DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Planificación

Fernando Berzal, berzal@acm.org

Planificación

- La crisis "crónica" del software
- Planificación temporal
- Descomposición en tareas
- Camino crítico
  - CPM
  - PERT
- Holguras [slack]
- Prioridades
  - MMF
  - Modelo en espiral
- Seguimiento del proyecto
- Objetivos y restricciones

Planificación

La crisis del software

1st NATO Software Engineering Conference
Garmisch, Germany, 1968

Software Engineering: "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"

Planificación

La crisis del software

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

— **Edsger Dijkstra**, "The Humble Programmer" (EWD340), Communications of the ACM, 1972 Turing Award Lecture

https://en.wikipedia.org/wiki/Software_crisis

Planificación

La crisis del software

The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet--something to make software costs drop as rapidly as computer hardware costs do.

-- **Frederick P. Brooks, Jr.**:
"No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer 20(4):10-19, April 1987

Planificación

La crisis "crónica" del software

- En media, los proyectos de software consumen un 50% más de tiempo del planeado inicialmente (aún peor en los grandes proyectos).
- El 75% de los grandes sistemas presentan fallos (no funcionan como se esperaba o, simplemente, no funcionan).
- El 25% de los grandes proyectos se cancelan.

W. Wayt Gibbs: "Software's Chronic Crisis"
Scientific American, September 1994

Planificación

La crisis "crónica" del software


SPECIAL REPORT

AIR JAM: The U.S. Federal Aviation Administration spent $2.6 billion trying to revamp its air-traffic-control software. Why? To cancel the project in 1994. Gridlocked skies are still with us today.

Why Software FAILS

We waste billions of dollars each year on entirely preventable mistakes

By Robert N. Charette

Have you heard the one about the disappearing warehouse? One day, it vanished—not from physical view, but from the watchful eyes of a well-known retailer's automated distribution system. A software glitch had somehow erased the warehouse's existence, so that goods destined for the warehouse were rerouted elsewhere, while goods at the warehouse languished. Because the company was in financial trouble and had been shuttering other warehouses to save money, the employees at the "missing" warehouse kept quiet. For three years, nothing arrived or left. Employees were still getting their paychecks, however, because a different computer system handled the payroll. When the software glitch finally came to light, the merchandise in the warehouse was sold off, and upper management told employees to say nothing about the episode.

42 IEEE Spectrum | September 2005 | NA

www.spectrum.ieee.org

Robert N. Charette: "Why software fails",
IEEE Spectrum, September 2005

Planificación

La crisis "crónica" del software

- Los proyectos más grandes tienen una probabilidad mayor de fracasar que los proyectos pequeños, principalmente por motivos sociológicos y de comunicación.
  Tom DeMarco & Timothy Lister: **Peopleware**, 2nd edition, 1999

- Las opciones de éxito de que un proyecto se complete con éxito desaparecen casi por completo en proyectos de gran escala.
  Edward Yourdon: **Death March**, 2004

Planificación

¿Por qué se retrasan los proyectos?

- **Plazos** poco realistas (impuestos desde fuera).
- **Cambios** en los requisitos del cliente (que no se reflejan en cambios en la planificación).
- Subestimación [honesta] del **esfuerzo** requerido.
- **Riesgos** no considerados al inicio del proyecto.
- Fallos de **comunicación**.
- Errores de **gestión** al no reconocer que el proyecto se está retrasando con respecto al plan (y ausencia de acciones correctivas).

Planificación

"Excessive or irrational schedules are probably the single most destructive influence in all of software."

-- Capers Jones

"Overly optimistic scheduling doesn't result in shorter actual schedules, it results in longer ones."

"Every hour of planning saves about a day of wasted time and effort."

-- Steve McConnell

Planificación

Tareas de planificación

- Establecer el ámbito del proyecto.
- Determinar su viabilidad.
- Analizar posibles riesgos.
- Determinar los recursos necesarios.
- Estimar coste y esfuerzo.
- Desarrollar una planificación temporal del proyecto.

Planificación temporal

Principios

- Compartimentación
  (definición de tareas)
- Interdependencias
  (identificación de relaciones entre tareas)
- Recursos
  (¿están disponibles los recursos necesarios?)

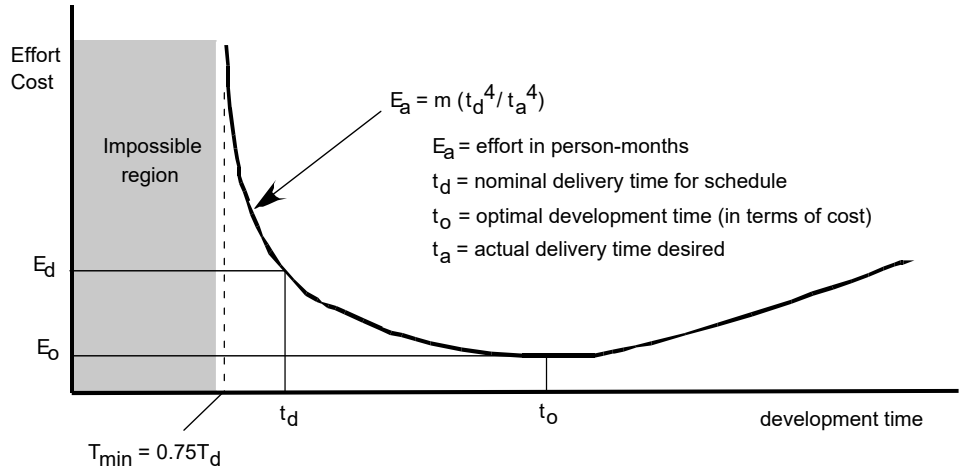
Planificación temporal

Principios

- Responsabilidades
  (asignar un responsable concreto para cada tarea)
- Resultados
  (definición de los resultados esperados de cada tarea)
- Hitos
  (revisión de puntos clave del proyecto)

Planificación temporal

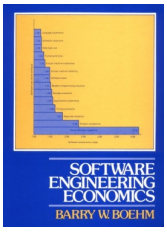
Curva PNR [Putnam-Norden-Rayleigh]



Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Planificación temporal

La región imposible



Relación entre el tiempo de desarrollo de un proyecto y el número de personas-mes necesarias:

$$T > 2.15 \sqrt[3]{PM}$$

El 99% de los proyectos completados obedecen esta regla: ¿Algún motivo para pensar que su proyecto es diferente?

