and rewards are ineffective tools for motivating people. Instead, try enriching their jobs by removing controls, giving employees more information, and giving access to greater challenges.

Manzoni and Barsoux: Employees who are viewed as weak performers often live down to expectations because the supervisor's attempts at performance management result in worse rather than better performance.

Carol Walker: New managers often perform poorly because they have not learnt the skills of delegating, getting support from above, projecting confidence, focusing on the big picture, and giving constructive feedback.

Marcus Buckingham: Great managers do not try to change their employees. Instead, they tweak roles to capitalize on individual strengths, create personalised incentives, and tailor coaching to unique learning styles.

Kim and Mauborgne: Harmony in the workplace required fair process, including inviting input from employees affected by a decision, explaining the thinking behind decisions, and providing clear expectations.

Chris Argyris: An organization's smartest and most successful people are often poor learners because they have not had the opportunity for introspection that comes with failure.

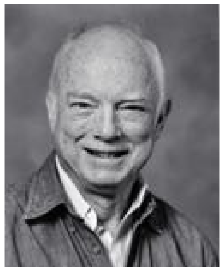
Banaji, Bazerman and Chugh: Everyone has unconscious biases which affect decisions. To counteract these biases, gather better data, get rid of stereotypical cues, and broaden your mind-set.

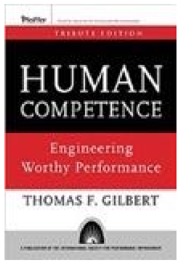
Katzenbach and Smith: A good team has a meaningful common purpose, specific performance goals, a mix of complementary skills, a strong commitment to how the work gets done, and mutual accountability.

Gabarro and Kotter: To have a good relationship with your boss, focus on compatible work styles, mutual expectations, information flow, dependability and honesty, and good use of time and resources.

Thomas F. Gilbert

- psychologist, founder of Performance Engineering (a.k.a. **Human Performance Technology (HPT)**)
- book "Human Competence: Engineering Worthy Performance".
- realized that formal learning programs often only change knowledge, not behavior.
- worked with the behavioral psychologist **B. F. Skinner at Harvard University** and with **Ogden R. Lindsley in Lindsley's laboratory** at Metropolitan State Hospital in Waltham, Massachusetts
- specialized in statistics, testing and measurement.
- https://en.wikipedia.org/wiki/Thomas_Gilbert_(engineer)


Thomas F. Gilbert (1927–1995)

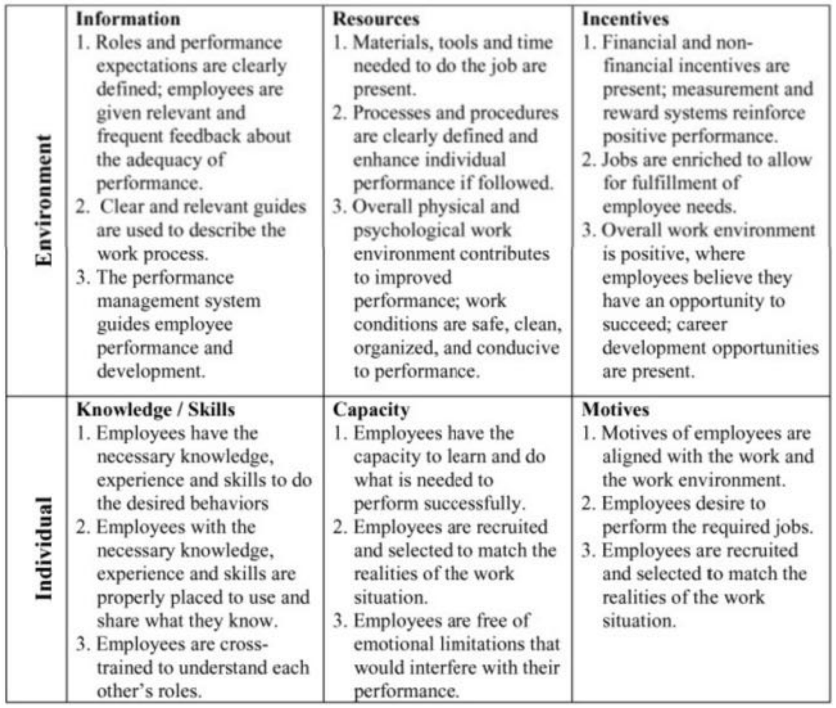


Gilbert classified important and manageable factors affecting performance in a 2 × 3 matrix that he called his **Behavior Engineering Model** (**BEM**). **BEM** identifies six variables necessary to improve human performance:

```
+-------------+-------------+------------+-------------+
|             | Stimulus    | Response   | Consequence |
+-------------+-------------+------------+-------------+
| Environment | information | resources  | incentives  |
+-------------+-------------+------------+-------------+
| Individual  | knowledge   | capacity   | motives     |
+-------------+-------------+------------+-------------+
```

