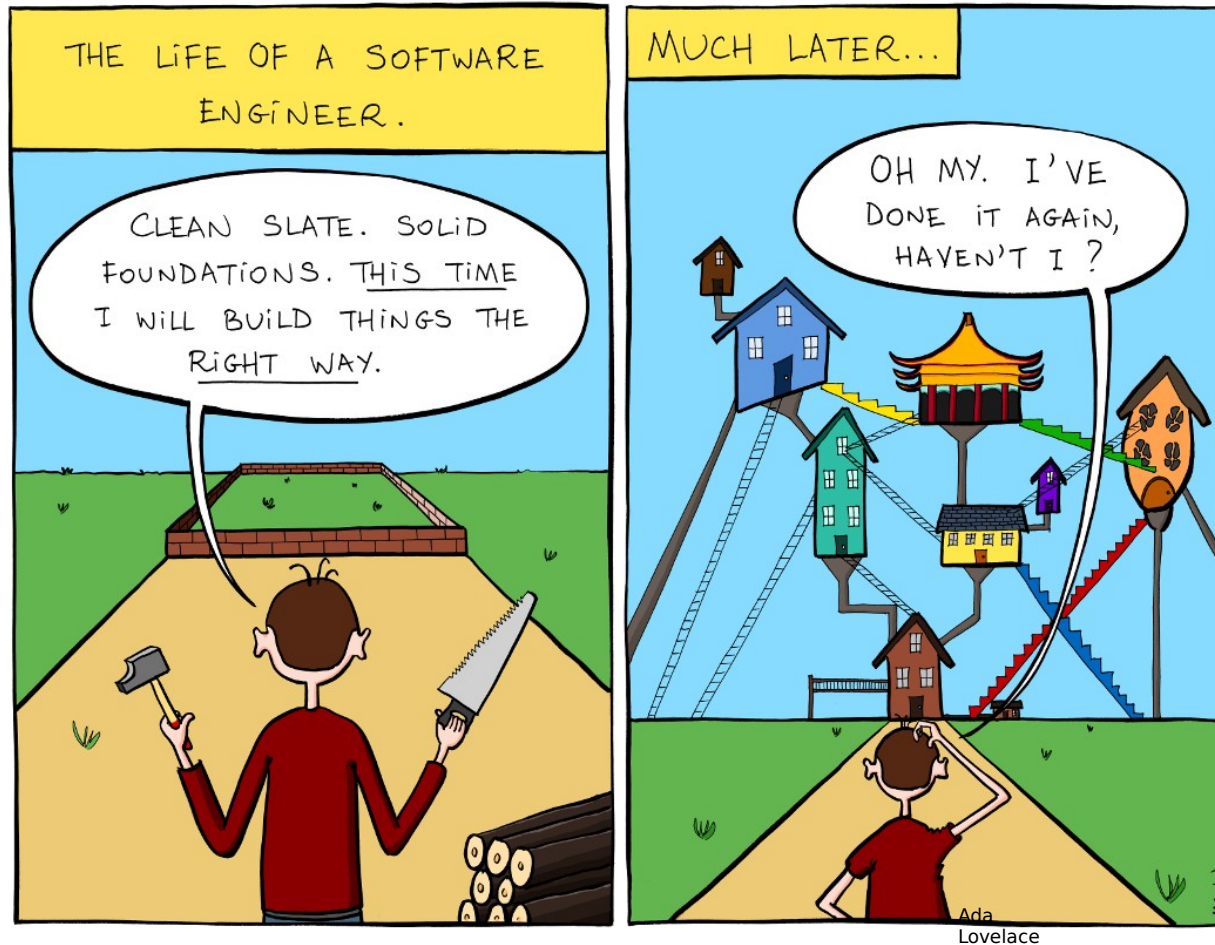


Enginyeria del Software



CFGs DAM - M05
Mònica Ramírez Arceda

Què és l'anàlisi?

Esbrinar **què** cal fer

Què és el disseny?

Fer-ho **com** cal

Què és la implementació?

Fer-ho

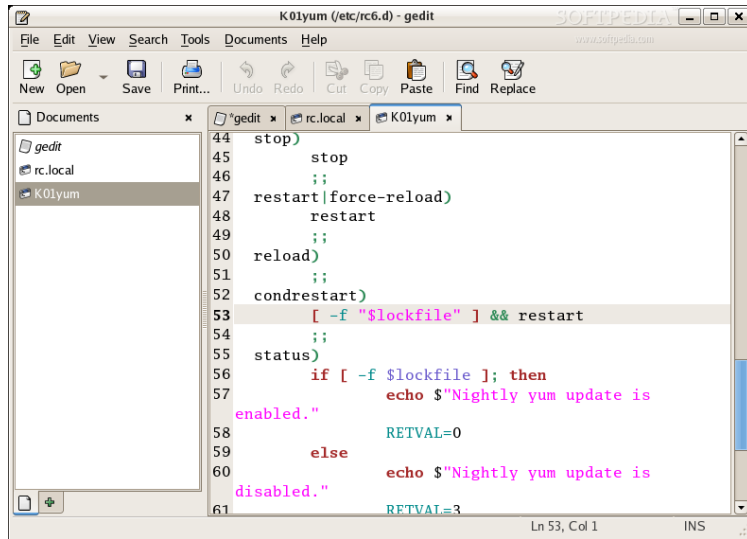
Conceptes Bàsics



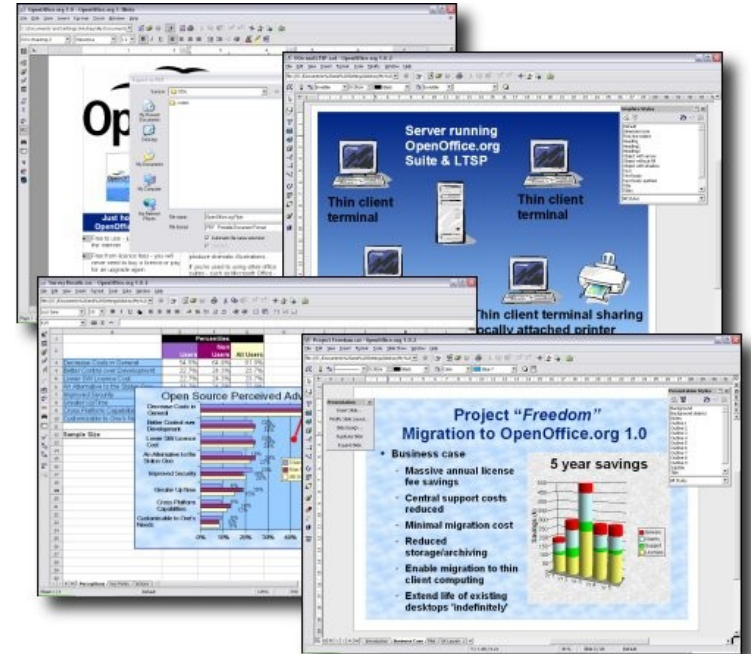
Funcionalitat	Baixa	Alta	Molt alta
Plànols	Un esquema	Arquitectura	Arquitectura/Enginyeria
Materials	Baixa	Sofisticats	Sofisticats
Eines	Senzilles	Complexes	Molt complexes
Personal	Una persona	Bastants	Molts
Temps A/D/I	Unes hores	Uns mesos	Uns anys
Durabilitat	Uns anys	Un segle	Segles
Variacions per modificació	Poques	Moltes	Moltes
Variacions per ampliació	Poques	Poques	Moltes
Robust	Poc	Bastant	Molt
Cost	Baix	Mitjà-alt	Molt alt

Conceptes Bàsics

El mateix passa amb el software!



```
K01yum (/etc/rc6.d) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
Documents
  gedit
  rc.local
  K01yum
44 stop
45     stop
46     ;;
47 restart|force-reload)
48     restart
49     ;;
50 reload)
51     ;;
52 condrestart)
53     [ -f "$lockfile" ] && restart
54     ;;
55 status)
56     if [ -f $lockfile ]; then
57         echo $"Nightly yum update is
58         enabled."
59         RETVAL=0
60     else
61         echo $"Nightly yum update is
62         disabled."
63         RETVAL=3
```



Cancel·lació de projectes

Development of large applications in excess of 5000 function points (~500,000 LOC) is one of the most risky business undertakings in the modern world (Capers Jones)

Risks of cancellation or major delays rise rapidly as overall application size increases (Capers Jones):

- 65% of large systems (over 1,000,000 LOC) are cancelled before completion
- 50% for systems exceeding half million LOC
- 25 % for those over 100,000 LOC

Failure or cancellation rate of large software systems is over 20% (Capers Jones)

Fracàs del software

Restrassos, sobre costos i mala qualitat

Of completed projects, 2/3 experience schedule delays and cost overruns (Capers Jones)

2/3 of completed projects experience low reliability and quality problems in first year of deployment (Capers Jones)

Average cancelled project in U.S. is about a year behind schedule and has consumed 200% of expected budget (Capers Jones).

Excés de documentació?

