Pragmatismo en la Planificación de Proyectos

Juan Cabral - jbc.develop@gmail.com

Mayo 19, 2017

Dos consideraciones:

1. Toda respuesta profesional empieza con la palabra **“depende”**
2. Tener en cuenta que error, falla y no-conformidades no son lo mismo.
3. Overkill

4. Lean Dilbert (http://dilbert.com/)

Agenda

- Ingeniería de Software.
- Proyectos (Triangulo de Hierro).
- Ciclo de vida de un proyecto
- Que es un objetivo (SMART).
- Estimación de tiempos (PERT).
- Complejidad de tareas.

Ingeniería de Software

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación, y mantenimiento del software.

Software

Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.

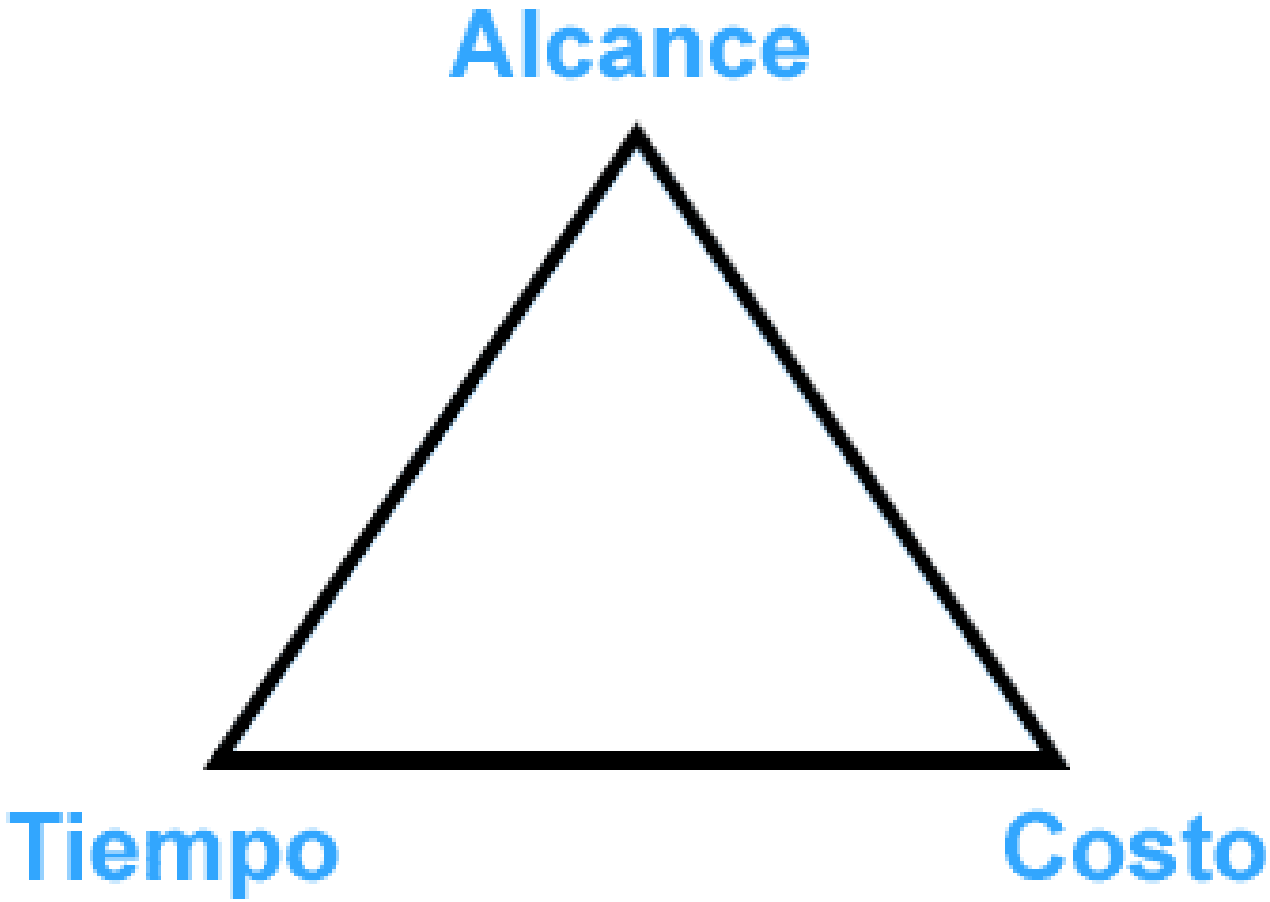
IEEE Standard Glossary
of Software Engineering Terminology

Proyectos

It's a temporary endeavor undertaken to create a unique product, service or result.