Barry Boehm: "Software Engineering Economics", 1981

Planificación temporal

Asignación de recursos

Regla 40-20-40

40-50% actividades iniciales [front end]

- Comunicación con el cliente y análisis (10-25%)
- Diseño (20-25%)
- Revisiones y modificaciones

15-20% actividades de construcción (implementación)

30-40% pruebas

Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Planificación temporal

Asignación de recursos

Fase	Tiempo	Esfuerzo
Análisis de requisitos	12%	6%
Diseño preliminar	8%	8%
Diseño detallado	15%	16%
Implementación	30%	40%
Pruebas del sistema	20%	20%
Pruebas de aceptación	15%	10%

Software Engineering Laboratory (SEL)
Relationships, Models, and Management Rules
NASA Software Engineering Laboratory, SEL-91-001, 1991.

Descomposición en tareas

"The perfect project plan is possible if one first documents a list of all the unknowns."
— Bill Langley

Descomposición en tareas

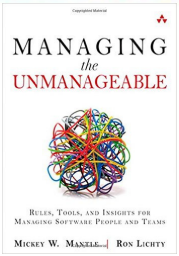
WBS [Work Breakdown Structure]

1.1 **Concept scoping** determines the overall scope of the project.

```
Task definition:  Task 1.1  Concept Scoping
1.1.1          Identify need, benefits and potential customers;
1.1.2          Define desired output/control and input events that drive the application;
               Begin Task 1.1.2
                  1.1.2.1        FTR [formal technical review]:  Review written description of need
                  1.1.2.2        Derive a list of customer visible outputs/inputs
                  1.1.2.3        FTR:  Review outputs/inputs with customer and revise as required;
               endtask Task 1.1.2
1.1.3          Define the functionality/behavior for each major function;
               Begin Task 1.1.3
                  1.1.3.1        FTR:  Review output and input data objects derived in task 1.1.2;
                  1.1.3.2        Derive a model of functions/behaviors;
                  1.1.3.3        FTR:  Review functions/behaviors with customer and revise as required;
               endtask Task 1.1.3
1.1.4          Isolate those elements of the technology to be implemented in software;
1.1.5          Research availability of existing software;
1.1.6          Define technical feasibility;
1.1.7          Make quick estimate of size;
1.1.8          Create a Scope Definition;
endTask definition:   Task 1.1
```

Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Descomposición en tareas



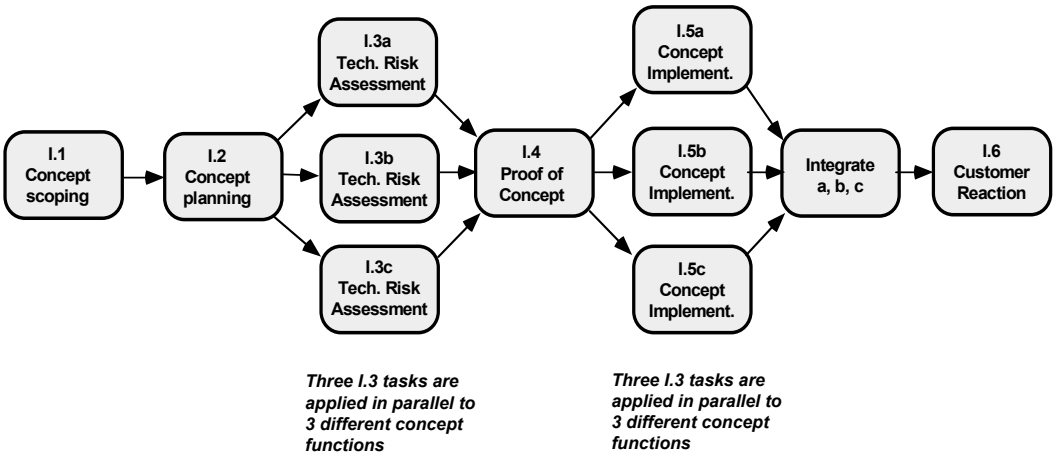
HEURÍSTICA

Al descomponer actividades en tareas individuales, las tareas deberías ser lo suficientemente grandes como para requerir al menos dos días de trabajo, pero lo suficientemente pequeñas para no requerir más de una semana.

- Las tareas de menos de dos días tienen a "rellenarse".
- Las tareas de más de una semana indican que no se entiende realmente qué es necesario.

