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

Seguimiento y control de proyectos

Fernando Berzal, berzal@acm.org

Seguimiento y control de proyectos

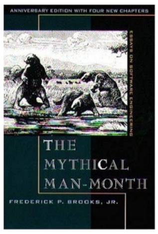
- Introducción
  - El rol del gestor de proyecto
  - El riesgo de la "microgestión"
  - Prácticas clave
  - Caso práctico: NASA Software Engineering Laboratory
- Seguimiento del proyecto
  - "Radiadores de información": Scrum & Kanban
  - EVM [Earned Value Management]
- Retrospectivas del proyecto
- Uso de métricas
- Uso de estándares

Seguimiento y control de proyectos

"The building metaphor has outlived its usefulness. It is time to change again. If, as I believe, the conceptual structures we construct today are too complicated to be accurately specified in advance, and too complex to be built faultlessly, then we must take a radically different approach. […] Let us turn to nature and study complexity in living things, instead of just the dead works of man. Here we find constructs whose complexities thrill us with awe.

The brain alone is intricate beyond mapping, powerful beyond imagination, rich in diversity, self-protecting, and self-renewing. The secret is that it is grown, not built. So it must be with our software systems."

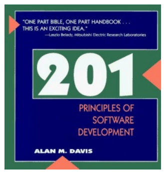
-- Frederick P. Brooks:
The Mythical Man-Month



Seguimiento y control de proyectos

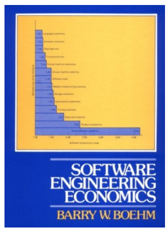
"Good management is more important than good technology."

-- **Alan M. Davis**
201 Principles of Software Development, 1995



"Poor management can increase software costs more rapidly than any other factor."

-- **Barry Boehm**,
Software Engineering Economics, 1981



Seguimiento y control de proyectos

El gestor es responsable de establecer un objetivo compartido para el equipo.

p.ej. Gestión por objetivos
[MBO: Management by objectives]

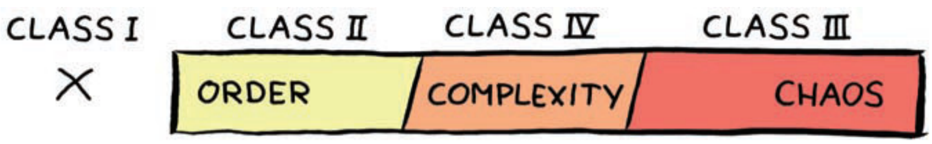
Existen distintos criterios para evaluar los objetivos.

p.ej. SMART
[specific, measurable, attainable, relevant & time-bound]

Seguimiento y control de proyectos

Clases de comportamiento

Stephen Wolfram: "Universality and Complexity in Cellular Automata"
Physica D, January 10:1-35, 1984



Si el gestor elige mal el conjunto de reglas, el sistema resultante será de tipo II (burocrático) o III (caótico), suponiendo que no acabe siendo de tipo I (muerto).

Idealmente, el gestor no diseña el juego, sino que establece las reglas para que el equipo pueda crecer (clase IV): gestiona el sistema, no a las personas.

El riesgo de la microgestión
[micromanagement]

"… a reliable way to bring an organization to its knees is for people to do exactly what the rules tell them to do and nothing else."
-- Ralph D. Stacey et al.:
Complexity and Management, 2000

- El exceso de reglas puede ser peligroso: disminuye la percepción del riesgo y crea una falsa seguridad.
- Eliminar reglas aumenta la percepción del riesgo (compensación de riesgo) y puede ayudar…

En la mayor parte de las ocasiones,
las reglas deberían usarse como **heurísticas**,
no como leyes:

- Las buenas reglas apuntan en una dirección que normalmente es buena (aunque no siempre lo sea).
- En ocasiones, es necesario abolir determinadas reglas para evitar que se sigan ciegamente.

Estrategia opuesta al principio de precaución (cuando algo puede ir mal, se establece una norma para prevenir que eso ocurra).

Principio de subsidiaridad

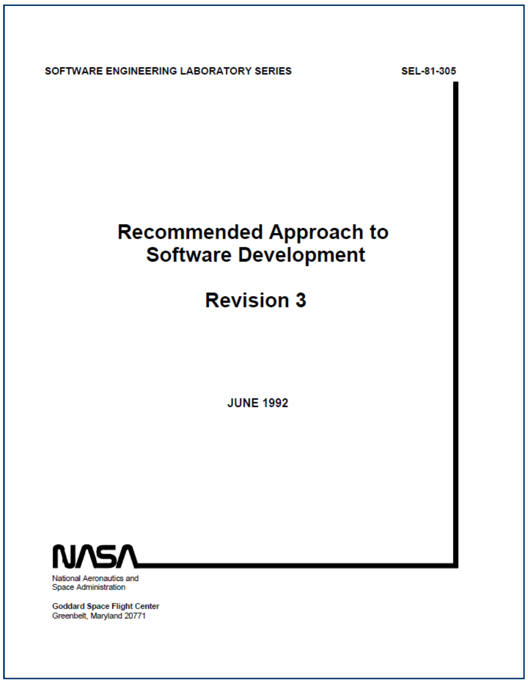
- Los asuntos deberían gestionarse por la autoridad menor o menos centralizada que sea competente.
- Una autoridad central debería tener una función subsidiaria, encargarse únicamente de aquellas tareas que no pueden realizarse de forma efectiva a un nivel más inmediato o local.

Algunas prácticas críticas

- Gestión de riesgos formal.
- Estimación empírica de costes (y planificación acorde).
- Uso de métricas.
- Seguimiento del valor generado por el proyecto.
- Seguimiento de defectos (objetivos de calidad).
- Factores humanos [peopleware].

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

Seguimiento y control de proyectos



SOFTWARE ENGINEERING LABORATORY SERIES SEL-81-305

Recommended Approach to
Software Development

Revision 3

JUNE 1992

NASA
National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

Seguimiento y control de proyectos

NASA Software Engineering Laboratory
Recommended Approach to Software Development

Para el éxito de un proyecto:

- Cree y siga un plan de desarrollo de software [SDP].
- Aproveche el potencial humano de su equipo.
- Minimice la burocracia.
- Establezca los requisitos y gestiona sus cambios.
- Evalúe periódicamente la salud y el progreso del proyecto (y replanifique cuando sea necesario).

Seguimiento y control de proyectos

NASA Software Engineering Laboratory

Recommended Approach to Software Development

Para el éxito de un proyecto:

- Revise las estimaciones de tamaño, esfuerzo y plan temporal periódicamente, sin insistir en mantener las estimaciones iniciales.
- Defina y gestione las transiciones de una fase a otra del proyecto de desarrollo de software.
- Fomente el espíritu de equipo.
- Comience el proyecto con un equipo pequeño formado por personal experimentado.

Seguimiento y control de proyectos

NASA Software Engineering Laboratory

Recommended Approach to Software Development

Para evitar el fracaso de un proyecto:

- No permita trabajar de forma no sistemática (aunque se trate de un trabajo creativo, existen principios y prácticas útiles cuyo uso es beneficioso).
- No establezca objetivos poco razonables.
- No implemente cambios sin evaluar su impacto y requerir su aprobación.
- No implemente lo que no sea necesario [gold-plating]: pequeños cambios que parecen mejorar el sistema tienden a aumentar su complejidad.

Seguimiento y control de proyectos

NASA Software Engineering Laboratory
Recommended Approach to Software Development

Para evitar el fracaso de un proyecto:

- No añada más personal del necesario, especialmente al comienzo del proyecto. Añádalo sólo cuando tenga trabajo útil que hacer para ellos.
- No asuma que incumplir un plazo intermedio se puede corregir más adelante.
- No relaje sus estándares para recortar costes o acortar plazos (tiende a introducir errores y desmotiva).
- No asuma que mucha documentación garantiza algo.

Seguimiento del proyecto

"The trouble with programmers is that you can never tell what a programmer is doing until it's too late."
— **Seymour Cray**

Seguimiento del proyecto

Nuestra forma de pensar lineal nos hace ver todo como un conjunto de eventos explicables con causas simples y efectos directos [causation fallacy]

Gerald M. Weinberg: *Quality Software Management*, 1992

El determinismo causal dio lugar a la "gestión científica" en el siglo XX [scientific management], que sirve para tareas repetitivas pero no para actividades creativas como el desarrollo de software.