Civilian software: at least 100 English words produced for every source code statement.
Military: about 400 words (Capers Jones)

El software es una industria que hace un uso intensivo del papel. El volumen total de documentos utilizados en proyectos software es superior al utilizado en la mayoría de los productos fabricados por el hombre... En proyectos software de gran tamaño del ámbito militar, el volumen total de documentos en papel puede superar el millón de páginas.

Lo que resulta sorprendente es que en sistemas muy grandes el volumen de documentación de las especificaciones y documentos técnicos puede ser lo suficientemente extenso como para ser leído en la vida de un solo analista. Suponiendo una velocidad de lectura técnica de alrededor de 200 palabras por minuto, hay algunos sistemas en los que un empleado puede llevarse una vida laboral de 40 años sin hacer nada salvo leer la documentación y encima no terminar de realizar esa tarea. (Capers Jones)

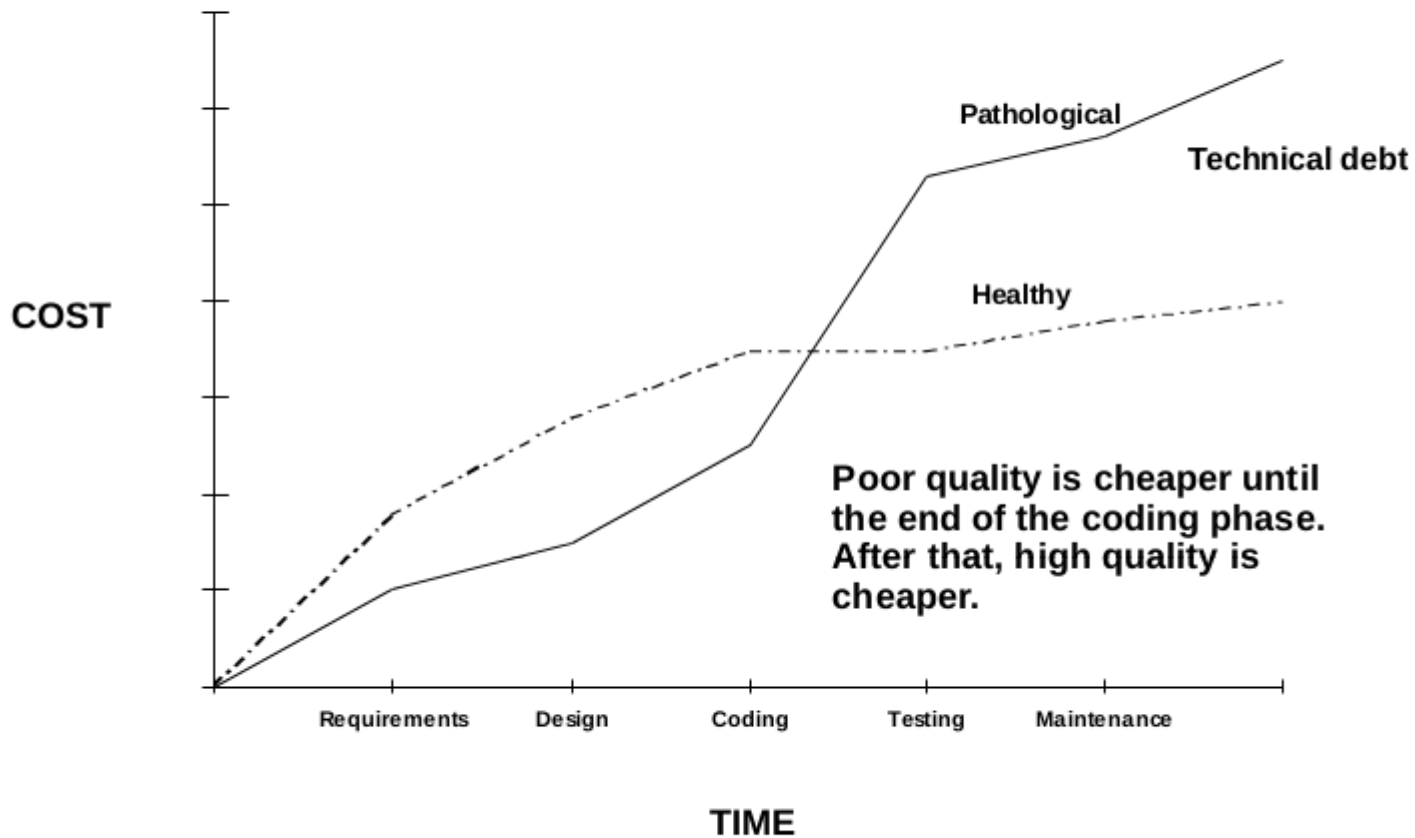
Why do projects fail so often?

The most common factors:

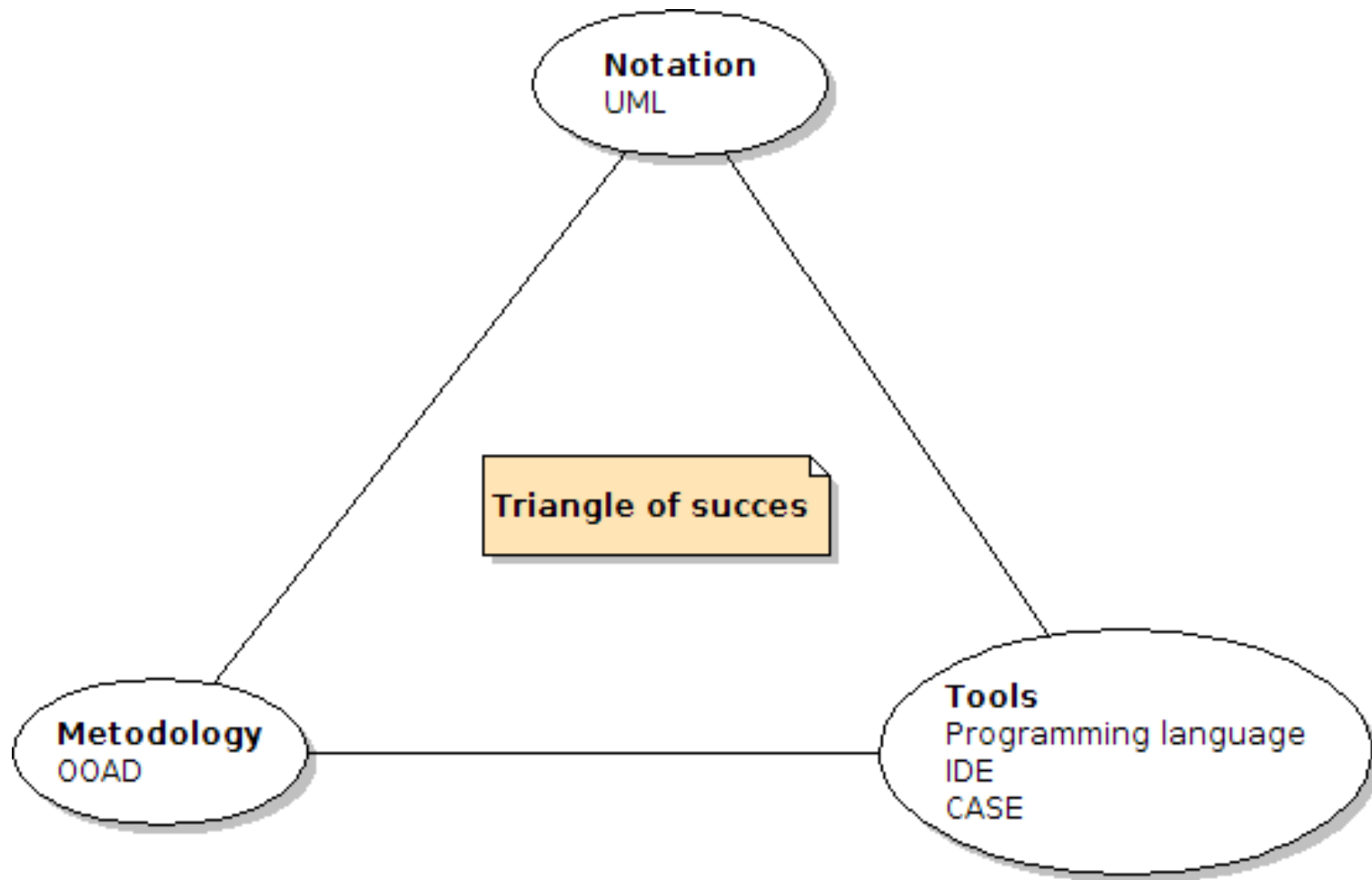
- 1.- Unrealistic or unarticulated project goals
- 2.- Inaccurate estimates of needed resources
- 3.- Badly defined system requirements
- 4.- Poor reporting of the project's status
- 5.- Unmanaged risks
- 6.- Poor communication among customers, developers, and users
- 7.- Use of immature technology
- 8.- Inability to handle the project's complexity
- 9.- Sloppy development practices
- 10.- Poor project management
- 11.- Stakeholder politics
- 12.- Commercial pressures