PMI

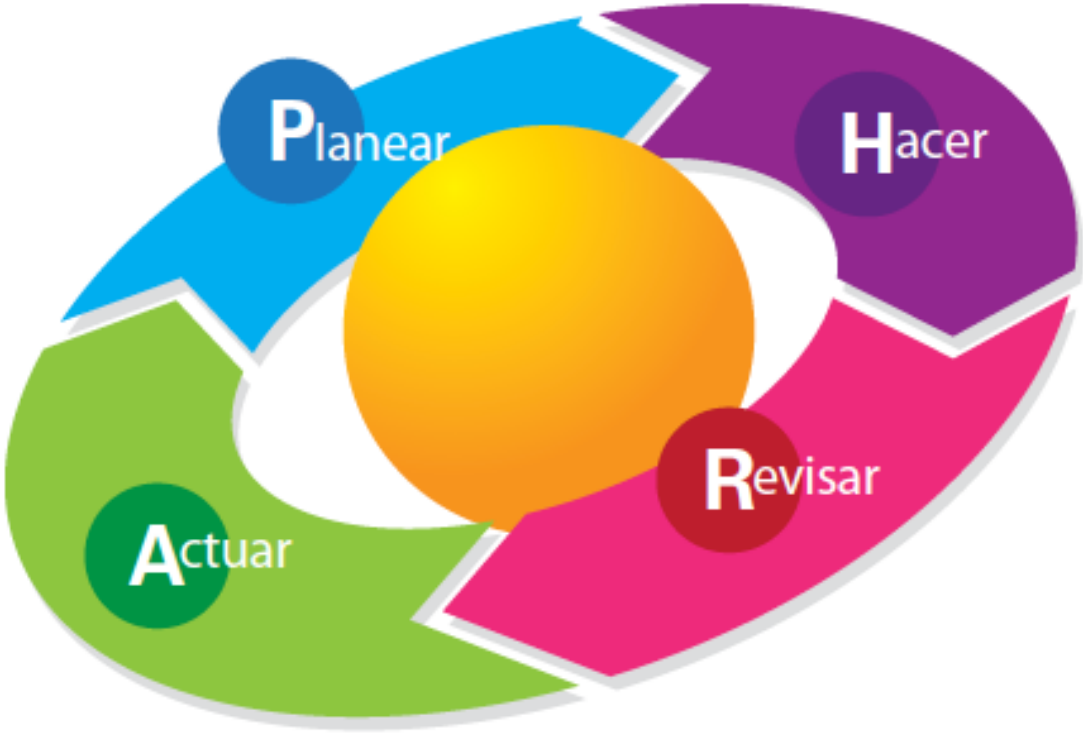
Triangulo de Hierro



Proyectos


Costo
Alcance
Tiempo
Costo
Alcance
Mayor Alcance
con Mayor Costo
Tiempo
Costo
Alcance
Mayor Alcance
con Mayor Tiempo
Tiempo