"For every complex problem
there is an answer that is clear, simple, and wrong."
— **H.L. Mencken**

Seguimiento del proyecto

- El progreso de un proyecto se mide conforme se van completando tareas y produciendo resultados [work products]: modelos, código, casos de prueba…
- Los resultados del trabajo no se consideran completados hasta que se aprueban (p.ej. usando revisiones técnicas), como parte del proceso de QA.

Seguimiento del proyecto

Periódicamente (p.ej. semanalmente),
el gestor del proyecto debería

- Recopilar datos resumidos del proyecto (planificación, defectos, tiempo invertido…).
- Comparar esos datos con el plan (hitos completados, defectos identificados, esfuerzo realizado…)
- Revisar y actualizar la lista de riesgos del proyecto.
- Revisar los cambios propuestos y aprobados (y sus efectos acumulativos sobre el plan del proyecto).

Seguimiento del proyecto

Periódicamente (p.ej. semanalmente),
el gestor del proyecto debería

- Informar del estado del proyecto a todos sus "stakeholders" (clientes, sponsors, desarrolladores…), preferiblemente mediante mecanismos que mantengan la visibilidad del estado del proyecto.
- Tomar medidas correctivas si los resultados reales se desvían significativamente de los planes (o aparecen nuevos riesgos que deben tenerse en cuenta).

Seguimiento del proyecto

Al final de cada fase o iteración,
el gestor del proyecto debería

- Considerar solicitudes de cambios.
- Reestimar esfuerzo, plan temporal y coste.
- Mantener un log del proyecto [McConnell]:

TABLE 17-1 CONTENTS OF THE SOFTWARE PROJECT LOG
Current project estimates for schedule and effort
Adjustments to schedule and effort approved by the change board during the stage
Dates, background, and results of major decisions during the stage
Planned vs. actual dates for each of the stage's major deliverables
Results of technical reviews conducted during the stage (pass/fail status and defect statistics)
Time-accounting data
Lines of code
Defect count
Number of changes proposed and accepted

Seguimiento del proyecto

¿Cómo se consigue que las personas adopten buenas prácticas?

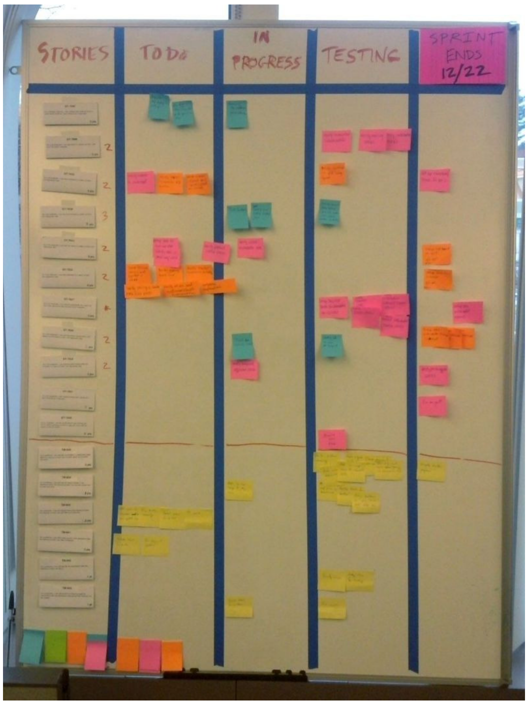
Muy fácil: Haciéndolas visibles.

La visibilidad de un proceso que funciona hace que la gente se dé cuenta de ello y quiera utilizarlo. Además, facilita el trabajo del gestor del proyecto.

Cualquier proceso visible
es un **"radiador de información"**.