Fracàs del software

HOW QUALITY AFFECTS SOFTWARE COSTS

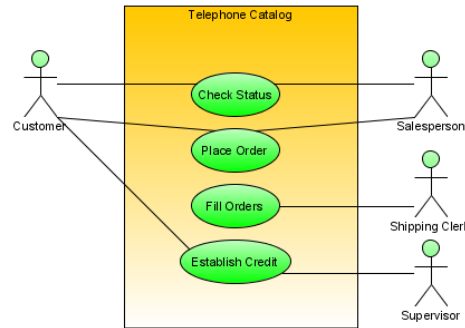


Triangle de l'èxit



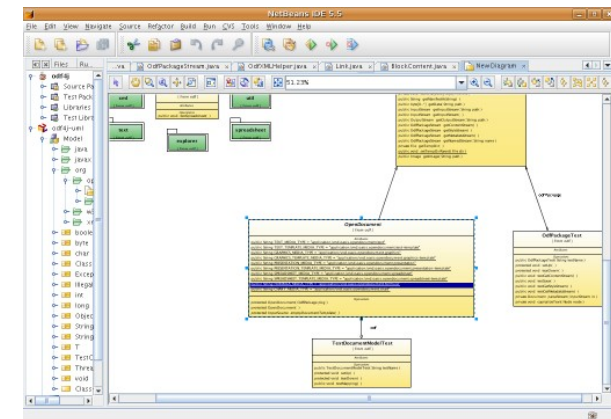
Cicle de vida del software

Analistes + clients



Requeriments

Analistes + dissenyadors



Model

Programadors



```
public class Graph extends Applet {
    GraphPanel panel;

    public void init() {
        setLayout(new BorderLayout());
        panel = new GraphPanel(this);
        add("Center", panel);
        Panel p = new Panel();
        add("South", p);
        p.add(new Button("Scramble"));
        p.add(new Button("Shake"));
        p.add(new Checkbox("Stress"));
        p.add(new Checkbox("Random"));
    }

    String edges = getParameter("edges");
    for (StringTokenizer t = new StringTokenizer(edges, ","); t.hasMoreTokens(); ) {
        String str = t.nextToken();
        int i = str.indexOf('-');
        if (i > 0) {
            int len = 50;
            int j = str.indexOf('/');
            if (j > 0) {
                len = Integer.valueOf(str.substring(j+1)).intValue();
                str = str.substring(0, j);
            }
            panel.addEdge(str.substring(0,i), str.substring(i+1, len));
        }
    }

    Dimension d = size();
    String center = getParameter("center");
    if (center != null) {
        Node n = panel.nodes(panel.findNode(center));
        n.x = d.width / 2;
        n.y = d.height / 2;
        n.fixed = true;
    }
}
```

Codi

Cicle de vida del software

Anàlisi

Independent de l'arquitectura

Disseny

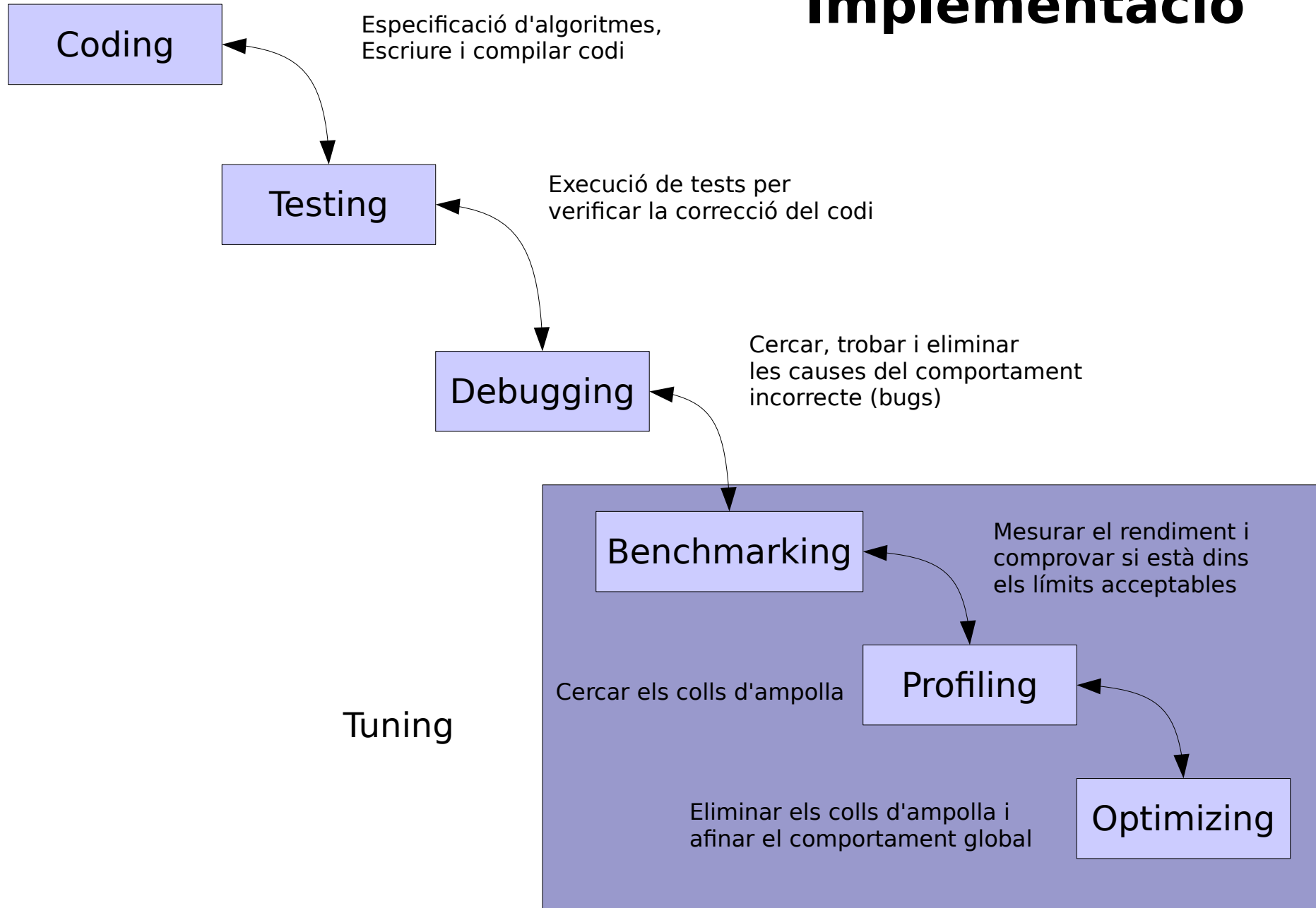
Dependent de l'arquitectura

Implementació

L'arquitectura ve determinada pels **recursos** de hardware i software disponibles i els **requeriments** de l'aplicació