Descomposición en tareas

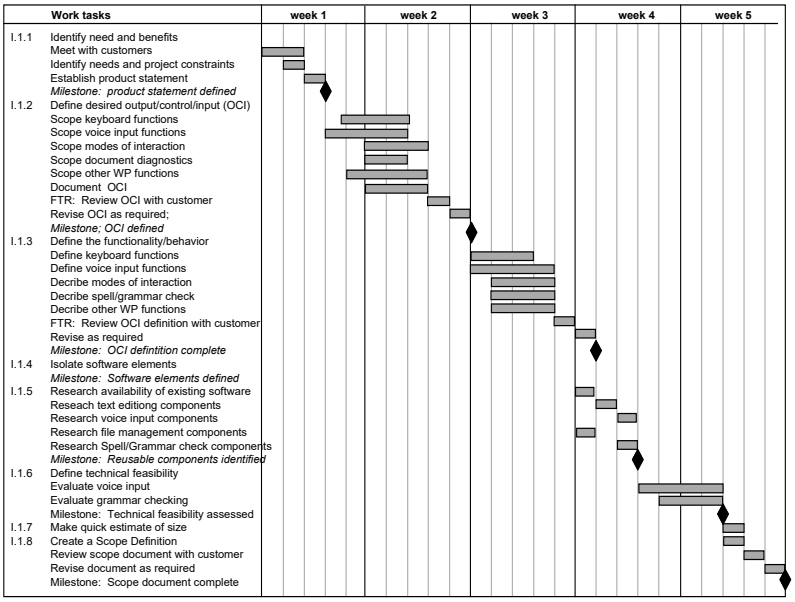
Red de tareas o actividades



Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Descomposición en tareas

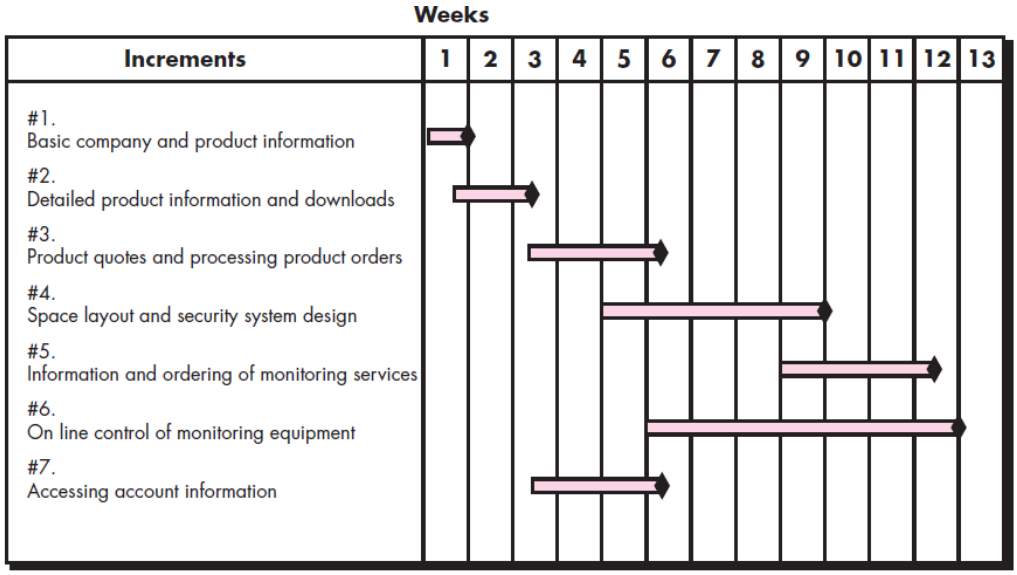
Diagrama de Gantt



Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Descomposición en tareas

Diagrama de Gantt



Roger S. Pressman:
"Software Engineering: A Practitioner's Approach", 7th edition, 2009

Descomposición en tareas

Diagrama de Gantt

- Tareas/actividades (fecha de inicio y duración)
- Dependencias entre actividades
- Hitos (rombos)
- Asignación de recursos y ajuste del calendario (con ayuda de herramientas software)

Camino crítico

Conjunto de tareas que deben completarse de acuerdo al plan si queremos que el proyecto en su conjunto termine en la fecha establecida.

Los métodos de planificación de proyectos proporcionan herramientas para determinar el camino crítico:

- **CPM** [Critical Path Method]
- **PERT** [Program Evaluation and Review Technique]

CPM

Dadas las duraciones de cada tarea
y las dependencias existentes entre ellas:

- **ES** [Earliest Start]: Comienzo más temprano
  $ES(t) = \max_{p \to t} \{ ES(p) + duración(p) \}$
  siendo ES(t)=0 para las tareas sin predecesoras.

- **LS** [Latest Start]: Comienzo más tardío
  $LS(t) = \min_{s \leftarrow t} \{ LS(s) - duración(t) \}$
  siendo LS(t)=ES(t) para las tareas sin sucesoras.

James Kelley (Remington Rand) & Morgan Walker (DuPont)
"Critical-Path Planning and Scheduling"
Proceedings of the Eastern Joint Computer Conference, 1959.

CPM

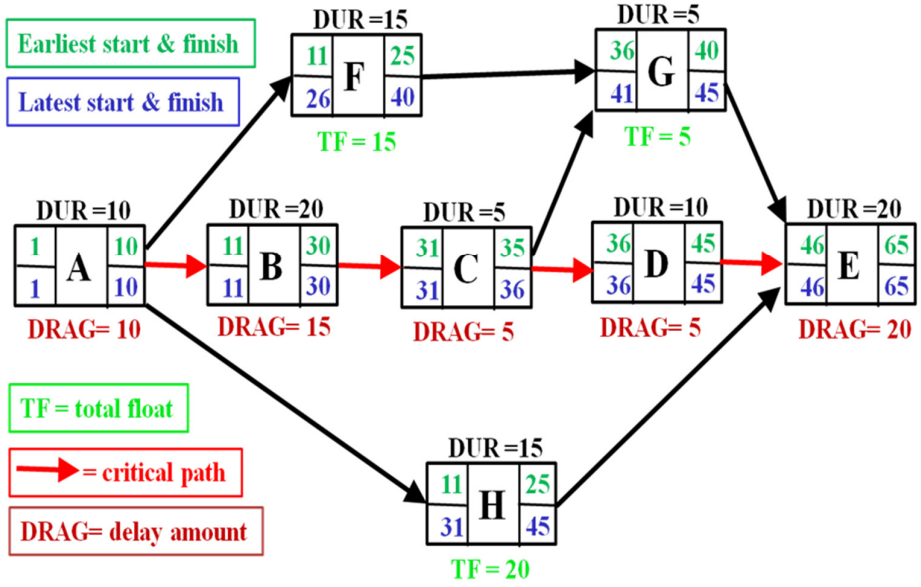
Dadas las duraciones de cada tarea
y las dependencias existentes entre ellas:

- **Holgura [slack, a.k.a. float]**
  slack(t) = LS(t) - ES(t)

- **Actividades críticas**:
  Tareas sin holgura.

Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

CPM



PERT

Project Evaluation and Review Technique
(U.S. Navy, 1950's)

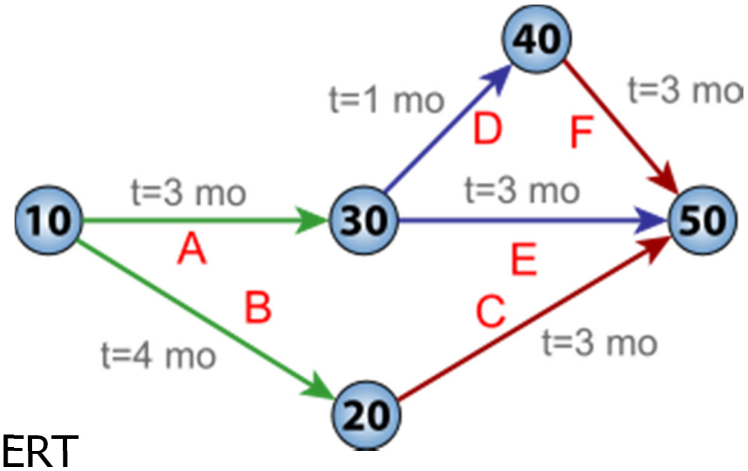


Diagrama PERT

- Actividades (arcos)
- Hitos (nodos)

PERT

Estimaciones

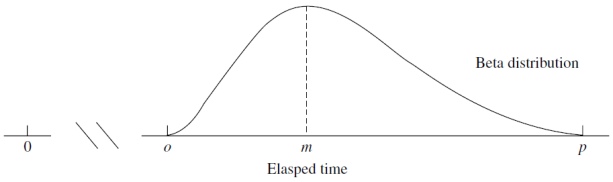
- O [optimista]: Tiempo mínimo posible requerido para realizar una tarea, asumiendo que todo va mejor de lo que normalmente se espera.
- P [pesimista]: Tiempo máximo posible requerido para realizar una tarea, asumiendo que todo va mal (salvo catástrofes que impidan la realización de la tarea).
- M [most-likely] / N [normal]:
  Mejor estimación del tiempo necesario para completar la tarea, en condiciones normales.

PERT

Estimaciones

Tiempo esperado

$T_E = (O+4M+P) / 6$



Tarea	Pred.	O	N	P	T_E
A	-	1	3	5	3.00
B	-	2	4	6	4.00
C	B	2	3	4	3.00
D	A	1	1	2	1.17
E	A	2	3	7	3.50
F	D	2	3	6	3.33

PERT

Camino crítico

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days

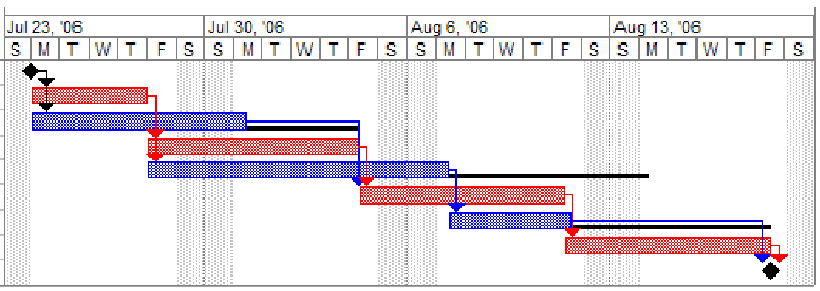


Diagrama de Gantt

Microsoft Project

PERT

Camino crítico

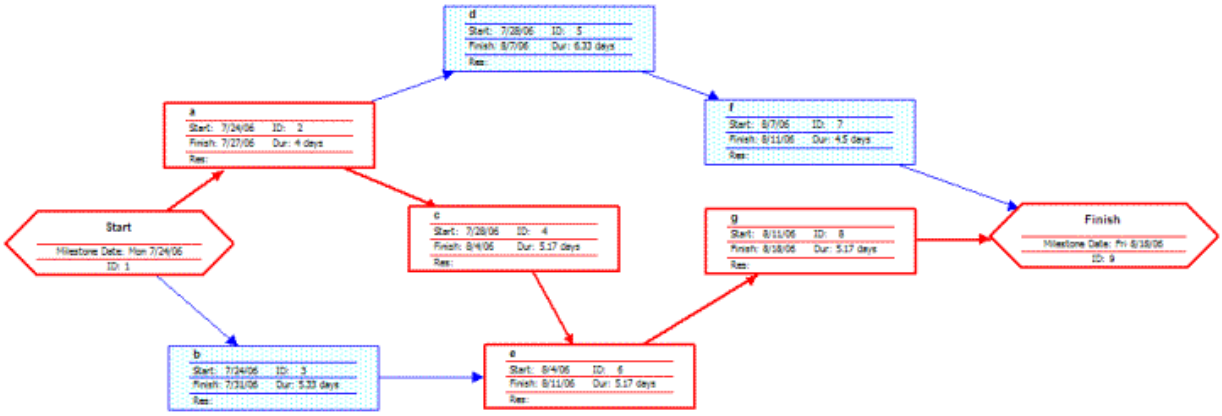


Diagrama AON [activity on node]

Microsoft Project

Slack

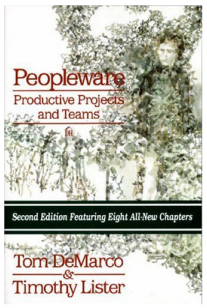
En un proyecto de desarrollo de software, existen tantas cosas que no se pueden planificar (p.ej. instalación y prueba de nuevas versiones del sistema operativo o del compilador), que intentar hacer un seguimiento de todo es imposible.