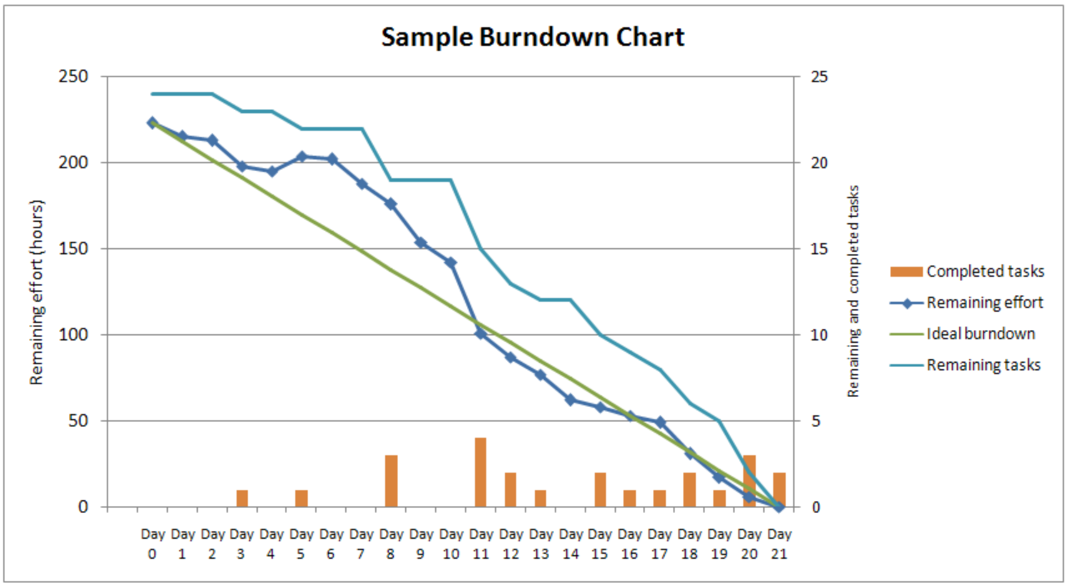
Seguimiento del proyecto

Scrum: Task board



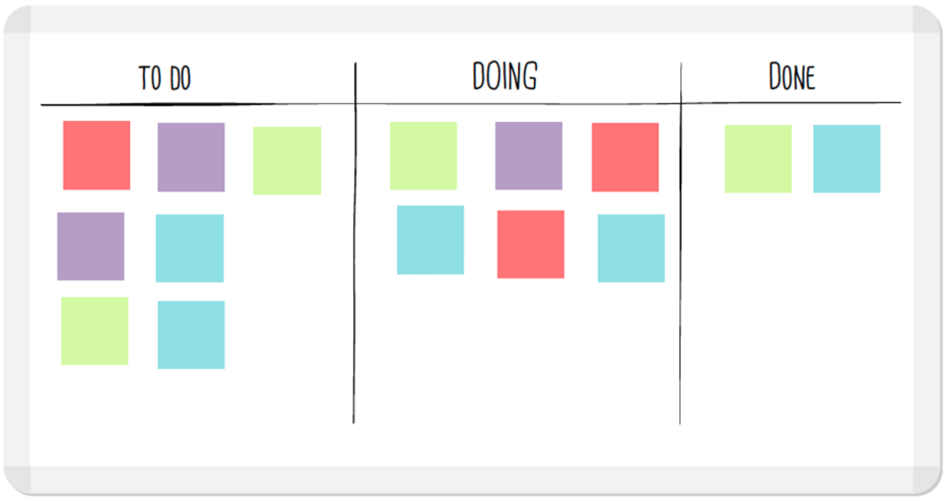
Seguimiento del proyecto

Scrum: Sprint burn-down chart



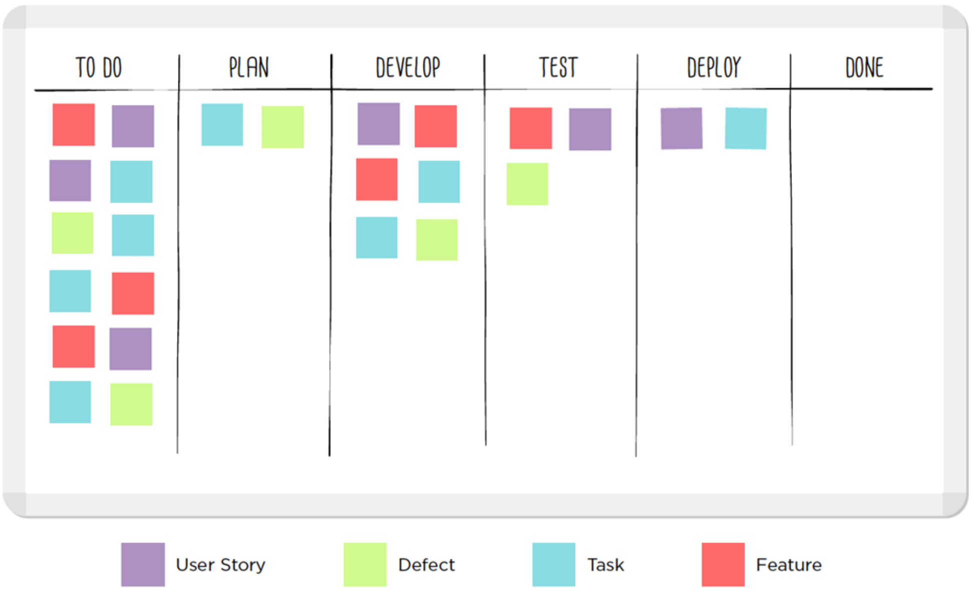
Seguimiento del proyecto

Kanban boards



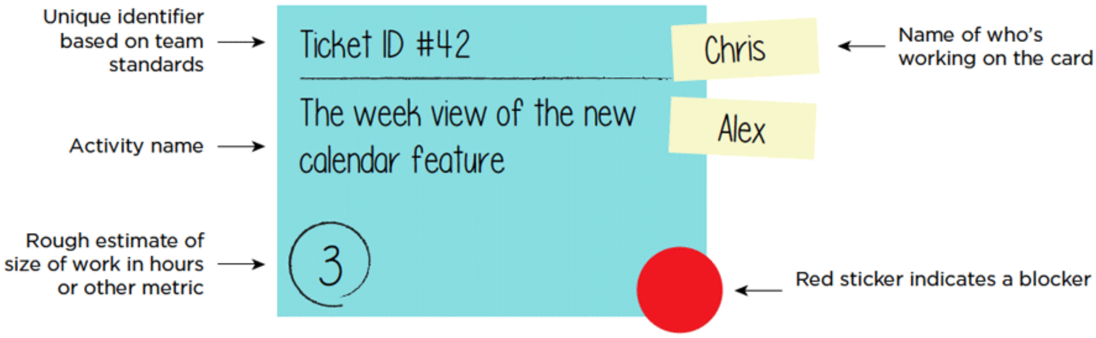
Seguimiento del proyecto

Kanban boards



Seguimiento del proyecto

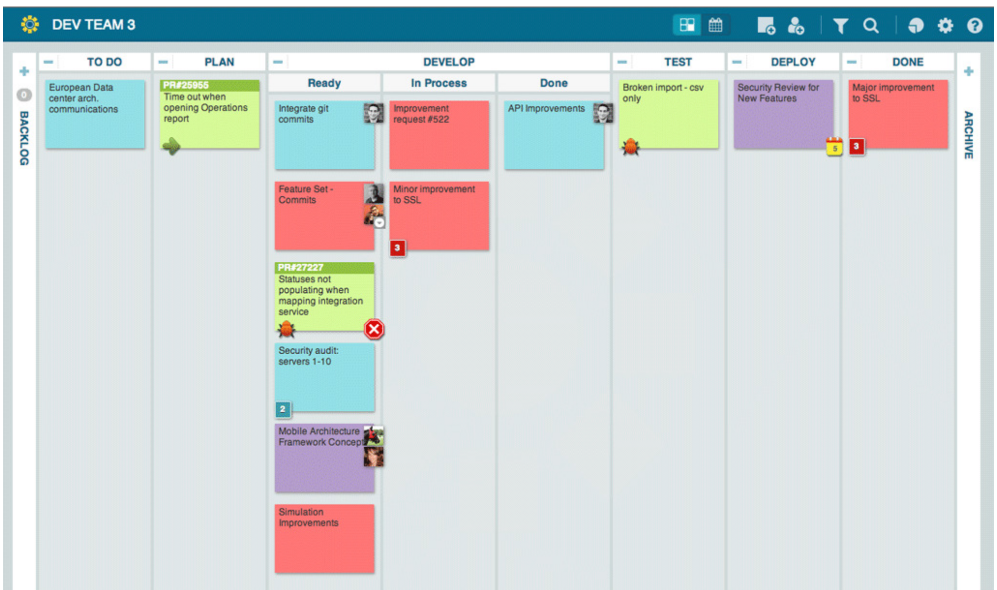
Kanban boards



Seguimiento del proyecto

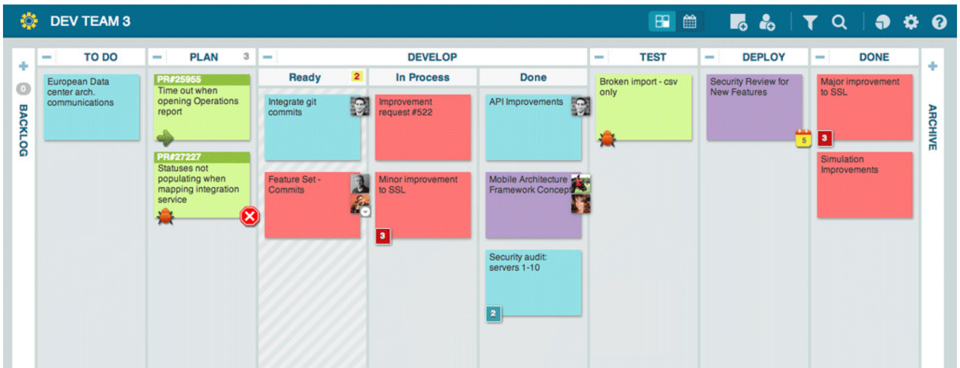
Kanban boards

leankit



Seguimiento del proyecto

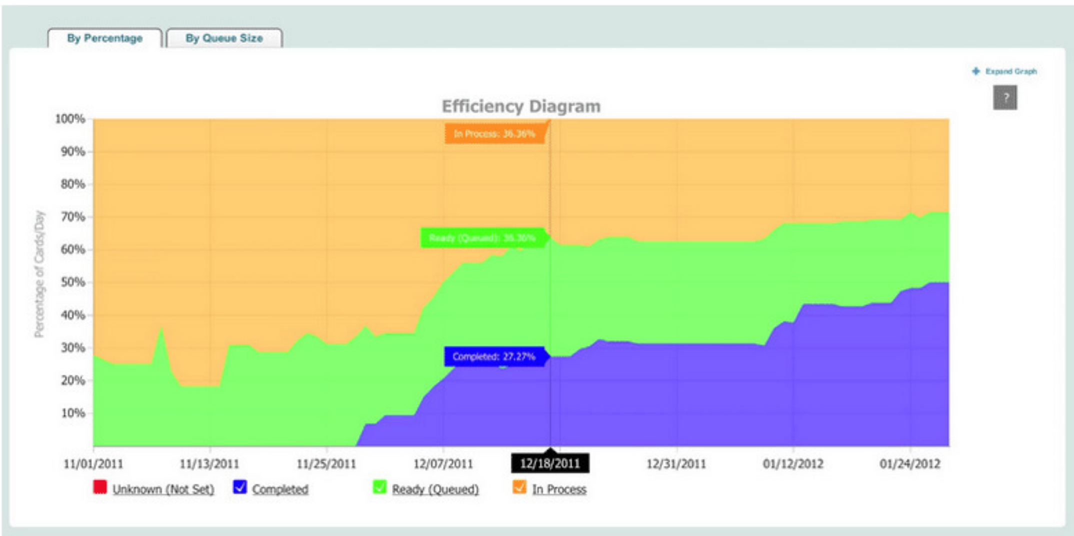
Kanban boards



WIP [Work in progress] limit

Seguimiento del proyecto

Kanban boards



Seguimiento del proyecto

Earned Value Management [EVM]

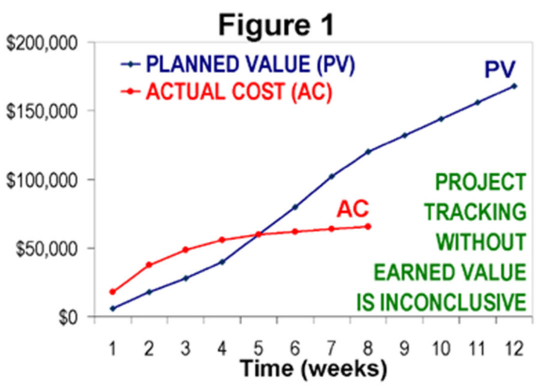
"Earned Value" [EV] o "valor ganado"
es una medida que permite evaluar
cuantitativamente el progreso de un proyecto.