Cicle de vida del software

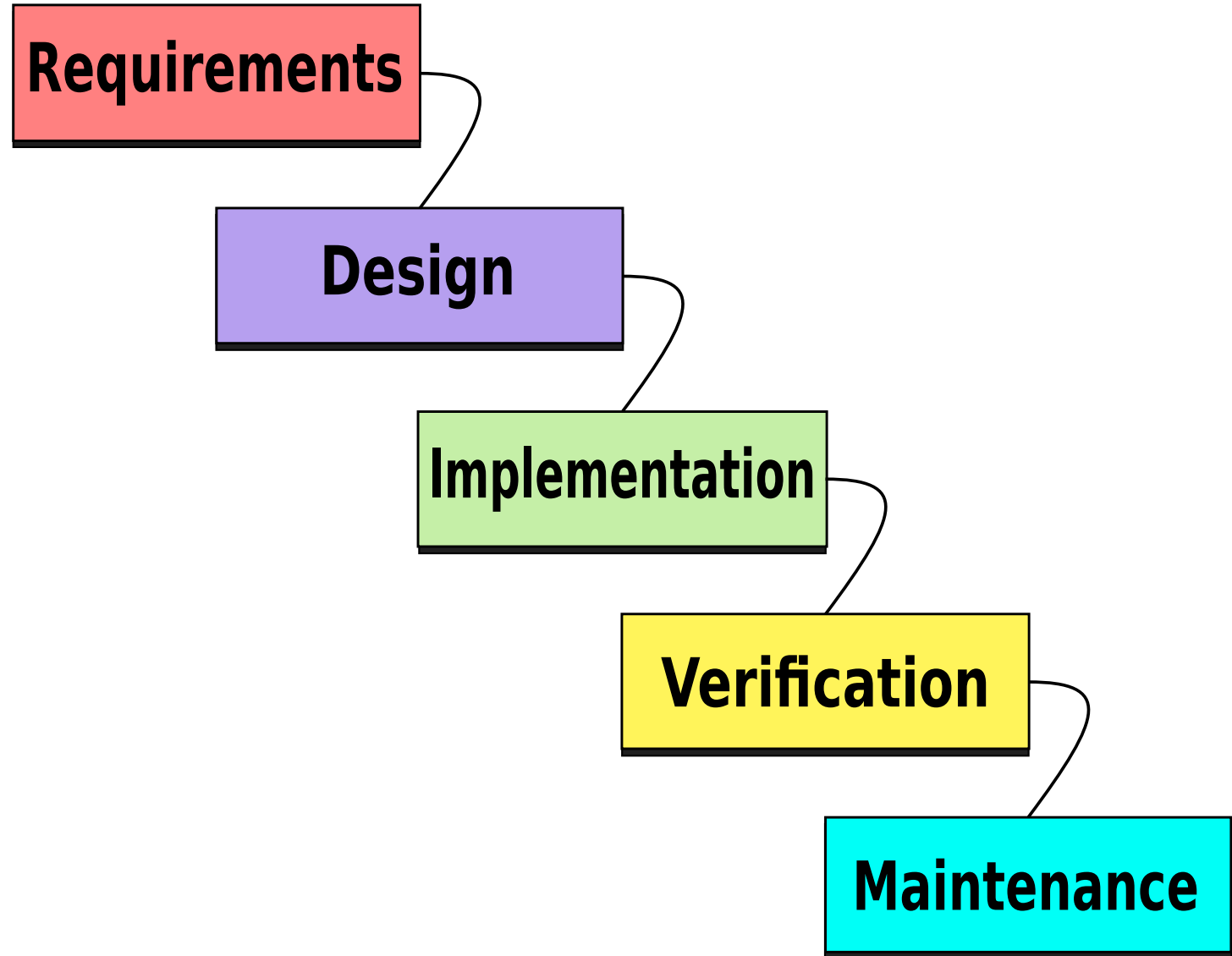
Implementació



Cicle de vida del software

- El **procés** de programari es defineix com un marc de treball de les tasques que es requereixen per a construir programari de qualitat.
- Els **models de procés** es poden definir com les diferents estratègies de desenvolupament de programari per part d'un desenvolupador, o d'un equip. També es coneix com paradigma de la enginyeria del programari.

Cicle de vida del software



Model en cascada

Cicle de vida del software

- El **model lineal o en cascada** és el paradigma més antic i extensament utilitzat, però les crítiques a ell han posat en dubte la seva eficàcia. Malgrat tot, té un lloc molt important en la Enginyeria de programari i continua sent el més utilitzat, i sempre és millor que un enfocament a l'atzar.

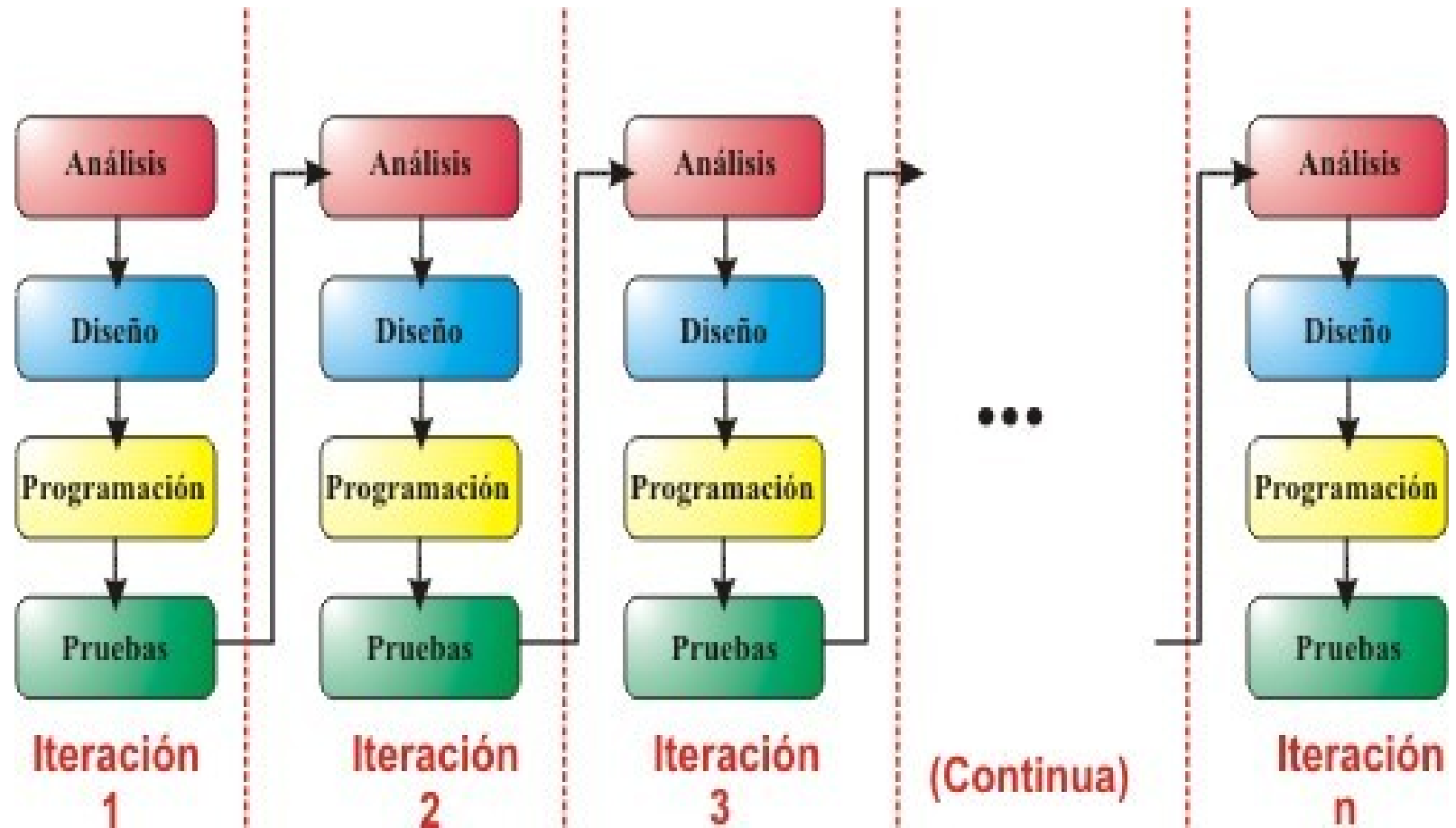
- Desavantatges:

Els canvis introduïts durant el desenvolupament poden confondre a l'equip professional en les etapes primerenques del projecte. Si els canvis es produeixen en etapa madura (codificació o prova) poden ser catastròfics per a un projecte gran.

No és freqüent que el client o usuari final expliciti clara i completament els requisits (etapa d'inici), i el model lineal ho requereix. La incertesa natural en els començaments és després difícil d'acomodar.

El client ha de tenir paciència ja que el programari no estarà disponible fins molt avançat el projecte. Un error detectat pel client (en fase d'operació) pot ser desastrós, implicant reiniciar del projecte, amb alts costos.

Cicle de vida del software



Model iteratiu i incremental

Cicle de vida del software

- El **Model Incremental** és particularment útil quan no es compta amb una dotació de personal suficient. Els primers passos els poden fer un grup reduït de persones i en cada increment es pot afegir personal, si cal. D'altra banda els increments es poden planejar per gestionar riscos tècnics.

- Desavantatges:

El model incremental no és recomanable per a casos de sistemes de temps real, d'alt nivell de seguretat, de processament distribuït, i/o d'alt índex de riscos.

Requereix de molta planificació, tant administrativa com tècnica.

Requereix de metes clares per conèixer l'estat del projecte.

- Avantatges del **model iteratiu i incremental**:

Amb un paradigma incremental es redueix el temps de desenvolupament inicial, ja que s'implementa la funcionalitat parcial.