Es aconsejable reservar una parte de tiempo (15%-20%) sin planificar para englobar a todas esas tareas que son necesarias para que el proyecto tenga éxito.

Ley de Parkinson: "El trabajo se expande hasta llenar todo el tiempo disponible"

Slack

En un trabajo creativo no se está el 100% del tiempo "haciendo" cosas (como en una cadena de montaje).



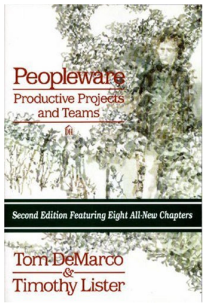
Hay que provisionar tiempo para…

- Intercambiar ideas, p.ej. brainstorming.
- Investigar nuevos métodos, técnicas y herramientas.
- Averiguar cómo evitar determinadas subtareas.
- Leer.
- Probar alternativas.
- … y "perder el tiempo" (visto desde fuera ;-)

Slack

"Spanish Theory managers"

Aumentar niveles de productividad mediante horas extra no pagadas (con la ayuda de "juegos mentales" con sus empleados, p.ej. sentimiento de culpa, seguridad del puesto de trabajo…)



- "Teoría española del valor" [según los historiadores] (existe una cantidad fija de valor sobre la Tierra). vs. "Teoría inglesa del valor" (la ingenuidad y la tecnología permiten crear valor).

- Resultado: Una hora "libre" por cada hora extra…

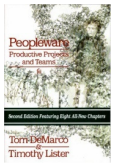
Slack

Estrategias típicas para aumentar la productividad:

- Presionar a los empleados para que echen más horas.
- Mecanizar el proceso de desarrollo.
- Comprometer la calidad del producto.
- Estandarizar procedimientos
  (como si las personas fuesen partes intercambiables).

Todas esas medidas pueden aumentar el estrés, reducir la satisfacción en el trabajo y destruir la motivación… además de causar bajas en el equipo de desarrollo.

People under pressure don't work better; they just work faster **-- Tom DeMarco**



Prioridades

- Establecer prioridades es fundamental:
  características esenciales vs. características deseables.

  p.ej. MMF [minimum marketable features]

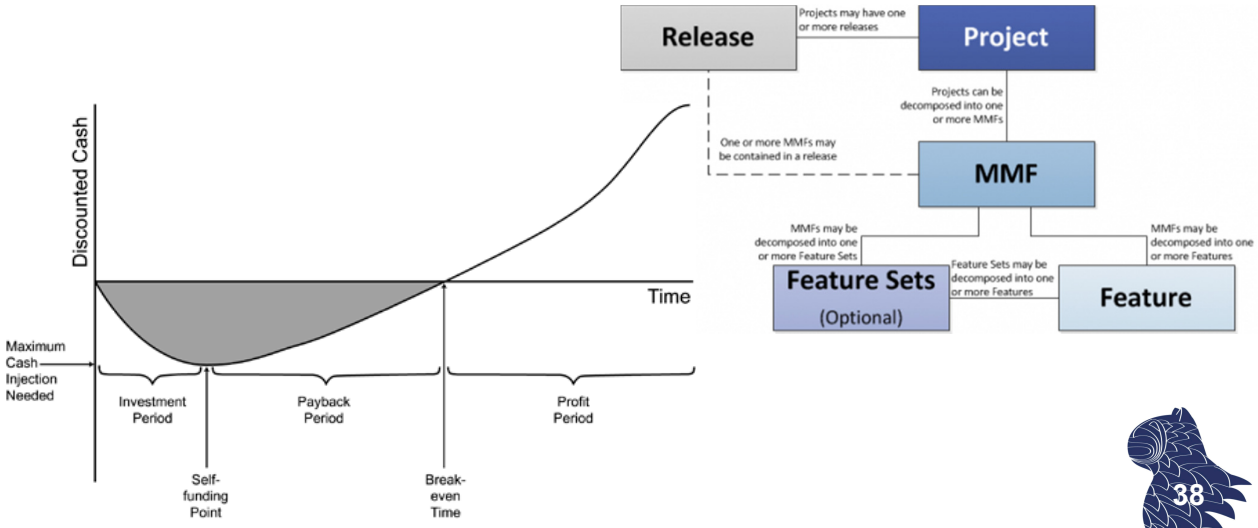
- Identificar riesgos y darles prioridad a las actividades que los minimizan o eliminan es de igual importancia:
  "risk-first development"

  p.ej. Modelo en espiral de Boehm

Prioridades

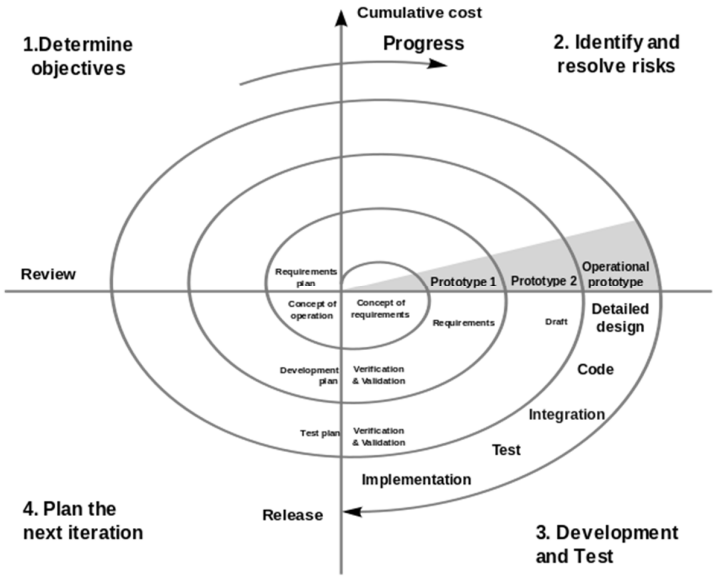
MMF: Minimum Marketable Features

Mark Denne & Jane Cleland-Huang:
"Software by Numbers: Low-Risk, High-Return Development"
Prentice Hall PTR, 2003. ISBN 0-13-140728-7