If you don't have time to calculate value,
we don't have time to calculate cost.
— Jim Highsmith

Q.W. Fleming & J. M. Koppelman,
"Earned Value Project Management,"
CrossTalk, vol. 11, no. 7, p. 19, July 1998.

Seguimiento del proyecto

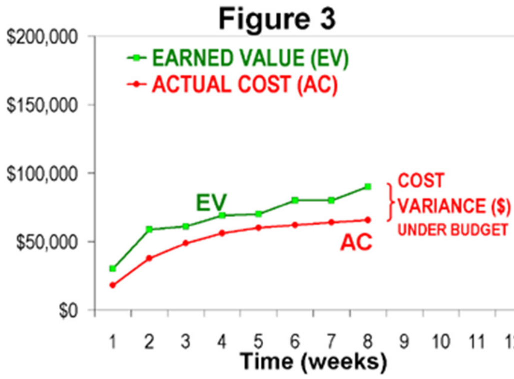
Earned Value Management [EVM]



Presupuesto vs. coste real acumulado
(inútil si no se cuantifica el valor del trabajo realizado)

Seguimiento del proyecto

Earned Value Management [EVM]

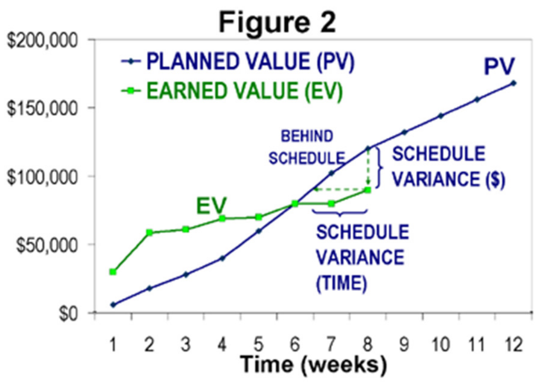


Valor vs. coste:

El coste del proyecto se mantiene por debajo del valor que proporciona durante todo el proyecto.

Seguimiento del proyecto

Earned Value Management [EVM]



Se comenzó creando valor real (EV)
por encima de lo previsto (PV)
pero el proyecto se retrasó en las semanas 7 y 8.

Seguimiento del proyecto

Earned Value Management [EVM]

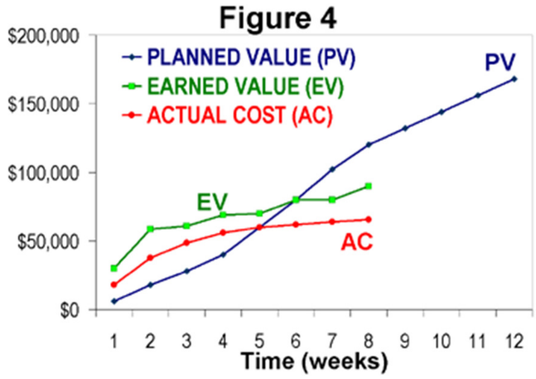
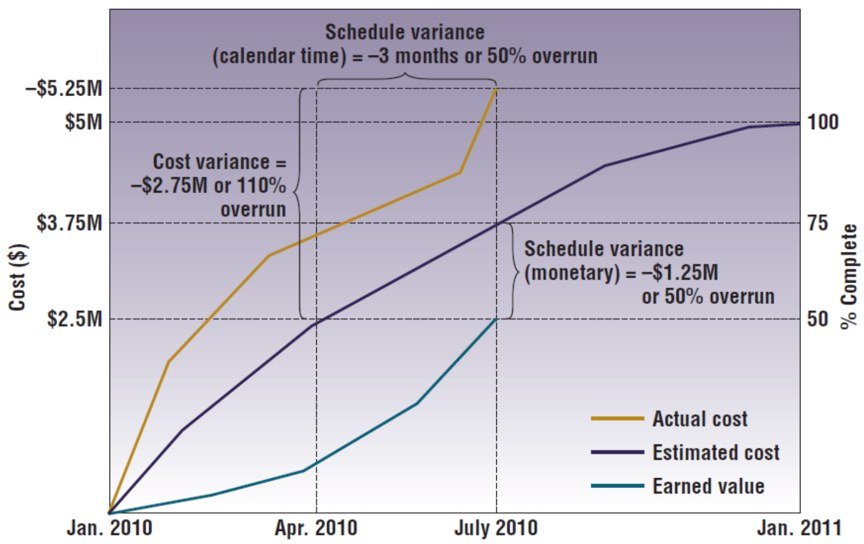


Gráfico EVM combinado

Seguimiento del proyecto

Earned Value Management [EVM]



Hakan Erdogmus: "Tracking Progress through Earned Value", IEEE Software 27(5):2-7, September/October 2010.

Seguimiento del proyecto

Earned Value Management [EVM]

Plan previsto: PV=BCWS

- **BCWS$_i$** [Budgeted Cost of Work Scheduled]:
  Coste previsto para cada tarea del plan.
- **BCWS** (progreso del proyecto previsto en plan):
  Suma de los BCWS de las tareas que deberían haberse completado en cada momento.
- **BAC** [Budget at Completion]:
  Suma de los BCWS de todas las tareas del proyecto

Seguimiento del proyecto

Earned Value Management [EVM]

Ejecución del plan: EV=BCWP

- **BCWP** [Budgeted Cost of Work Performed]:
  Suma de los BCWS de las tareas que se han completado realmente en cada momento.
- **SPI** [Schedule Performance Index] = BCWP/BCWS
  Eficiencia con la que el proyecto está utilizando los recursos previstos en el plan.
- **SV** [Schedule Variance] = BCWP - BCWS

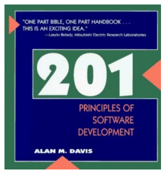
Seguimiento del proyecto

Earned Value Management [EVM]
Ejecución del plan

- **Porcentaje de trabajo completado** = BCWP / BAC
- **ACWP** [Actual Cost of Work Performed]
  Suma del esfuerzo realmente realizado para completar las tareas realizadas
- **CPI** [Cost Performance Index] = BCWP/ACWP
- **CV** [Cost Variance] = BCWP - ACWP

Seguimiento del proyecto

Earned Value Management [EVM]
Medidas relevantes de progreso

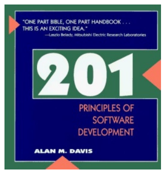


Cuánto se ha completado:
- BCWP (lo que se esperaba gastar).
- ACWP (lo que se ha gastado realmente).

Cuánto se ha presupuestado:
- BCWS

Seguimiento del proyecto

Earned Value Management [EVM]
Medidas relevantes de progreso



Estado actual del proyecto:

- (BCWP-BCWS)/BCWE = **% sobre el plan previsto**
  - >0 por delante del plan previsto
  - <0 retrasados con respecto al plan previsto
- (BCWP-ACWP)/BCWP = **% sobre el presupuesto**
  - >0 por debajo del presupuesto
  - <0 por encima del presupuesto

Retrospectivas del proyecto

The Second Law of Consulting
No matter how it looks at first,
it's always a people problem.
-- Gerald M. Weinberg: “The Secrets of Consulting”, 1985

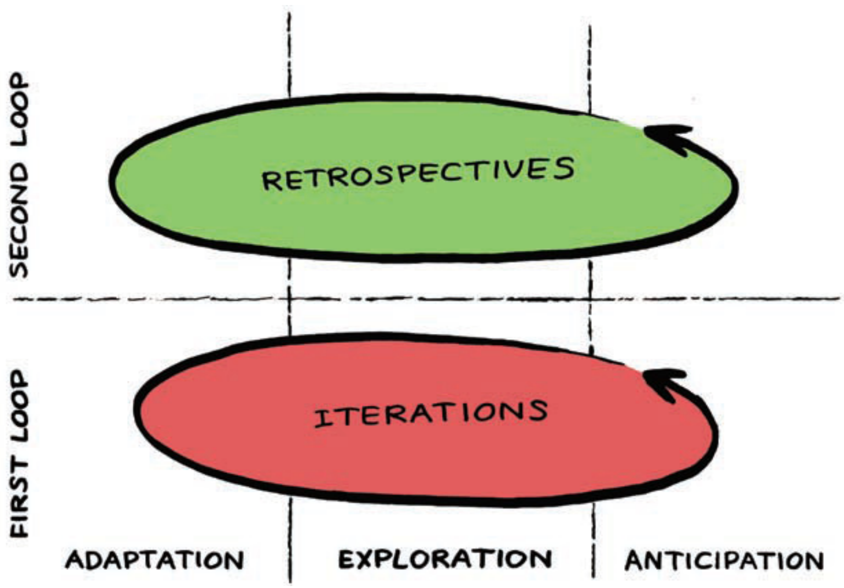
Retrospectivas del proyecto

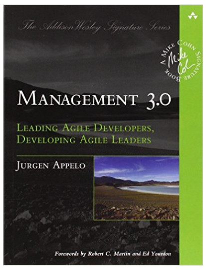
Everyone sits in the prison of his own ideas;
he must burst it open.
-- Albert Einstein

- Necesarias para la introducción de medidas de mejora continua [continuous improvement]: los procesos y prácticas utilizados se evalúan y ajustan reflexionando sobre lo acontecido durante la ejecución del proyecto.