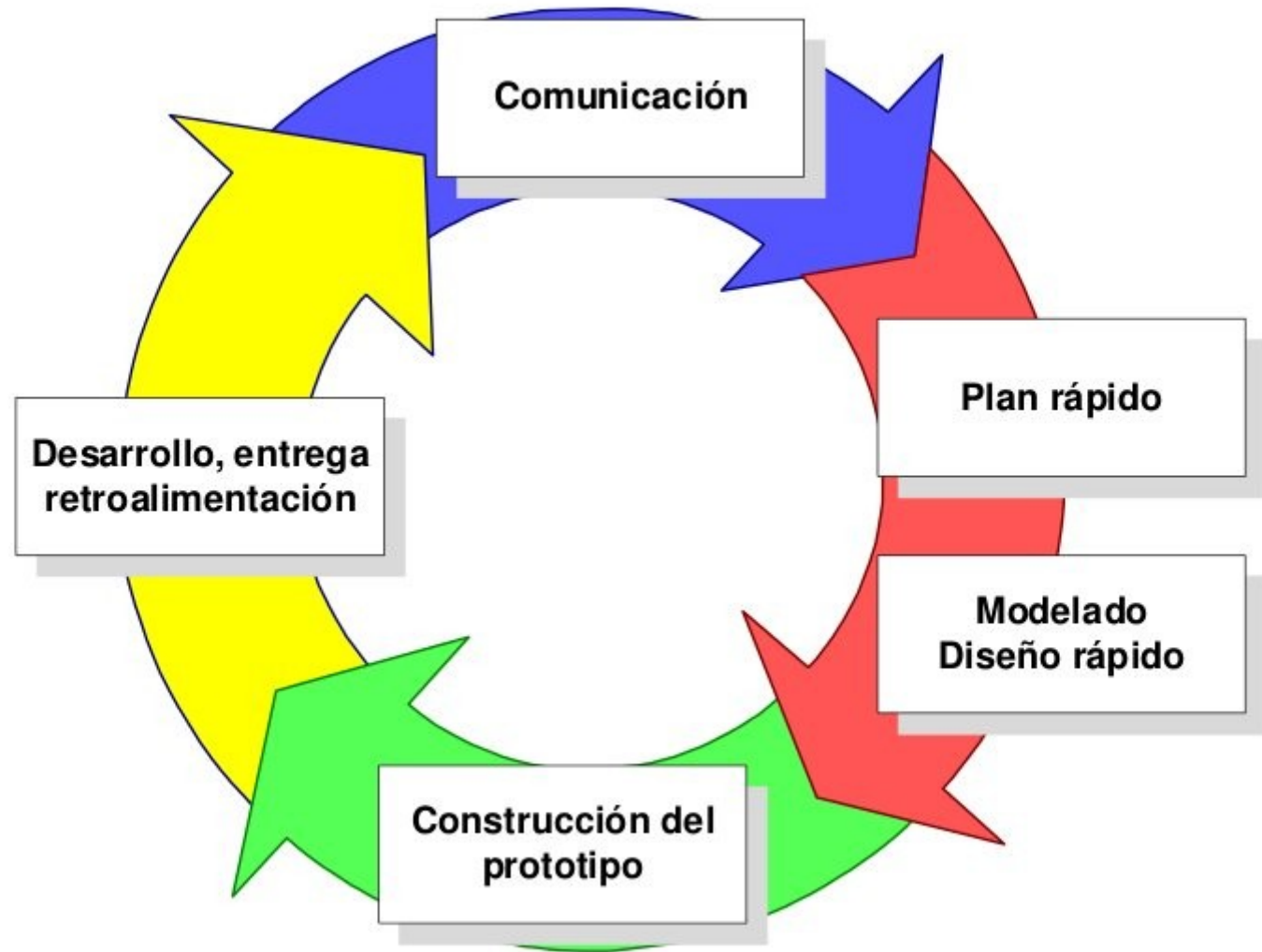
També proveeix un impacte avantatjós enfront del client, que és el lliurament primerenc de parts operatives del programari.

El model proporciona tots els avantatges del model en cascada re-alimentat, reduint els seus desavantatges només a l'àmbit de cada increment.

Permet lliurar al client un producte més ràpid en comparació del model de cascada.

Resulta més senzill acomodar canvis en acotar la mida dels increments.

Cicle de vida del software



Model en prototips

- El **model de construcció de prototips** és pràctic quan el client defineix objectius generals per al programari, però no detalla requeriments d'entrada, processament o eixida.
- Avantatges:

Aquest model és útil quan el client coneix els objectius generals per al programari, però no identifica els requisits detallats d'entrada, processament o sortida.

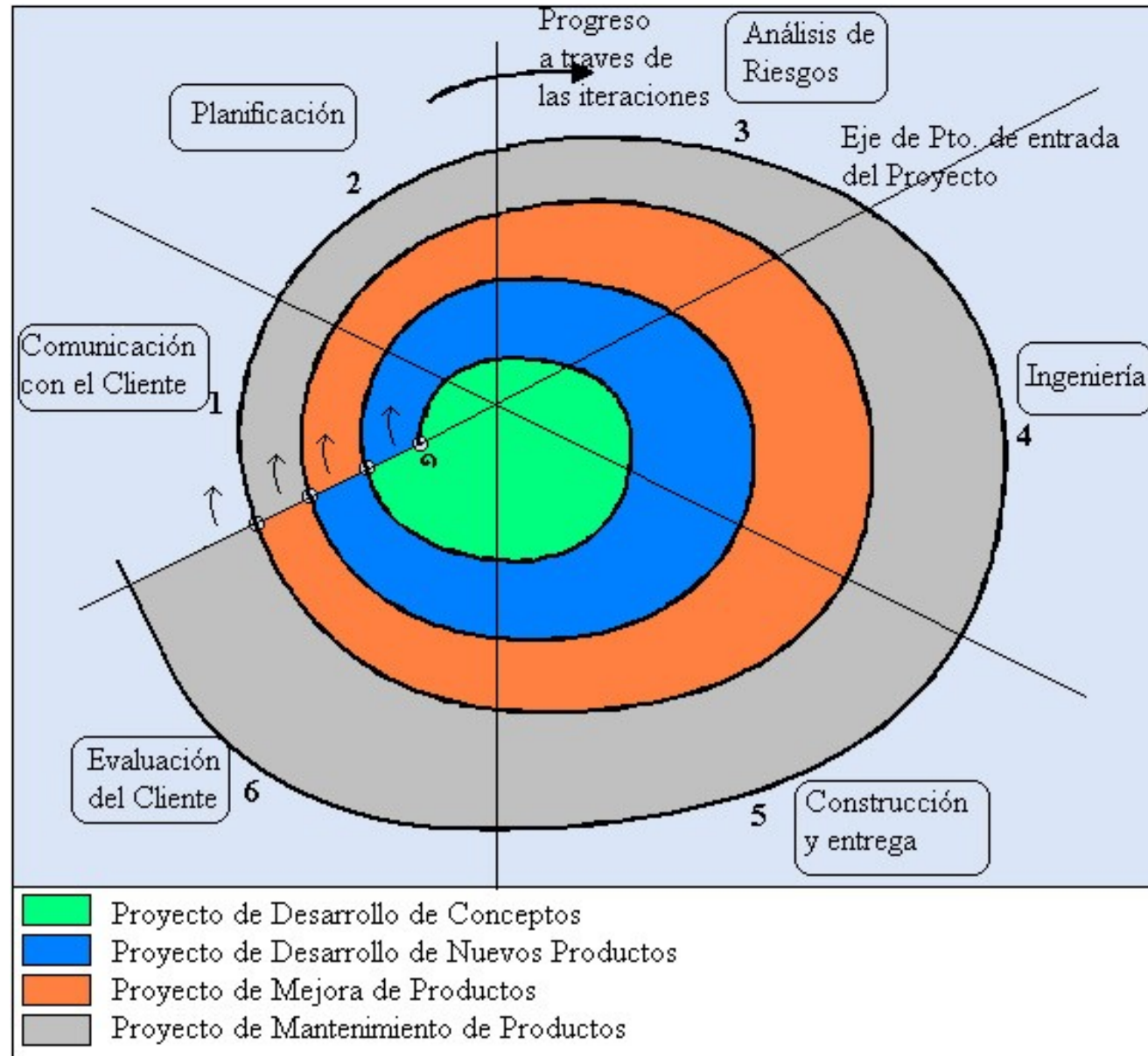
També ofereix un millor enfocament quan el responsable del desenvolupament del programari està insegur de l'eficàcia d'un algorisme, de l'adaptabilitat d'un sistema operatiu o de la forma que hauria de prendre la interacció humà-màquina.

- Inconvenients del **model de construcció de prototips**:

L'usuari tendeix a crear-se unes expectatives quan veu el prototip de cara al sistema final. A causa de la intenció de crear un prototip de forma ràpida, se solen **desatendre aspectes importants**, com ara la qualitat i el manteniment a llarg termini, la qual cosa obliga a la major part dels casos a reconstruir una vegada que el prototip ha complert la seva funció. És freqüent que l'usuari es mostri reticent a això i demani que sobre aquest prototip es construeixi el sistema final, el que el convertiria en un prototip evolutiu, però partint d'un estat poc recomanat.

Per tal de desenvolupar ràpidament el prototip, el desenvolupador sol prendre algunes decisions d'implementació **poc convenients** (per exemple, triar un llenguatge de programació incorrecte perquè proporcionï un desenvolupament més ràpid). Amb el pas del temps, el desenvolupador pot oblidar-se de la raó que el va portar a prendre aquestes decisions, de manera que es corre el risc que aquestes eleccions passin a formar part del sistema final.

Cicle de vida del software



Model en espiral

Cicle de vida del software

- El **model en espiral** és un model evolutiu que conjuga la naturalesa iterativa del Model de Prototips amb els aspectes controlats i sistemàtics del Model en Cascada.
- Proporciona potencial per a **desenvolupament ràpid de versions incrementals**. En el model Espiral el programari es construeix en una sèrie de versions incrementals. En les primeres iteracions la versió incremental podria ser un model en paper o bé un prototip. En les últimes iteracions es produeixen versions cada vegada més completes del sistema dissenyat.
- El model es divideix en un nombre d'Activitats de marc de treball, anomenades regions de tasques. En general hi ha entre tres i sis **regions de tasques** (hi ha variants del model).