Ciclo de Deming



- Solo nos fijamos un poco en Planear y Revisar

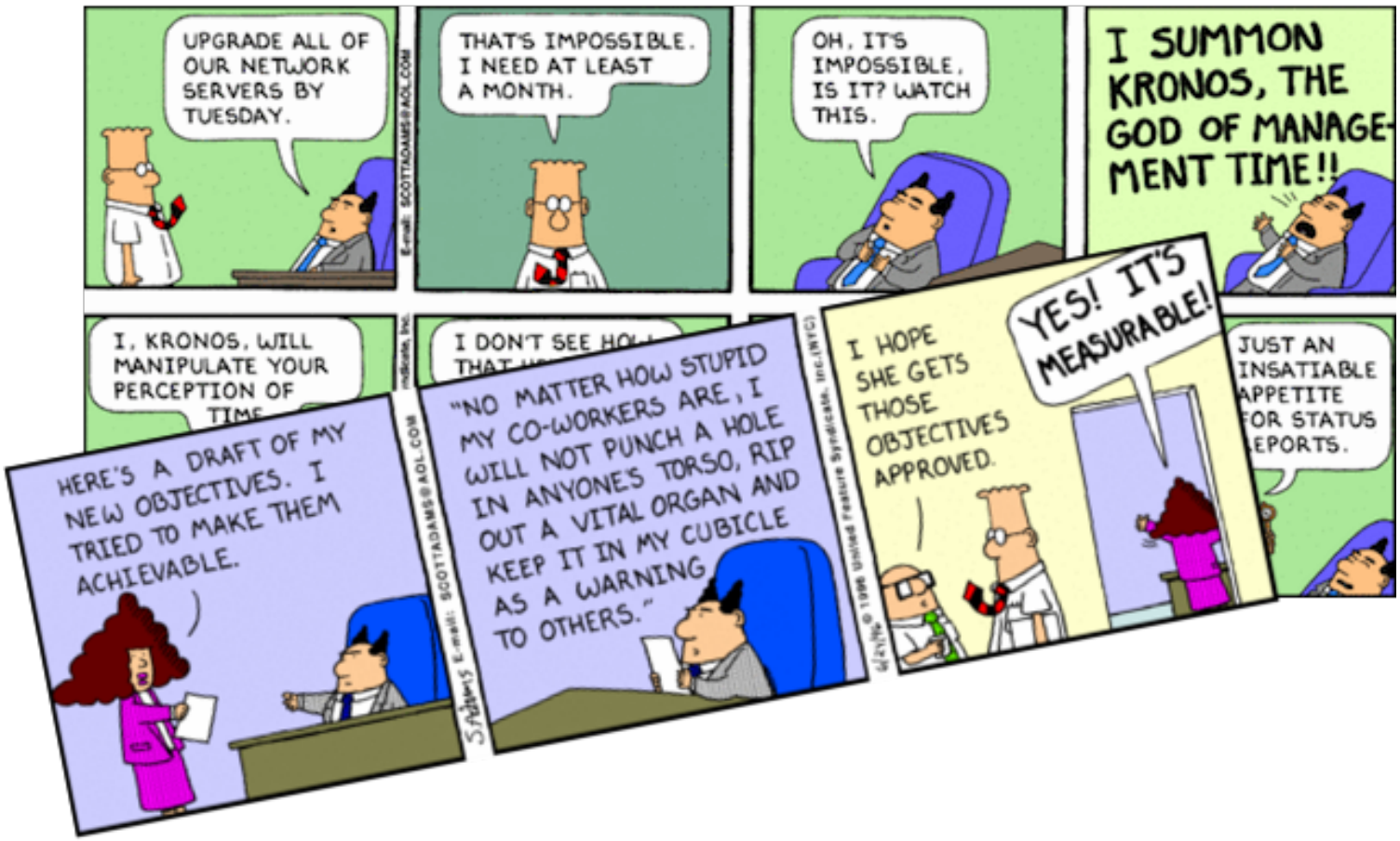
Objetivos

Según la teoría general de sistemas

El elemento programático que identifica la finalidad hacia la cual deben dirigirse los recursos y esfuerzos para dar cumplimiento a los propósitos.