Retrospectivas del proyecto

"Double-loop learning"





Retrospectivas del proyecto

"Aquéllos que no recuerdan el pasado
están condenados a revivirlo."
-- George Santayana, 1908

Se debe establecer un mecanismo consistente mediante el que realizar un **análisis postmortem** del proyecto (o de una iteración del proyecto):

- Extraer lecciones de lo sucedido [lessons learned].
- Establecer mecanismos de mejora de cara al futuro.

Retrospectivas del proyecto

- What was the plan at the start of the project? How did the plan change?
- What do you know now at the end of the project that you'd wished you had known earlier? How would this have changed the project?
- What most noticeably went right on the project? Why did they go right?
- What most noticeably went wrong on the project? Why did they go wrong?
- What sorts of information came in too late? What could have been done to get this information in more on time?
- In what ways did the project backtrack or rework ground already covered? How could this rework have been avoided?
- Did the project team have enough expertise or training at the start of the project? by the end of the project? What skills or knowledge turned out to be most important?
- How effective was the development method or approach used in the project? In what ways did it succeed? In what ways did it fail? Would you choose to follow that approach again?
- What tools were used on the project? For what were they most effective? For what were they least effective? How would you change the use of these tools to make them more effective?
- What are the most important things you would point out to your manager or your staff if you joined a similar project in the future?

E. Chikofsky: "Changing your endgame strategy"
IEEE Software 7(6):87,112, November 1990

Retrospectivas del proyecto

Software Project History [McConnell]

Introduction

Describe the software's purpose, customer, vision statement, detailed objectives, and other general information.

Historical Overview

For each phase, describe the work products produced, ***milestones****, major risks addressed, schedules,* ***staffing*** *levels, and other project planning information.*

Describe the following phases:

User-interface prototyping and requirements gathering

Architectural design

Quality assurance planning

General stage planning

Activities from detailed design through release (including detailed design, construction, system testing, and stage releases) for each stage 1 – *n*

Final software release

Retrospectivas del proyecto

Software Project History [McConnell]

Project Data

Describe the organizational structure used, including the executive sponsor, project ***participants****, their roles, and their levels of participation over the course of the project.*

The Software Project ***History*** *should also contain the following hard data about the project:*

Actual schedule and effort as of the release date

Time-accounting data as of the release date

Number of subsystems as of the release date

Lines of source code as of the release date

Lines of reused code as of the release date

Amount of media (sound, graphics, video, and so on)

Defect count as of the release date

Number of changes proposed and accepted as of the release date

Graph showing each schedule estimate compared to the actual schedule over time

Graph showing each effort estimate compared to the actual effort over time

Graph of project's code growth by week

Graph of project's open and closed defect count by week

Retrospectivas del proyecto

Software Project History [McConnell]

Lessons Learned

Describe the lessons learned on the project.

Planning. Were the plans useful? Did the project team adhere to the plans? Was the quality of the project personnel sufficient? Was the number of personnel in each category sufficient?

Requirements. Were the requirements complete? Were they stable or were there many changes? Were they easy to understand, or were they misinterpreted?

Development. How did the design, coding, and unit testing work out? How did the daily build work? How did software integration work? How did the releases work?

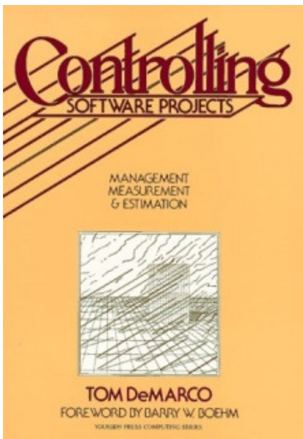
Testing. How did the test planning, test case development, and smoke test development work? How did automated testing work?

New Technology. What impacts did new technology have on costs, schedules, and quality? Did managers and developers interpret these impacts the same way?

Uso de métricas

"You can't control what you can't measure"

-- Tom DeMarco, Controlling Software Projects, 1982



Uso de métricas

WYMIWYG = What You Measure Is What You Get

"To measure is to know"
-- James Clark Maxwell

"What gets measured get managed"
-- Peter Drucker

"Invisible targets are usually hard to hit"
-- Tom Gilb

Uso de métricas

Uso de métricas

"One great irony inherent in the management of software projects is that despite the digital precision of the materials programmers work with, the enterprise of writing software is uniquely resistant to measurement."
— Scott Rosenberg: "Dreaming in Code"

"If you torture the data long enough, it will confess"
— Ronald Coase

Uso de métricas

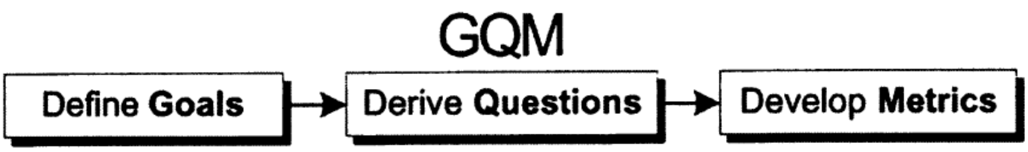
Ley de Gilb

"Anything you need to quantify can be measured
in some way that is superior to not measuring it at all."

-- Tom Gilb: "Software Metrics,"
Winthrop Publishers, 1977
ISBN 0876268556

GQM [Goal-Question-Metric]

Victor R. Basili



Uso de métricas: PSP

PSP [Personal Software Process]

Enfoque disciplinado basado en el uso de métricas:

- Personalmente, se recopilan métricas del resultado de nuestro trabajo y de su calidad.
- Se analizan los tipos de errores que se cometen, de forma que se puedan desarrollar estrategias para eliminarlos en el futuro.

Uso de métricas: PSP

PSP [Personal Software Process]

5 actividades:

- **Planificación** (estimaciones de tamaño, recursos necesarios y número esperado de defectos).
- **Diseño.**
- **Revisión del diseño** (uso de métodos de verificación formal para descubrir errores en el diseño).
- **Desarrollo** (pruebas incluidas).
- **Postmortem** (análisis estadístico de las métricas, con el objetivo de modificar el proceso para mejorar su efectividad).