Prioridades

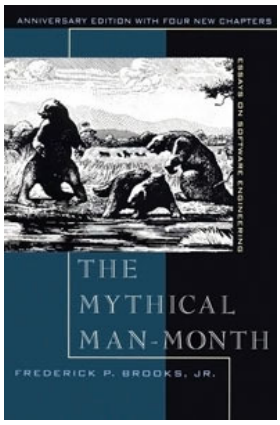
Modelo en espiral de Boehm



Barry Boehm:
"A Spiral Model of Software Development and Enhancement",
ACM SIGSOFT Software Engineering Notes 11(4):14-24, 1986

Seguimiento del proyecto

"How does a project get to be a year late?...
One day at a time."
-- Frederick P. Brooks

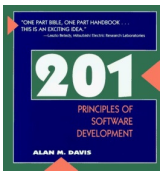


Seguimiento del proyecto

Periódicamente, se realizan reuniones de seguimiento [project status meetings]:

- Cada miembro del equipo informa del progreso del proyecto y de los problemas que han aparecido.
- Se evalúan los resultados de las revisiones realizadas.
- Se determina si se han alcanzado los hitos del proyecto en las fechas previstas.
- Se comparan las fechas de inicio previstas con las fechas de inicio reales para cada tarea.

Seguimiento del proyecto



Hay que mantener actualizado el plan del proyecto:

Un plan no actualizado es peor que no tener plan (puede inducir a pensar que todo está bajo control).

Don Reifer: "The Nature of Software Management: A Primer", Tutorial: Software Management, 1986

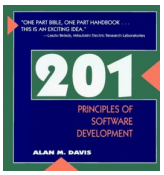
Seguimiento del proyecto

¿Y si utilizamos metodologías ágiles?

Agile planning is deceptive. I'm always amused, and sometimes saddened, by the lack of understanding about agile planning... First, agile teams do a lot of planning, but it is spread out much more evenly over the entire project. Second, agile teams squarely face the critical factor that many non-agile teams ignore—uncertainty.

-- Jim Highsmith, *Agile Estimating and Planning*

Seguimiento del proyecto



Un proyecto retrasado difícilmente se recupera.

Cada desliz en el cumplimiento del plan previsto ha de comunicarse y se han de analizar las estrategias alternativas que permitan seguir avanzando (p.ej. recortar funcionalidad, aumentar plazos… pero nunca recortar calidad buscando atajos).

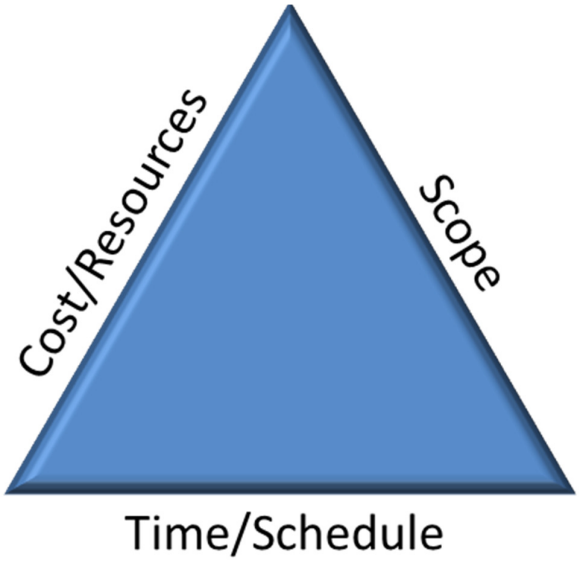
Tom Gilb: "Principles of Software Engineering Management", 1988

Seguimiento del proyecto

Formas de "comprimir" el calendario de un proyecto:

- Paralelizar determinadas tareas.
- Compromiso tiempo vs. esfuerzo [schedule vs. effort].
- Reconsiderar la inclusión de algunas características.
- Resolver conflictos para liberar a personal clave.
- Desarrollo incremental

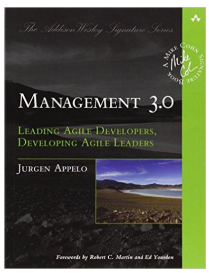
Objetivos y restricciones

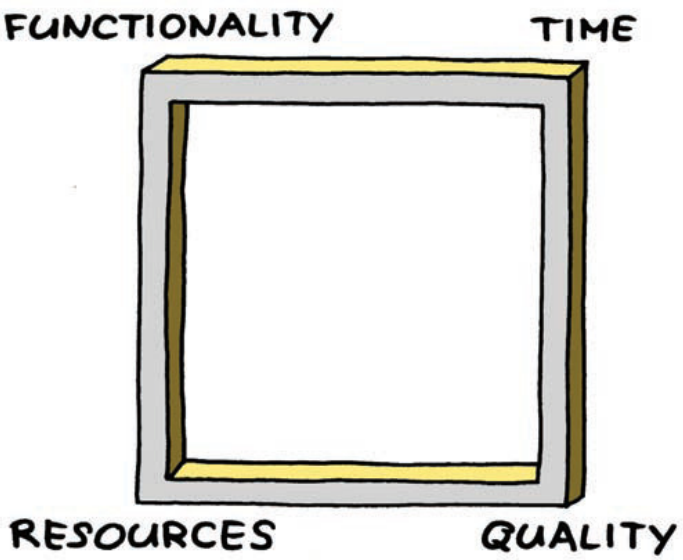


"I can make it for you fast, cheap, or good. Pick any two."
https://en.wikipedia.org/wiki/Project_management_triangle