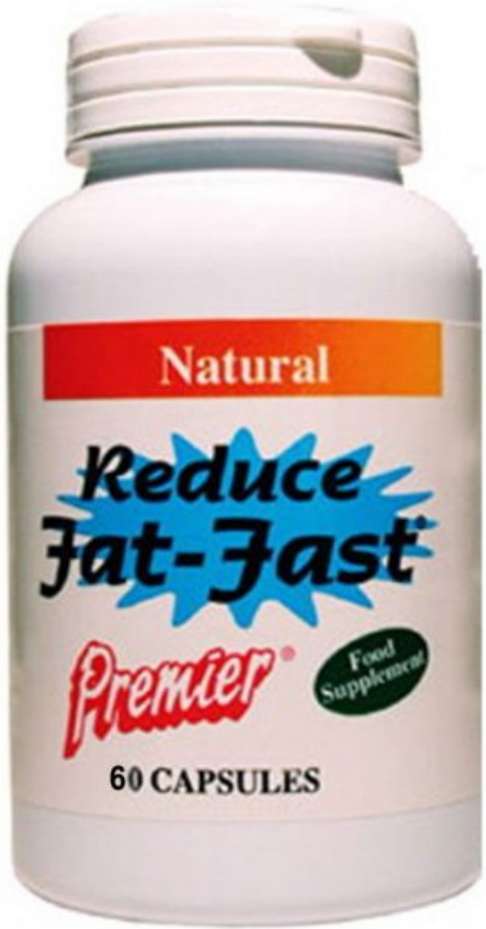
Objetivos - SMART

- **S**pecific – target a specific area for improvement.
- **M**easurable – quantify or suggest an indicator of progress.
- **A**chievable – can be realistically achieved, given.
- **R**esponsible – specify who will do it.
- **T**ime-related – specify when the result(s) can be achieved.



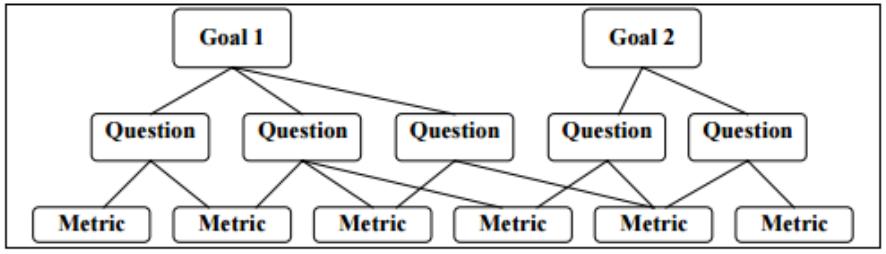
Objetivo - SMART

- Voy adelgazar. (No Smart)
- Voy a adelgazar 100 Kg, en un mes usando reduce-fat-fast. (**SMART**)



Métricas

- No se puede controlar lo que no se puede medir.
- El enfoque que se usa es GQM el cual deriva Objetivos (**G**) a preguntas (**Q**) las cuales tratan de responderse con métricas (**M**).



Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question		What is the current change request processing speed?
Metrics		Average cycle time Standard deviation % cases outside of the upper limit
Question		Is the performance of the process improving?
Metrics		$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction

Estimación de Tiempos y tareas criticas (PERT)

- Preguntas razonables cuando uno plantea una idea:
  - Como lo vas a hacer?
  - Quien lo va a hacer?
  - Cuanto vas a tardar? (**y aca morimos todos**)
- PERT (del inglés, **P**roject **E**valuation and **R**eview **T**echniques), son un modelo para la administración y gestión de proyectos inventado en 1957 por la Oficina de Proyectos Especiales de la Marina de Guerra del Departamento de Defensa de EE.UU. como parte del proyecto Polaris de misil balístico móvil lanzado desde submarino. Este proyecto fue una respuesta directa a la crisis del Sputnik.

PERT (cont.)

En planificación y programación de proyectos se estima que la duración esperada de una actividad es una variable aleatoria de distribución de probabilidad Beta Unimodal:

$$t_e = \frac{(t_o + 4t_m + t_p)}{6}$$

- t_e: Expected time
- t_o: the minimum possible time required to accomplish a task, assuming everything proceeds better than is normally expected
- t_p: the maximum possible time required to accomplish a task, assuming everything goes wrong but excluding major catastrophes.
- t_m: the best estimate of the time required to accomplish a task, assuming everything proceeds as normal.

PERT (cont.)