Uso de métricas: SEL

Proyectos

Characteristics	Low	Average	High
Duration			
Time (months)	19	24	43
Cost			
Effort (staff-years)	3	14	32
Size			
Delivered code (KSLOC)	31	107	246
Staff			
Average staff (FTE)	4	8	15
Peak staff (FTE)	5	13	30
Individuals (total)	6	22	44
Application Experience			
Managers (years)	4	9	15
Technical staff (years)	2	4	7
Overall Experience			
Managers (years)	10	14	19
Technical staff (years)	4	6	9

Productividad

Characteristics	Nominal Values
Productivity	
Coding rate	3.3 developed SLOC per hour
Reuse percentage	30% of code is "reused"
Reliability	
Error rate	4 errors per developed KSLOC during implementation
Error rate	2 errors per developed KSLOC during system testing
Error rate	1 error per developed KSLOC during acceptance testing
Error rate	0.5 errors per developed KSLOC during maintenance/ operations
Change rate	14 changes to components per developed KSLOC during implementation
Maintainability	
Effort to change	85% of all changes classified as "easy" or "very easy"
Effort to repair	85% of all repairs classified as "easy" or "very easy"

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

Uso de métricas: SEL

Esfuerzo

Staffing Category	% of Total Effort
Programmers	84
Managers	10
Support Staff	6

Reported Activity	% of Total Hours	
	FORTRAN	Ada
Design	23	19
Code	21	16
Test	30	35
Other	26	30

Reutilización

Module Classification	Percent of Code Modified or Added	Relative Cost
New	100	100
Extensively Modified	>25	100
Slightly Modified	1-25	20
Old	0	20

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

Uso de métricas: SEL

Esfuerzo vs. Líneas de código

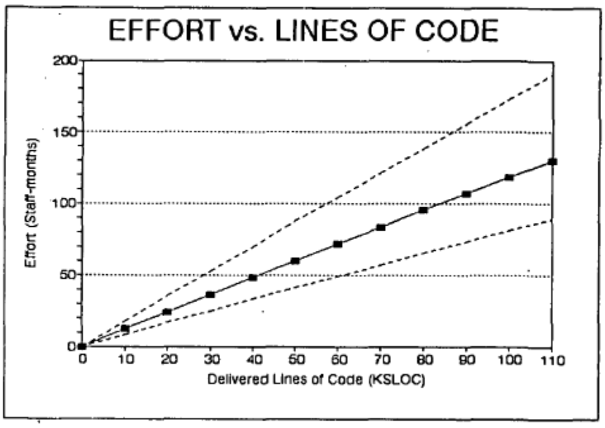
$E = 1.266 * L^{0.986}$

where

E is total effort in staff-months (with 172 hours per staff-month)

L is total delivered lines of code in KSLOC

<u>NOTE</u>: uncertainty = 0.456
$E_{upper\ bound} = E * (1.0 + uncertainty)$
$E_{lower\ bound} = E / (1.0 + uncertainty)$



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

Uso de métricas: SEL

Duración vs. Líneas de código

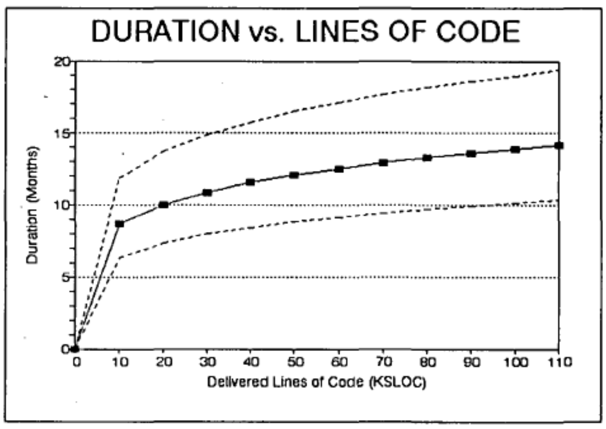
$D = 5.450 * L^{0.203}$

where

D is project duration in months (from start of project to end of acceptance testing)

L is total delivered lines of code in KSLOC

<u>NOTE</u>: uncertainty = 0.367
$D_{upper\ bound} = D * (1.0 + uncertainty)$
$D_{lower\ bound} = D / (1.0 + uncertainty)$



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

Uso de métricas: SEL

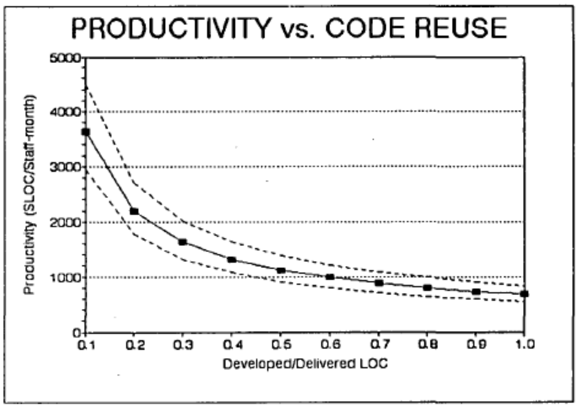
Productividad vs. Reutilización

$P = 678.311 * (DL/L)^{-0.730}$

where

- **P** is productivity in delivered SLOC per staff-month (defined as total delivered SLOC divided by total effort, or L/E, with 172 hours per staff-month)
- **DL** is total developed lines of code in KSLOC (excludes reused code)
- **L** is total delivered lines of code in KSLOC

NOTE: uncertainty = 0.235
$P_{upper\ bound} = P * (1.0 + \text{uncertainty})$
$P_{lower\ bound} = P / (1.0 + \text{uncertainty})$



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

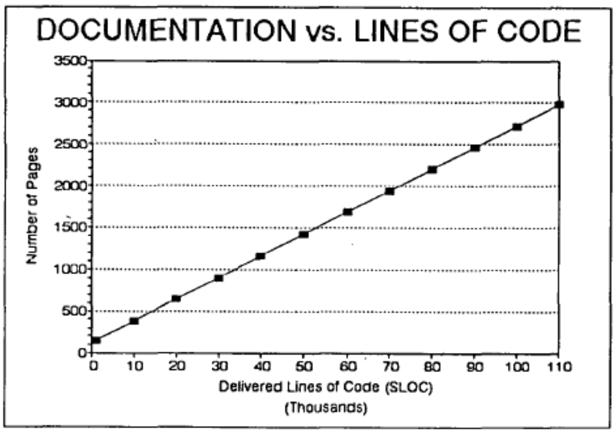
Uso de métricas: SEL

Páginas de documentación vs. Líneas de código

$P = 120 + 0.026 * L$

where

- **P** is total pages of documentation (includes the requirements analysis report, design documents, system description, and user's guide)
- **L** is total delivered lines of code in SLOC



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

Uso de métricas: SEL

Tamaño, esfuerzo y duración por fases

End of Phase	Relationship	Uncertainty
Requirements Analysis	$L = 11000 * SS$ $E = 3000 * SS$ $D = 83 * SS$	0.75
Preliminary Design	$L = 190 * M$ $E = 52 * M$ $D = 1.45 * M$	0.40
Detailed Design	$DL = 200 * DM$ $E = 0.31 * DL$ $D = 0.0087 * DL$	0.25
Implementation	$L = 1.26 * L_{Cur}$ $E = 1.43 * E_{Cur}$ $D = 1.54 * D_{Cur}$	0.10
System Testing	$E = 1.11 * E_{Cur}$ $D = 1.18 * D_{Cur}$	0.05

where

- L is total size in SLOC
- E is total effort in staff-hours
- D is total duration in weeks per staff member (based on a full-time employee's average work week with 1864 hours annually)
- SS is total number of subsystems in project
- M is total number of modules in project
- DM is total number of developed modules in project (defined as N + 0.2*R, with N being new modules and R being reused modules)
- DL is developed lines of code in SLOC
- L_{Cur} is size in SLOC through the current phase
- E_{Cur} is effort expended in staff-hours through the current phase
- D_{Cur} is schedule expended in calendar weeks through the current phase

NOTE: Uncertainty applies to effort or size estimates as follows:
$Estimate_{upper\ bound} = Estimate * (1.0 + uncertainty)$
$Estimate_{lower\ bound} = Estimate\ /\ (1.0 + uncertainty)$

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

Uso de métricas: SEL

Ajuste de las estimaciones

Complejidad del proyecto

Project	Environment	Multiplier
OLD	OLD	1.0
OLD	NEW	1.4
NEW	OLD	1.4
NEW	NEW	2.3

where

- Project Type is considered OLD when the development team has more than 2 years experience with the application area (e.g., orbit determination, simulator)
- Environment Type is considered OLD when the development team has more than 2 years experience with the computing environment (e.g., IBM 4341, VAX 8810)

Experiencia del equipo

Team Years of Application Experience	Effort Multiplier
10	0.5
8	0.6
6	0.8
4	1.0
2	1.4
1	2.6

where

Team Years is the average of all team member's years of application experience weighted by the member's participation on the team

with

Application experience is defined as prior work on similar applications (e.g., attitude and orbit determination).

Member's participation is defined as time spent working on the project as a proportion of total project effort.