Objetivos y restricciones

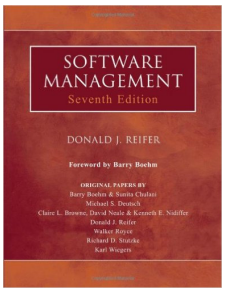
La calidad también impone restricciones…

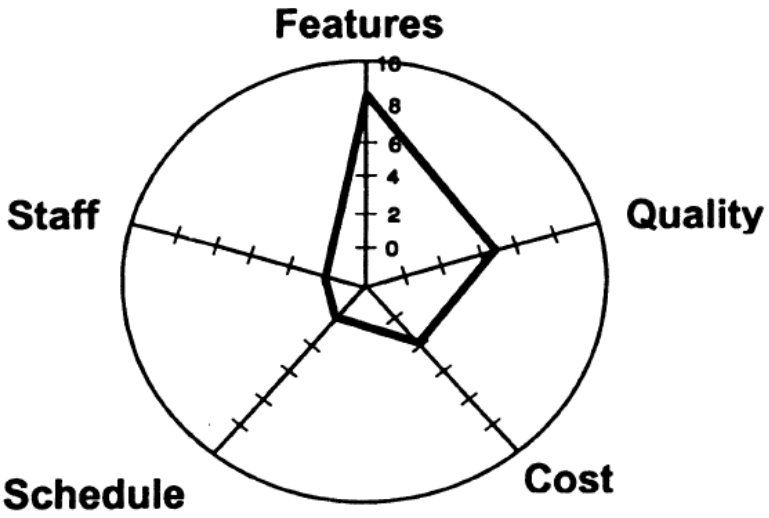




Objetivos y restricciones

Representación mediante un diagrama de Kiviat: "Diagrama de flexibilidad"

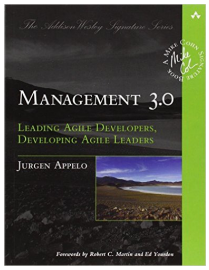


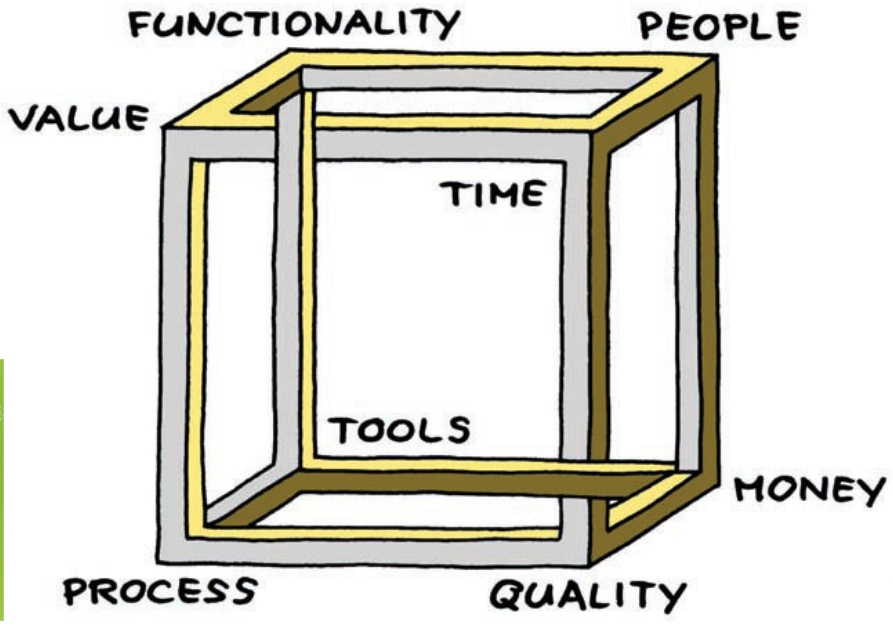


Karl E. Wiegers: "Creating a Software Engineering Culture", 1996

Objetivos y restricciones

Puede ser difícil equilibrar todas las dimensiones clave…

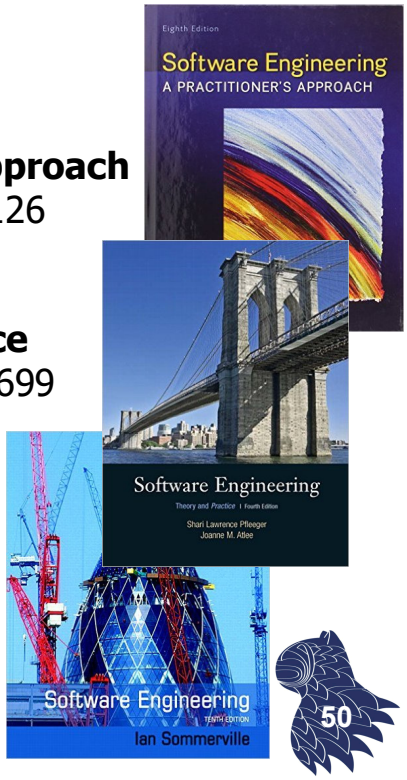




Bibliografía

Libros de texto

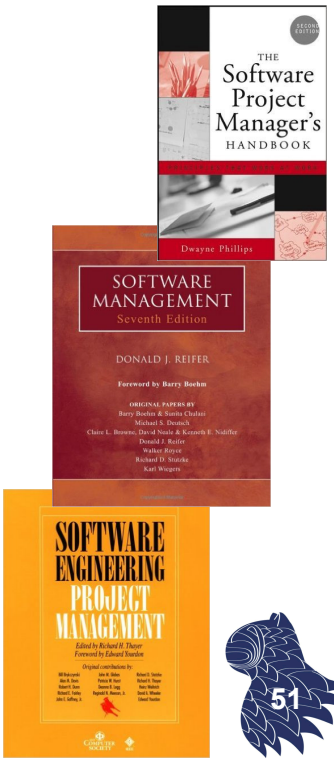
- Roger S. Pressman:
  Software Engineering: A Practitioner's Approach
  McGraw-Hill, 8th edition, 2014. ISBN 0078022126
- Shari Lawrence Pfleeger & Hoanne M. Atlee:
  Software Engineering: Theory and Practice
  Prentice Hall, 4th edition, 2009. ISBN 0136061699
- Ian Sommerville:
  Software Engineering
  Pearson, 10th edition, 2015. ISBN 0133943038