-

Cicle de vida del software

- El **Model Espiral** és particularment apte per al desenvolupament de Sistemes Operatius (complexos); també en sistemes d'alts riscos o crítics (per exemple navegadors i controladors aeronàutics) i en tots aquells en què sigui necessària una forta gestió del projecte i els seus riscos, tècnics o de gestió.
- Aquest model no s'ha usat tant, com el Cascada (Incremental), de manera que no es té ben mesurada la seva eficàcia, és un paradigma relativament nou i difícil d'implementar i controlar.
- Desavantatges:

Requereix molta experiència i habilitat per a l'avaluació dels riscos, la qual cosa és requisit per a l'èxit del projecte.

És difícil convèncer els grans clients que podran controlar aquest enfocament evolutiu.

Agile

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

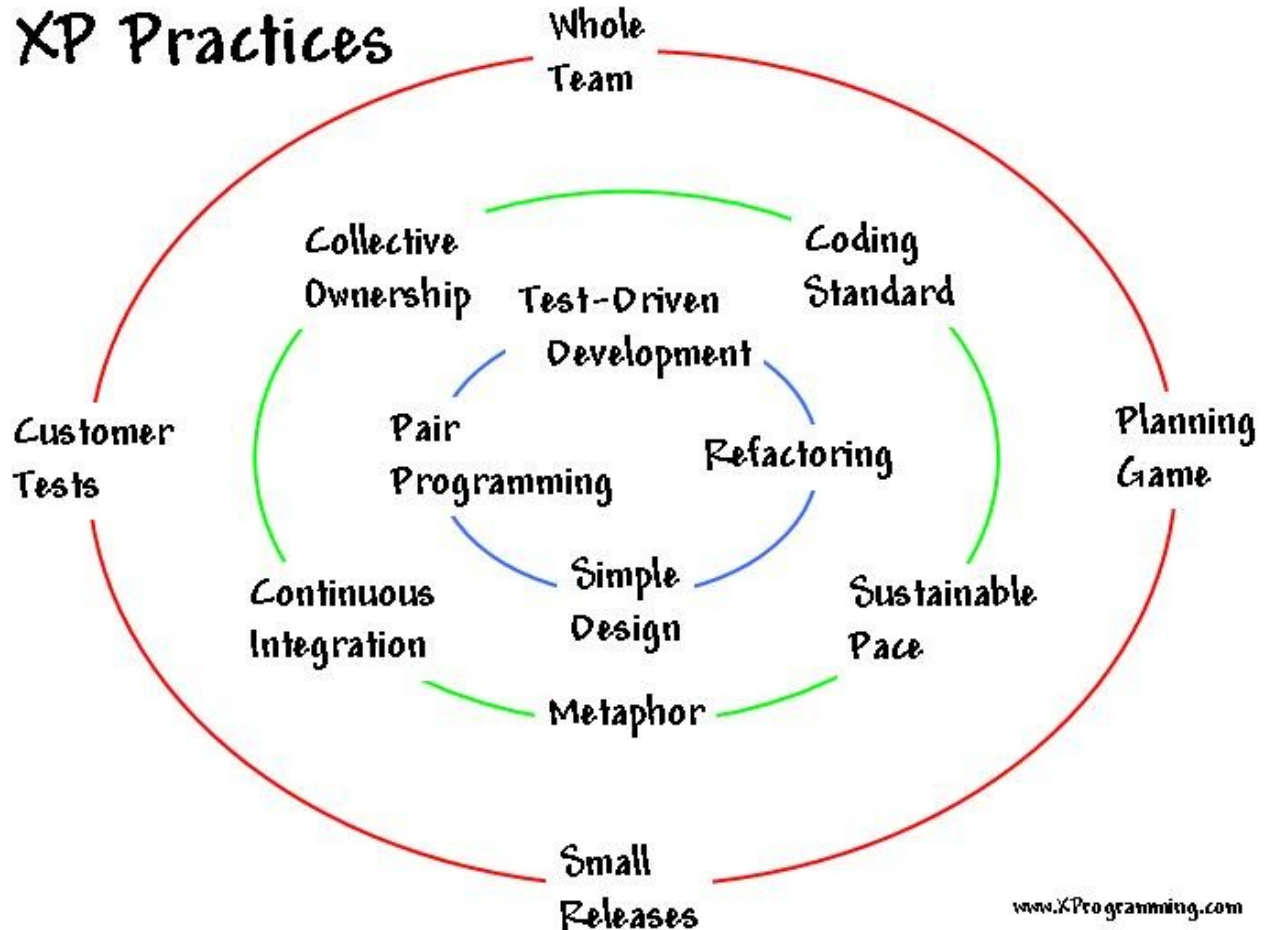
<http://agilemanifesto.org/>

eXtreme Programing

Values

- Communication
- Simplicity
- Feedback
- Courage
- Respect

XP Practices