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

Uso de métricas: SEL

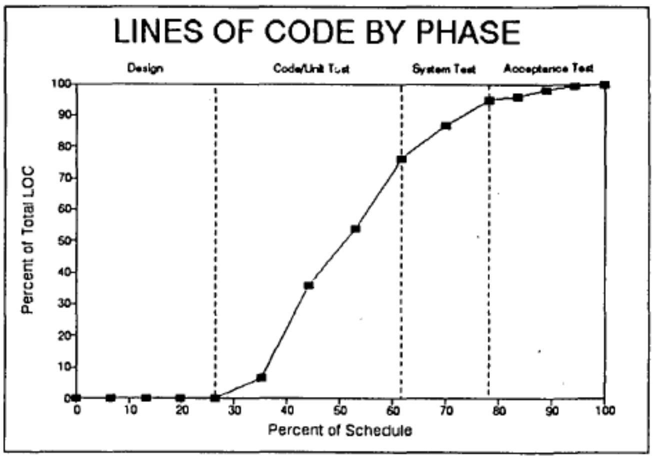
El ciclo de vida de un proyecto

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.

Uso de métricas: SEL

Líneas de código y errores esperados

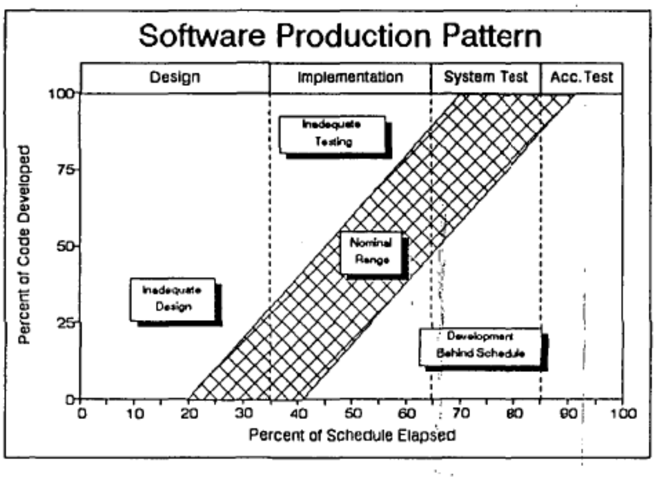


Phase	Errors Detected Per Developed KSLOC
Code/Unit Test	4
System Test	2
Acceptance Test	1
Maintenance/Operations	0.5

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

Uso de métricas: SEL

Desviaciones en las métricas: Reglas de gestión



If
the phase is design and lines of code is high
then
there is inadequate design effort being expended

If
the phase is implementation and lines of code is high
then
there is inadequate unit testing being done

If
the phase is implementation or test and lines of code is low
then
the development effort is behind schedule

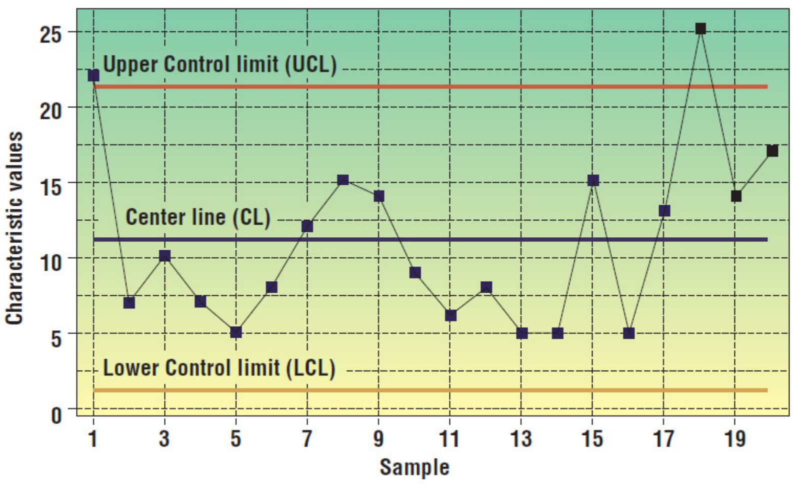
Software Engineering Laboratory (SEL)
Relationships, Models, and Management Rules
NASA Software Engineering Laboratory, SEL-91-001, 1991.

Uso de métricas

Gráficos de control

p.ej. c-chart

$$CL = \bar{C}$$
$$UCL = \bar{C} + 3\sigma$$
$$LCL = \bar{C} - 3\sigma$$

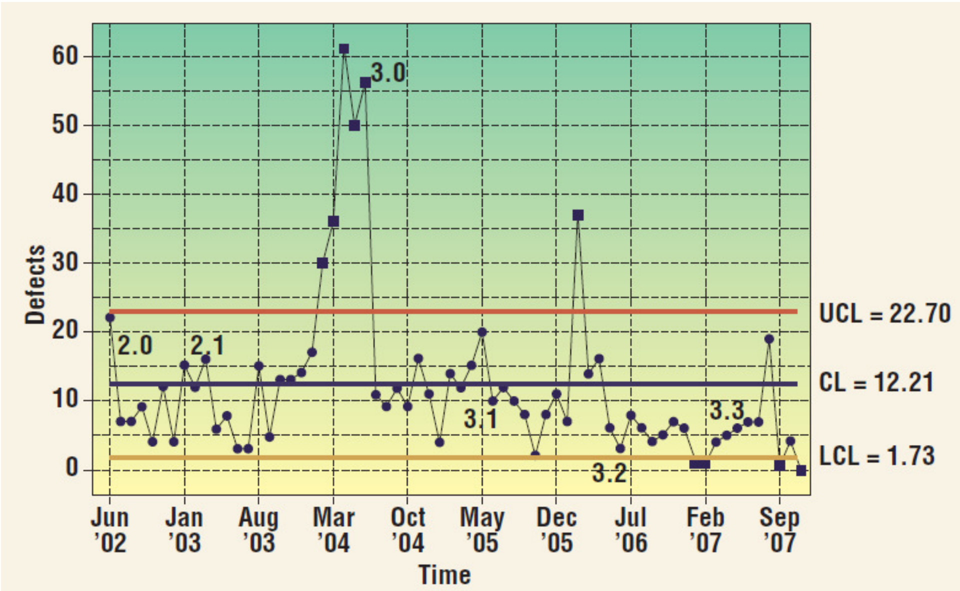


Hongyu Zhang & Sunghun Kim:
Monitoring Software Quality Evolution for Defects
IEEE Software 27(4):58-64, July/August 2010.

Uso de métricas

Gráficos de control

Eclipse Search Component



Uso de métricas

"Behavior revolves around what you measure."
— **Jim Highsmith**, BayALN, February 16, 2010

Todas las métricas son "disfuncionales" y tienden a causar lo contrario de lo que se pretende, por lo que es una buena práctica utilizar una amplia gama de métricas.

¡Cuidado con su uso para evaluar la productividad!

"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs." — **Bill Gates**

Uso de métricas

Principio de suboptimización

L. Skyttner: **General Systems Theory: Ideas and Applications**, 2001.

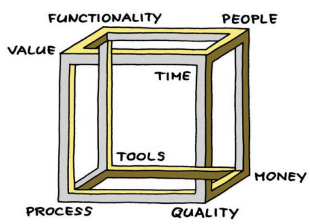
Si cada subsistema, considerado por separado, se hace operar con su máxima eficiencia, el sistema en su conjunto no operará de la forma más eficiente.

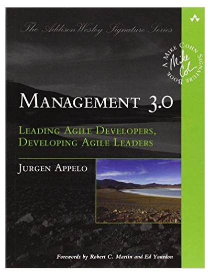
- Hay que optimizar el conjunto (p.ej. Lean Software Development).
- Hay que combinar métricas desde múltiples perspectivas que cubran el sistema completo (p.ej. Balanced Scorecard).