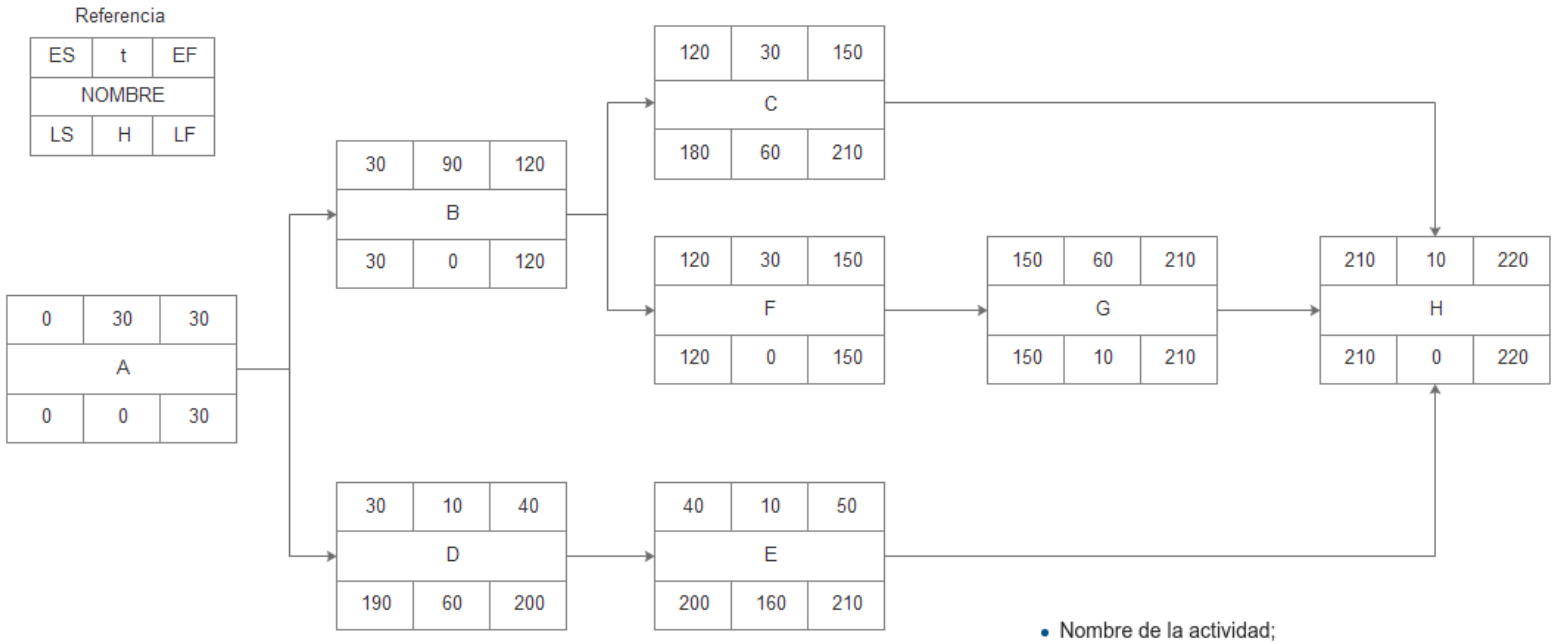
La desviación estandar de la tarea esta dada por:

$$\sigma = \frac{t_p - t_o}{6}$$

Ejemplos

- `https://github.com/leliel12/otree_korbinian/blob/master/_doc/estimation.ipynb`
- `https://github.com/leliel12/otree_wissink/blob/master/_doc/estimation.ipynb`