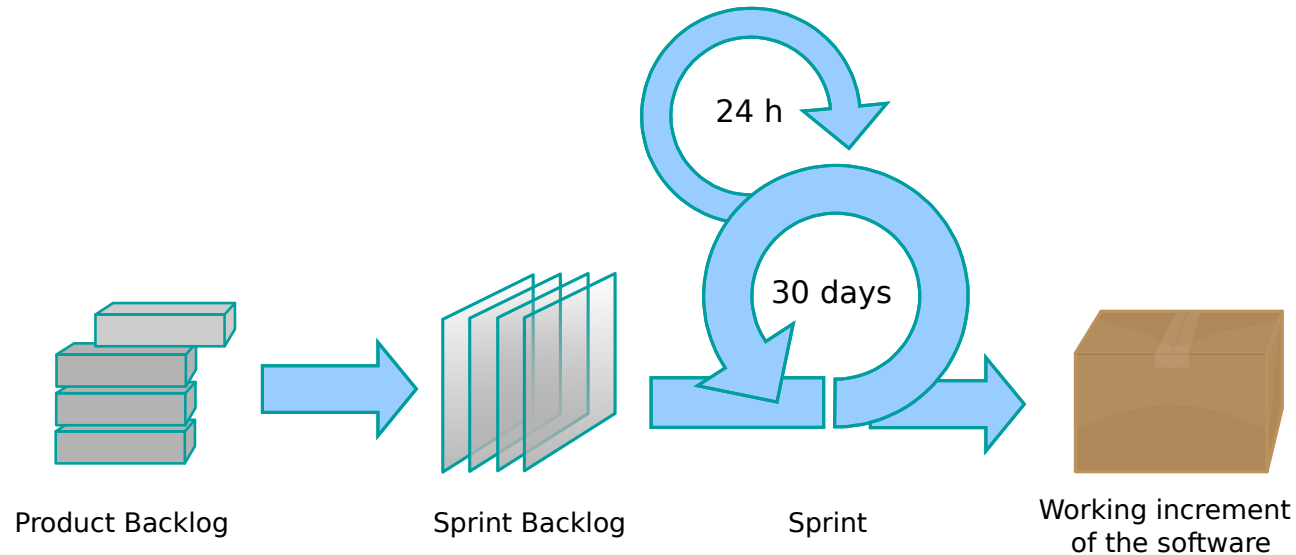


www.XProgramming.com

<http://www.youtube.com/watch?v=W8CuRtCHWD8>

Cicle de vida del software

Scrum



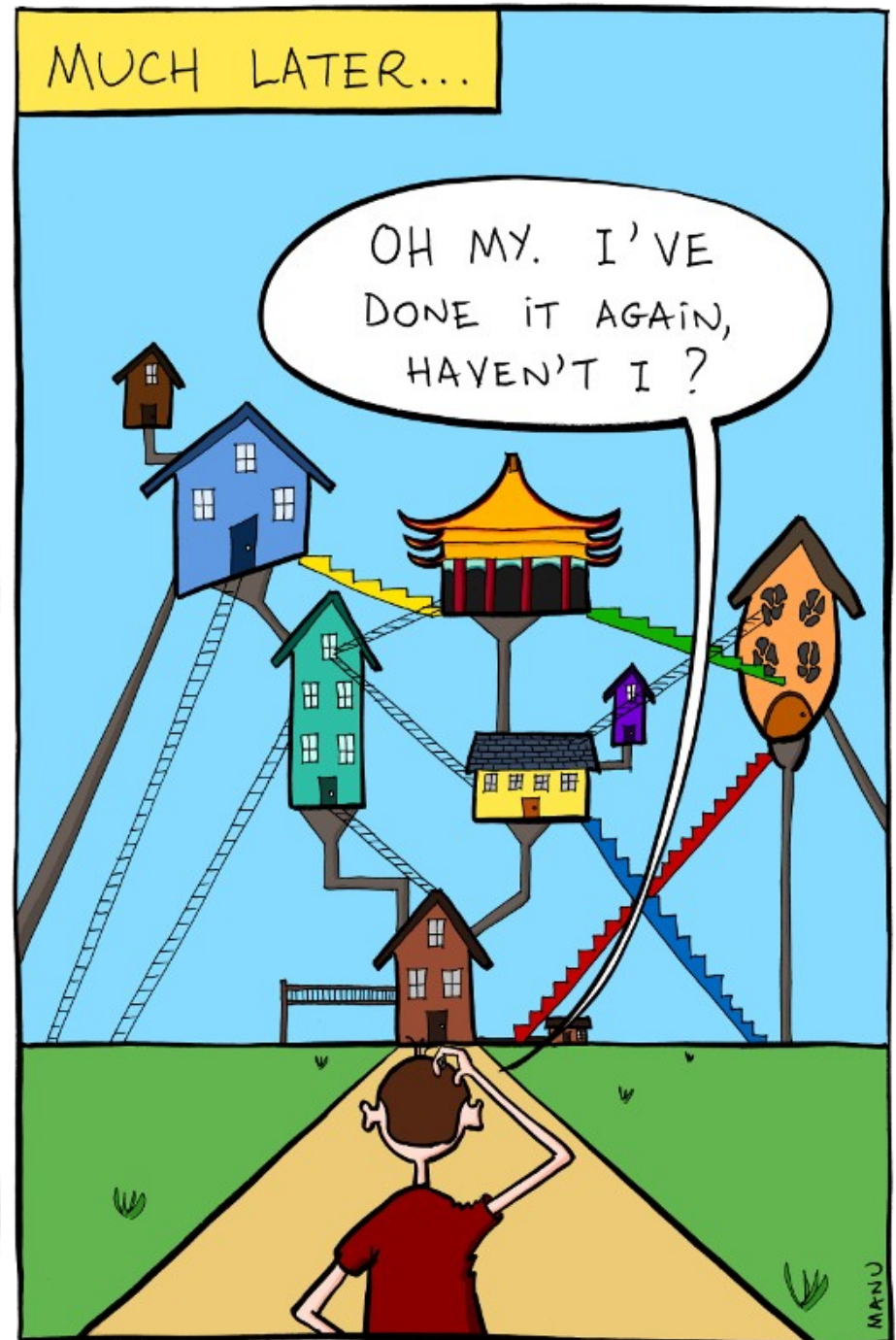
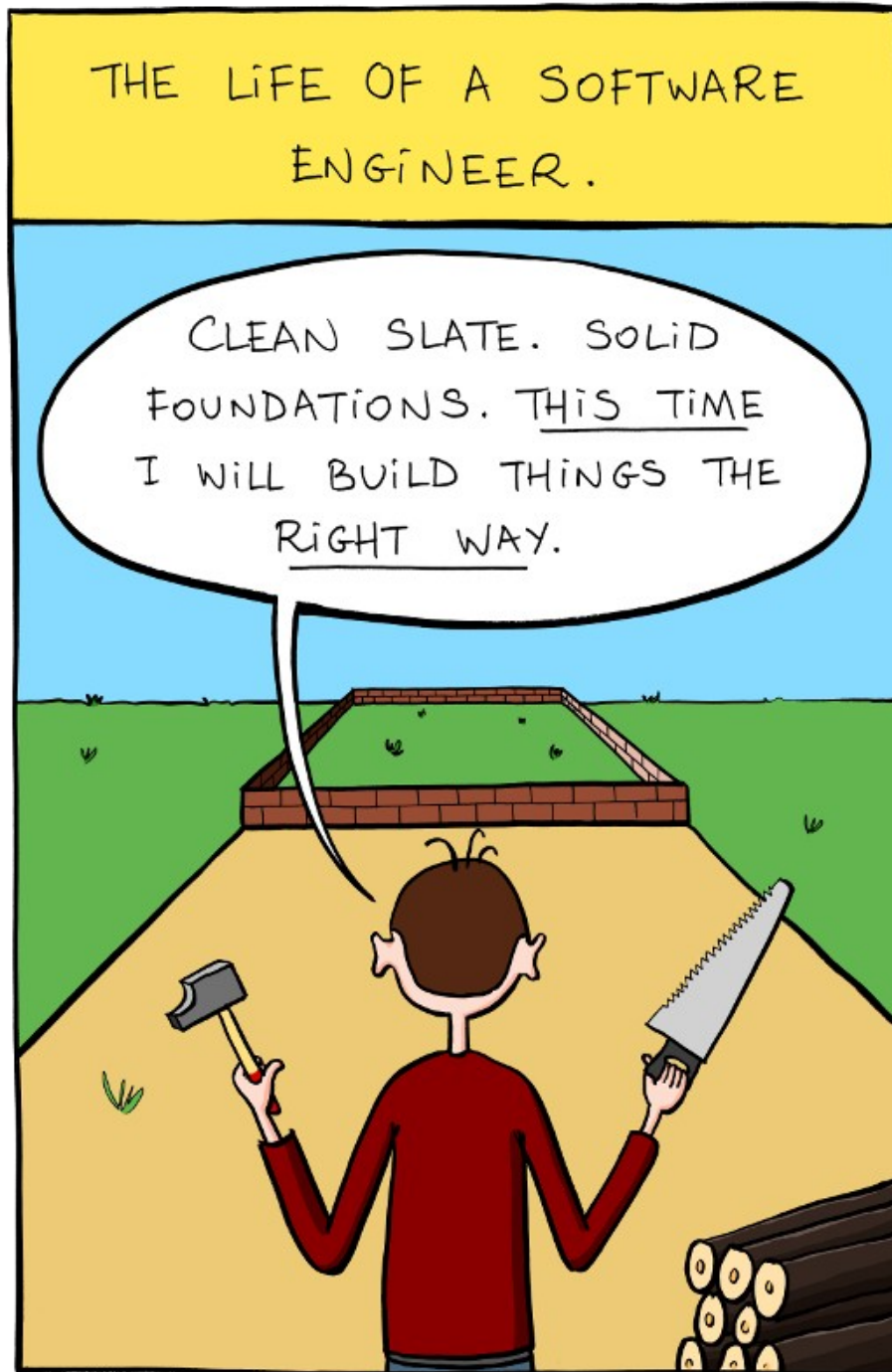
<http://www.allaboutagile.com/>
<http://agilevideos.com/>

<http://agilevideos.com/videos/scrum-101-part-1-scrum-basics/>
<http://agilevideos.com/videos/scrum-101-part-2-the-scrum-process/>
<http://vimeo.com/4587652>

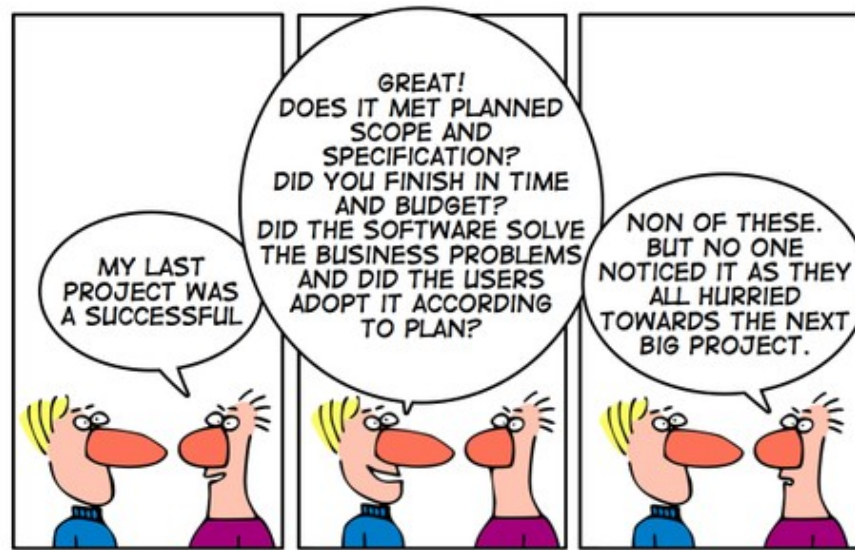
Cicle de vida del software

- **Scrum** és un model de desenvolupament àgil caracteritzat per:
- Adoptar una estratègia de desenvolupament incremental, en lloc de la planificació i execució completa del producte.
- Basar la qualitat del resultat més en el coneixement tàcit de les persones en equips autoorganitzats, que en la qualitat dels processos emprats.
- Solapament de les diferents fases del desenvolupament, en lloc de realitzar una després d'una altra en un cicle seqüencial o de cascada.