Uso de métricas

Dimensiones del proyecto y métricas

Dimensión	Métrica
Funcionalidad	Casos de uso completados (velocidad)
Calidad	Problemas identificados en las pruebas
Herramientas	Coste mensual
Personal	Obstáculos encontrados por los miembros del equipo de desarrollo
Tiempo	Días que quedan hasta la próxima entrega [release]
Proceso	Listas de comprobación completadas
Valor	Incremento en el uso del sistema

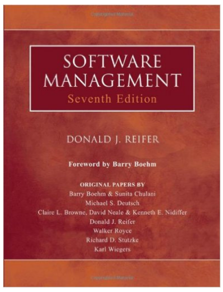




Uso de métricas

Métricas habituales

Category	Examples	How defined?	Questions answered
Project performance	Budget performance	Actuals versus targets	Is our rate of expenditure in line with the budget?
	Schedule performance	Rate of progress of milestone achievements	Is our rate of progress as planned?
	Earned value performance	Value earned for milestones achieved versus budget set for work to be performed and actual expenditures	Is the progress we are making commensurate with our rate of expenditure and our budget?
	Technical performance	Indicators like: • Requirements growth • Size growth	Are we making suitable progress relative to the indicators? For example, have our requirements stabilized or are they volatile? Is growth under control?
Process performance	Rework rate	Number of times it takes you to get it right	Is the process working the first time through?
	Defect rates	Number of defects discovered and removed as a function of time	Are defects being detected versus fixed at anticipated rates?
Product quality	Product complexity	Cyclomatic number or some similar metric	Is the product overly complex?
	Defect density	Number of defects as a function of size	Is the defect density as expected?
Personnel performance	Personal productivity	Individual output as function of inputs used to generate them	Are staff members generating products at anticipated rates?
Organizational performance	Process maturity	SEI rating using either the discrete or continuous models of the CMMI	Are projects using organizational processes?
	Product quality	Complaint rate	Are customers happy with the product?
	Productivity	Group output as function of inputs used to generate them	Are teams producing at anticipated rates?
Enterprise performance	Profitability	Price/earnings ratio	Is the enterprise profitable?
	Return on equity	Earnings as a function of capital used to generate it	Are investments generating acceptable earnings?
	Cost of sales	Dollars spent on sales as a function of revenue earned	Is the cost of sales acceptable?
	Competitiveness	Productivity versus benchmarks or competitive figures	Are we as productive as our competition?



Donald J. Reifer: "Metrics and Management: A Primer", 2006

Uso de métricas

Métricas más frecuentes

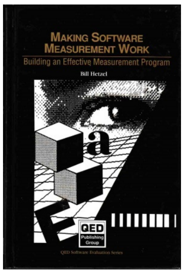
Software Metrics	% reported using
Number of defects found after release	61
Number of changes or change requests	55
User or customer satisfaction	52
Number of defects found during development	50
Documentation completeness/accuracy	42
Time to identify/correct defects	40
Defect distribution by type/class	37
Error by major function/feature	32
Test coverage of specifications	31
Test coverage of code	31

Bill Hetzel: "Making Software Measurement Work", 2003

Uso de métricas

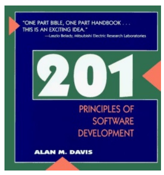
Métricas menos frecuentes

Software Metrics	% reported using
Module/design complexity	24
Number of source lines delivered	22
Documentation size/complexity	20
Number of reused source lines	16
Number of function points	10



Bill Hetzel: "Making Software Measurement Work", 2003

Uso de métricas



- Si se da prioridad a un factor, se optimiza ese factor y los otros tienden a olvidarse.
  G. Weinberg & E. Schulman: "Goals and Performance in Computer Programming", Human Factors 16:70-77, 1974

- Si se dice que todo es importante, nada se optimiza.

- La recolección de datos debe automatizarse (si interfiere con las tareas medidas o requiere la colaboración consciente y voluntaria del desarrollador, los datos pierden gran parte de su valor)
  S. Pfleeger: "Lessons Learned in Building a Corporate Metrics Program", IEEE Software 10(3):67-74, May 1993

Uso de estándares

"In the absence of meaningful standards, a new industry like software comes to depend instead on folklore."

— **Tom DeMarco**

"It's OK not to follow standards provided
(1) you know why, and
(2) you can articulate it."

— **Robert Marshall**, VP, Schwab.com

Uso de estándares

MITO

Existen estándares que ya establecen los procedimientos que hay que seguir en todo momento.

REALIDAD

Puede que el estándar necesario exista, pero

- ¿se usa?
- ¿es consciente el equipo de su existencia?
- ¿es completo?
- ¿se puede adaptar adecuadamente?

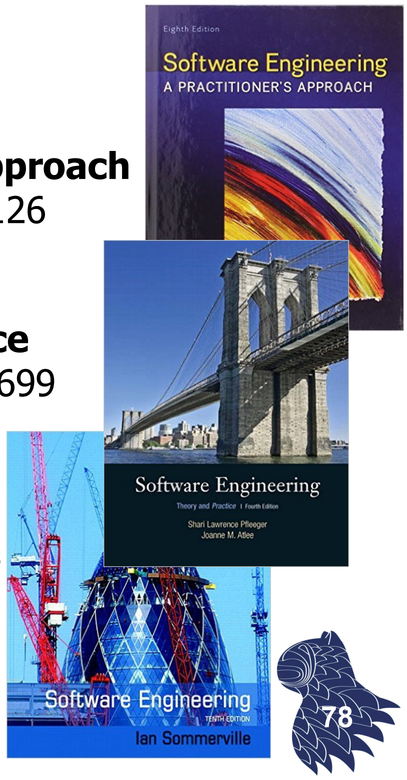
Bibliografía

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



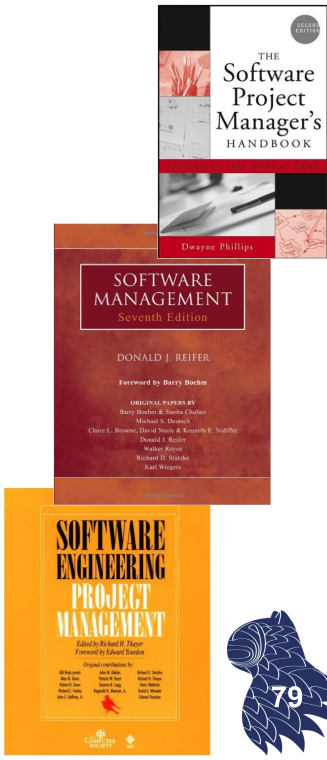
Bibliografía

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

Métricas

- Lawrence H. Putnam & Ware Myers:
  Five Core Metrics:
  The intelligence behind successful software management
  Dorset House, 2003. ISBN 0932633552

- Norman E. Fenton & Shari Lawrence Pfleeger:
  Software Metrics:
  A Rigorous and Practical Approach, Revised
  Course Technology, 1998. ISBN 0534954251

- Tom DeMarco:
  Controlling Software Projects:
  Management, measurement & estimation
  Prentice-Hall PTR, 1982. ISBN 0131717111

Bibliografía complementaria

Métricas

- **Software Measurement Guidebook**
  NASA Software Engineering Laboratory, SEL-94-102, rev.1, 1995

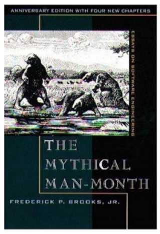
- Lawrence H. Putnam & Ware Myers
  Measures for Excellence:
  Reliable software on time, within budget
  Yourdon Press, 1992. ISBN 0135676940

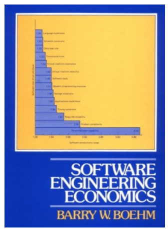
- Robert B. Grady & Deborah L. Caswell:
  Software Metrics: Establishing a Company-Wide Program
  Prentice Hall PTR, 1987. ISBN 0138218447

- Capers Jones:
  Programming Productivity
  McGraw-Hill, 1986, ISBN 0070328110

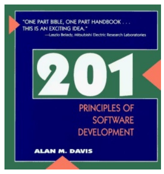
Bibliografía complementaria

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

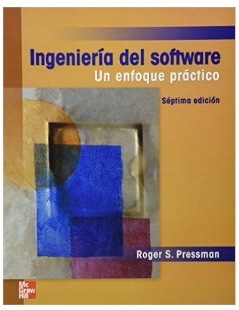


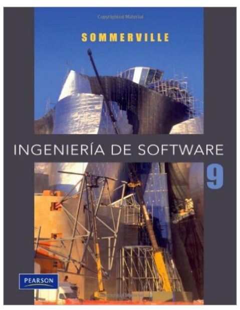
- Barry W. Boehm:
  Software Engineering Economics
  Prentice-Hall PTR, 1991. ISBN 0138221227
- **Manager's Handbook for Software Development**
  NASA Software Engineering Laboratory, SEL-84-101, rev.1, 1990.
- **Software Engineering Laboratory (SEL)**
  Relationships, Models, and Management Rules
  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

- Seleccionar un conjunto de métricas adecuadas para las 7 dimensiones de un proyecto (funcionalidad, calidad, herramientas, personal, tiempo, proceso, valor) y evaluar sus características (p.ej. SMART).

- Analizar los aspectos positivos que puede tener la presión social dentro del equipo de un proyecto.