Bibliografía

Lecturas recomendadas

- Dwayne Phillips:
  The Software Project Manager's Handbook: Principles That Work at Work
  Wiley / IEEE Computer Society, 2nd edition, 2004
  ISBN 0471674206
- Donald J. Reifer (editor):
  Software Management
  Wiley / IEEE Computer Society, 7th edition, 2006
  ISBN 0471775622
- Richard H. Thayer (editor):
  Software Engineering Project Management
  Wiley / IEEE Computer Society, 2nd edition, 2000
  ISBN 0818680008



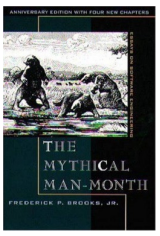
Bibliografía complementaria

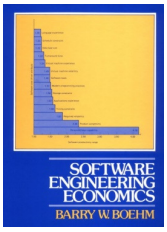
Ingeniería del Software

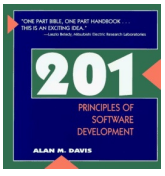
- Robert L. Glass:
  Facts and Fallacies of Software Engineering
  Addison-Wesley, 2003. ISBN 0321117425
- Barry W. Boehm & Richard Turner:
  Balancing Agility and Discipline: A Guide for the Perplexed Addison-Wesley, 2003. ISBN 0321186125
- Steve Tockey: **Return on Software:**
  Maximizing the return on your software investment
  Addison-Wesley Professional, 2004. ISBN 0321228758
- Steve McConnell:
  Software Project Survival Guide
  Microsoft Press, 1997. ISBN 1572316217
- Steve McConnell
  Rapid Development: Taming wild software schedules
  Microsoft Press, 1996. ISBN 1556159005

Bibliografía complementaria

Clásicos





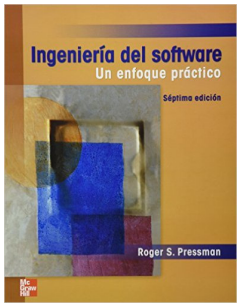


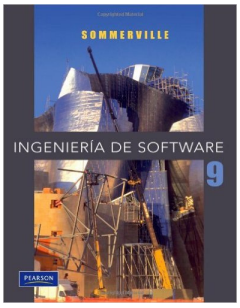
- Frederick P. Brooks, Jr.:
  The Mythical Man-Month:
  Essays on Software Engineering
  Addison-Wesley, 1995. ISBN 0201835959
- Alan M. Davis:
  201 Principles of Software Development
  McGraw-Hill, 1995. ISBN 0070158401
- Barry W. Boehm:
  Software Engineering Economics
  Prentice-Hall PTR, 1991. ISBN 0138221227
- **Manager's Handbook for Software Development**
  NASA Software Engineering Laboratory, SEL-84-101, rev.1, 1990.
- **Software Engineering Laboratory (SEL)**
  Relationships, Models, and Management Rules
  NASA Software Engineering Laboratory, SEL-91-001, 1991.

Bibliografía

Bibliografía en castellano

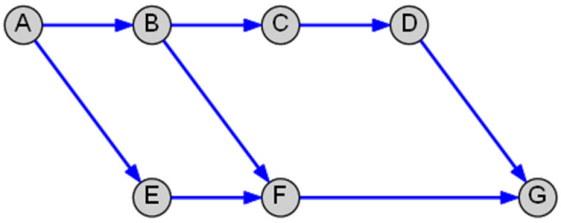
- Roger S. Pressman:
  Ingeniería de Software: Un enfoque práctico
  McGraw-Hill, 7ª edición, 2010. ISBN 6071503140
- Ian Sommerville:
  Ingeniería de Software
  Pearson, 9ª edición, 2012. ISBN 6073206038



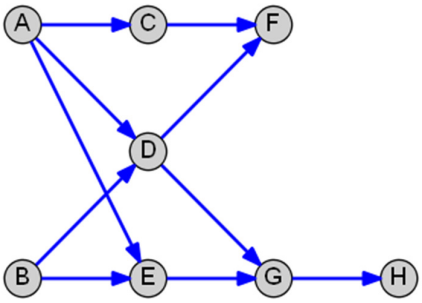


Ejercicios

Dadas las siguientes redes de tareas…



Tarea	Duración
A	6
B	4
C	5
D	3
E	2
F	7
G	1



Tarea	Duración
A	3
B	4
C	5
D	6
E	8
F	7
G	2
H	1

Ejercicios

... obtenga:

- La hora de comienzo más temprana (ES) para cada tarea que nos permite completar todas las tareas en un tiempo mínimo.
- La hora de comienzo más tardía (LS) para cada tarea que nos permite completar todas las tareas en un tiempo mínimo.
- La holgura [slack] de cada tarea.
- Los caminos críticos de cada proyecto.
- La duración del plan óptimo para cada proyecto.

Ejercicios

Escenarios alternativos:

- ¿Cuánto se alargaría la duración del proyecto si las tareas C y E requieren el uso exclusivo de un recurso específico? Asuma que nuestro presupuesto no nos permite adquirir varias unidades del recurso necesario.
- ¿Cuál sería la duración del proyecto si lo tenemos que hacer en solitario? En otras palabras, no podemos contratar a nadie para realizar tareas en paralelo.