Humor



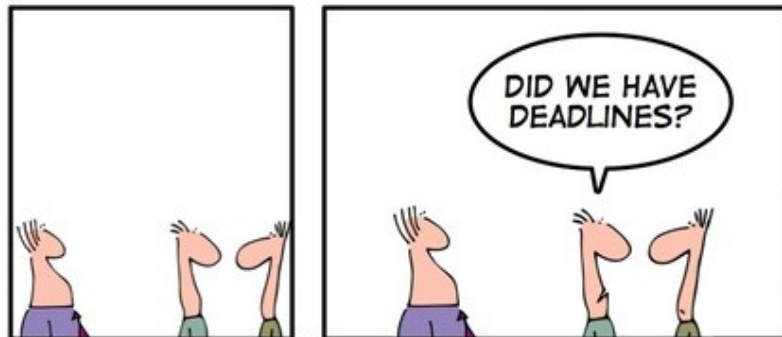
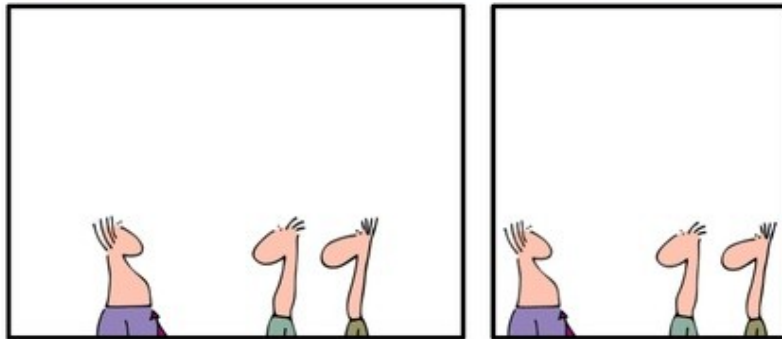
Humor



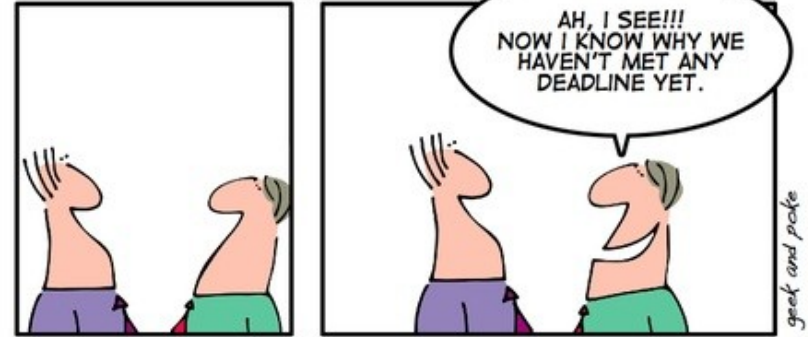
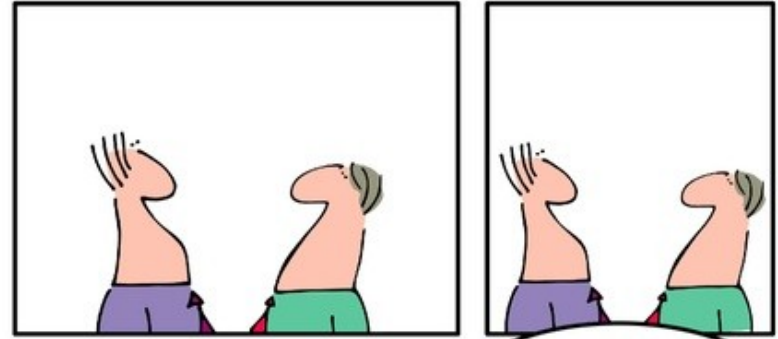
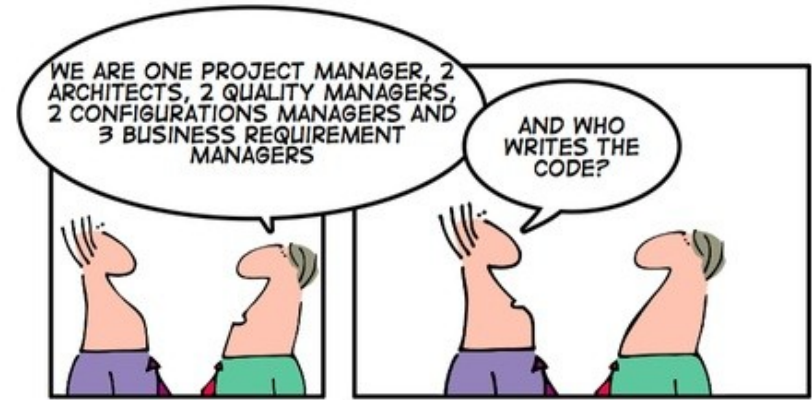
ONE YEAR IN A IT PROJECT - DAY 4:
SOMETIMES IT'S GOOD TO LOOK BACK TO PAST SUCCESSES



THE CONSULTANTS HANDBOOK PART 4:
A GOOD DOCUMENTATION IS ESSENTIAL FOR THE SUCCESS OF YOUR PROJECT



**HOW TO RESCUE A PROJECT - CHAPTER 1:
COMMUNICATE**



**HOW TO RESCUE A PROJECT - THE LAST CHAPTER:
SOMETIMES IT'S JUST THE RIGHT QUESTION**

Humor



How the customer explained it



How the Project Leader understood it



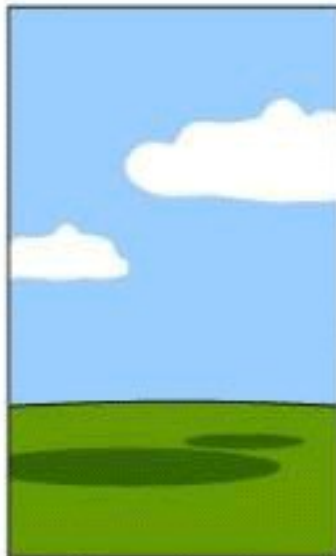
How the Analyst designed it



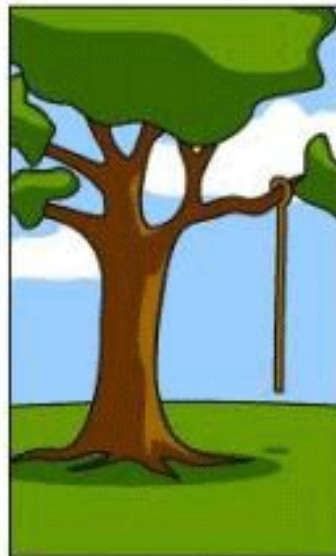
How the Programmer wrote it



How the Business Consultant described it



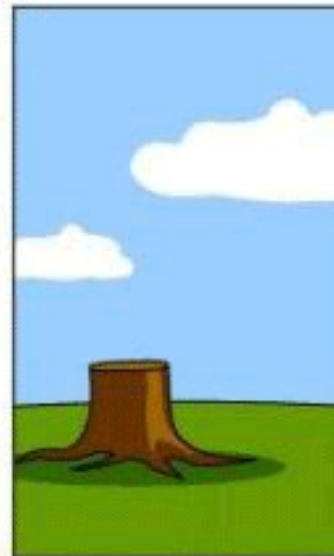
How the project was documented



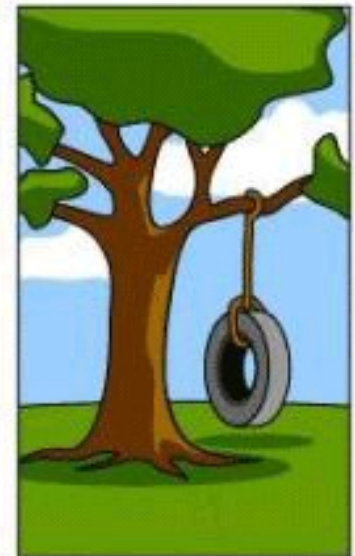
What operations installed



How the customer was billed



How it was supported



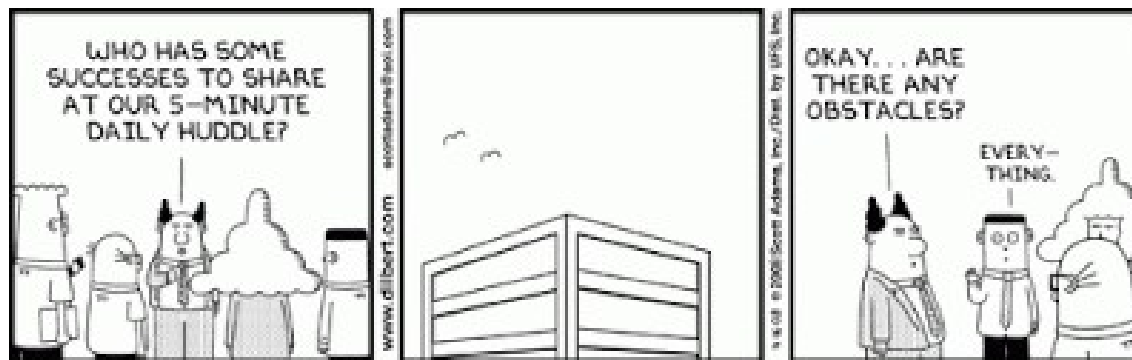
What the customer really needed



© Scott Adams, Inc./Dist. by UFS, Inc.



Copyright © 2003 United Feature Syndicate, Inc.



© Scott Adams, Inc./Dist. by UFS, Inc.

Copyright (c) 2012 Mònica Ramírez Arceda

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/>.



Attribution-ShareAlike 3.0 Unported