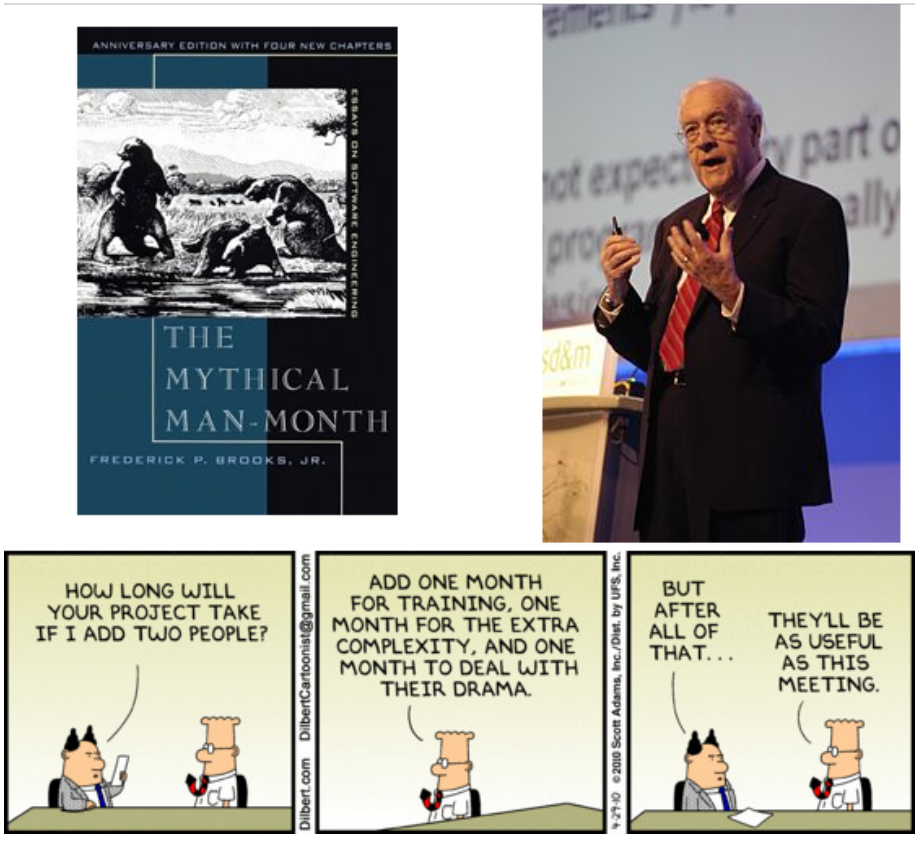
Redes PERT



- Nombre de la actividad;
- Duración esperada de la actividad (t);
- Tiempo de inicio más temprano (ES = Earliest Start);
- Tiempo de término más temprano (EF = Earliest Finish);
- Tiempo de inicio más tardío (LS = Latest Start);
- Tiempo de término más tardío (LF = Latest Finish);
- Holgura de la Actividad (H);

Frederick Brooks (cont)

Turing Award 199: For landmark contributions to computer architecture, operating systems, and software engineering.



Frederick Brooks (cont)

The Mythical Man-Month

Complex programming projects cannot be perfectly partitioned into discrete tasks that can be worked on without communication between the workers and without establishing a set of complex interrelationships between tasks and the workers performing them. Therefore, assigning more programmers to a project running behind schedule will make it even later.

Group intercommunication formula

$$G_i = \frac{n(n-1)}{2}$$

Example: 50 personas

$$1225 = \frac{50(50-1)}{2}$$

Frederick Brooks (cont)

No-Silver Bullets

There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity.

Frederick Brooks (cont)

The tendency towards irreducible number of errors

In a suitably complex system there is a certain irreducible number of errors. Any attempt to fix observed errors tends to result in the introduction of other errors.

```
99 little bugs in the code.
99 little bugs.
Take one down, patch it around.
127 little bugs in the code..
```

Frederick Brooks (cont)

Progress tracking

- **Question:** How does a large software project get to be one year late?
- **Answer:** One day at a time!"

Frederick Brooks (cont)

Conceptual integrity

To ensure a user-friendly system, a system may deliberately provide fewer features than it is capable of. The point is that, if a system is too complicated to use, then many of its features will go unused because no one has the time to learn how to use them.

Frederick Brooks (cont.)

The pilot system

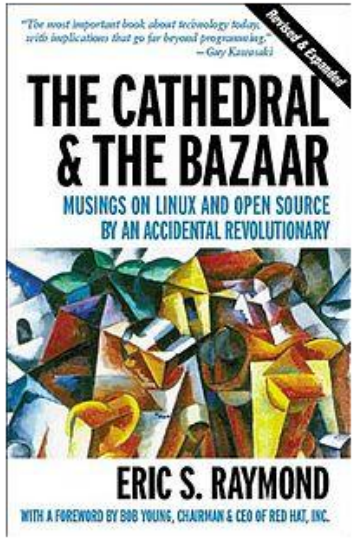
When designing a new kind of system, a team will design a throw-away system (whether it intends to or not).