Simple Performance Model:
- Quality (Accuracy, Class, Novel)
- Quantity (Rate, Timeliness, Volume)
- Cost (Labor, Material, Management)

BEM Example:



Environment	**Information** 1. Roles and performance expectations are clearly defined; employees are given relevant and frequent feedback about the adequacy of performance. 2. Clear and relevant guides are used to describe the work process. 3. The performance management system guides employee performance and development.	**Resources** 1. Materials, tools and time needed to do the job are present. 2. Processes and procedures are clearly defined and enhance individual performance if followed. 3. Overall physical and psychological work environment contributes to improved performance; work conditions are safe, clean, organized, and conducive to performance.	**Incentives** 1. Financial and non-financial incentives are present; measurement and reward systems reinforce positive performance. 2. Jobs are enriched to allow for fulfillment of employee needs. 3. Overall work environment is positive, where employees believe they have an opportunity to succeed; career development opportunities are present.
Individual	**Knowledge / Skills** 1. Employees have the necessary knowledge, experience and skills to do the desired behaviors 2. Employees with the necessary knowledge, experience and skills are properly placed to use and share what they know. 3. Employees are cross-trained to understand each other's roles.	**Capacity** 1. Employees have the capacity to learn and do what is needed to perform successfully. 2. Employees are recruited and selected to match the realities of the work situation. 3. Employees are free of emotional limitations that would interfere with their performance.	**Motives** 1. Motives of employees are aligned with the work and the work environment. 2. Employees desire to perform the required jobs. 3. Employees are recruited and selected to match the realities of the work situation.

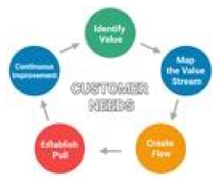
Human Performance Technology (HPT)

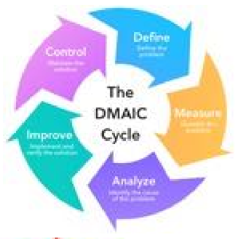
Human Performance Technology (HPT),
a.k.a. Human Performance Improvement (HPI)
or Human Performance Assessment (HPA)

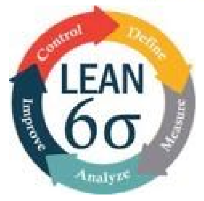
Process improvement methodologies such as:

- Lean management
  - https://en.wikipedia.org/wiki/Lean_manufacturing -
- Six Sigma (tools to improve business processes)
  - https://en.wikipedia.org/wiki/Six_Sigma -
- Lean Six Sigma
  - https://en.wikipedia.org/wiki/Lean_Six_Sigma -
- organization development
- motivation
- instructional technology
- human factors
- learning
- performance support systems
- knowledge management
- training

- https://en.wikipedia.org/wiki/Human_performance_technology







International Society for Performance Improvement (ISPI)
- https://ispi.org/default.aspx -

DMAIC Cycle
Define-Measure-Analyze-Improve-Control

The origins of HPT can be primarily traced back to the work of

- **Thomas Gilbert**
- **Geary Rummler**
- Karen Brethower
- Roger Kaufman
- Bob Mager
- Donald Tosti
- Lloyd Homme
- Joe Harless

Project Management

What we have learned:

- Crystal clear understanding of the requirements. Involve all business stakeholders into the planning process. Make sure they agree on the plan.
- Crystal clear understanding of priorities: what is important – and what is "nice to have".
- Concentrate only on important features, sacrifice the rest.
- Simplifty, simplify, simplify.
- Split big job into small pieces, small steps.
- Grow project as a child from few features to full-featured product.
- Military hierarchical management is not effective. Respect workers, leverage their brains, give them freedom to be effective, to communicate and collaborate.
- Give people taks that match their abilities.
- Create safe and engaging atmosphere.