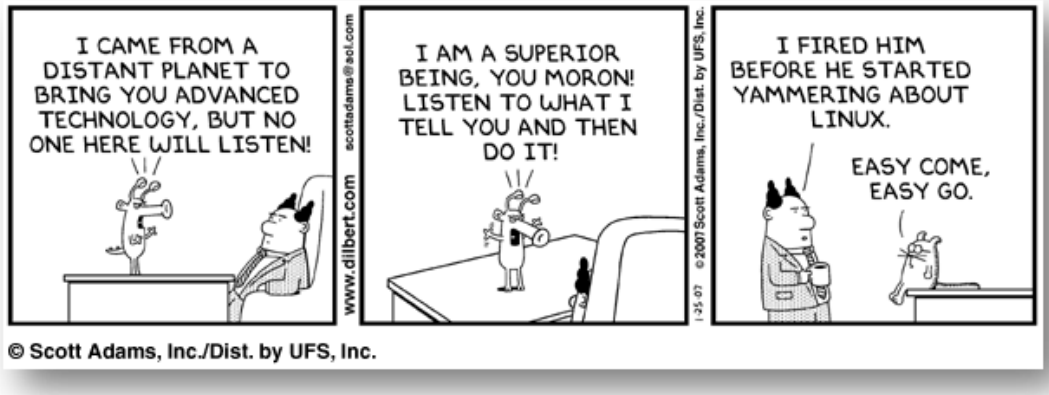
Frederick Brooks (cont.)

Code freeze

Software is invisible. Therefore, many things only become apparent once a certain amount of work has been done on a new system, allowing a user to experience it. This experience will yield insights, which will change a user's needs or the perception of the user's needs.

Eric Raymond

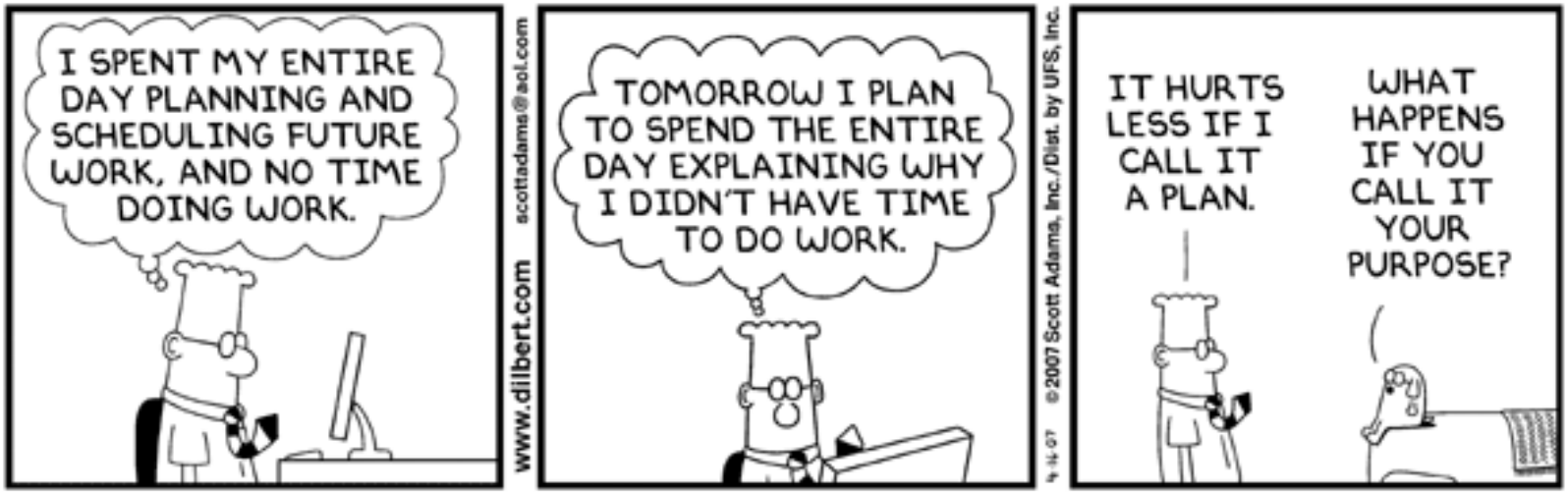




Eric Raymond (cont.)

- Given enough eyeballs, all bugs are shallow (linus' law)
- Good programmers know what to write. Great ones know what to rewrite (and reuse)
- Release early. Release often. And listen to your customers.
- Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
- Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away

Conclusiones & Preguntas



Slides

`https://github.com/leliel12/talks/blob/master/iate2017/proyectos_sem/slides.pdf`