



Compendio de Ingeniería del Software

Rev. 1.0 Mayo 2014

Juan Palacio

Tabla de contenido

- Prólogo
- Introducción a la ingeniería del software
- Gestión de proyectos predictiva
- Producción basada en procesos
- Ingeniería de procesos de software
- Modelo de procesos CMMI
- Modelo de procesos ISO IEC 15504
- Ciclo de vida del software
- Ciclos de desarrollo y patrones de gestión de proyectos
- Los requisitos del software
- Diseño del software
- Documentación de usuario
- Verificación y validación
- Mantenimiento
- Gestión de la configuración
- Agilidad
- Conocimiento
- Síntesis entre procesos y agilidad
- Scrum: el marco
- Scrum: Métricas
- Scrum: prácticas
- Kanban
- Gestión en organizaciones ágiles

Prólogo

Este *Compendio de Ingeniería del Software* recopila apuntes preparados por el autor para formación, que resumen aspectos pragmáticos de Ingeniería del Software.

Se comparten con una licencia abierta¹ para su uso en autoformación, o en foros que requieran la exposición de alguna de las áreas de la Ingeniería del Software que cubren, con un enfoque ejecutivo o general, sin la densidad propia del texto académico:

- Formación de Ingeniería del Software como asignatura complementaria en programas de estudio técnicos.
- Formación continua de gestores intermedios o directivos de empresas de software.
- Presentaciones de asesoría y formación profesional durante la implantación de procesos de mejora.
- Etc.

Este no es un trabajo completo, y por su carácter general no pretende cubrir todos los modelos, técnicas o líneas de trabajo de la Ingeniería del Software, sino las más pragmáticas para la industria del software.

Si resulta posible, en futuras versiones, además de la revisión de este contenido, se incluirán temas que por razones de tiempo y prioridad se quedan fuera.

Juan Palacio



INFO ABOUT RIGHTS

1401289956290

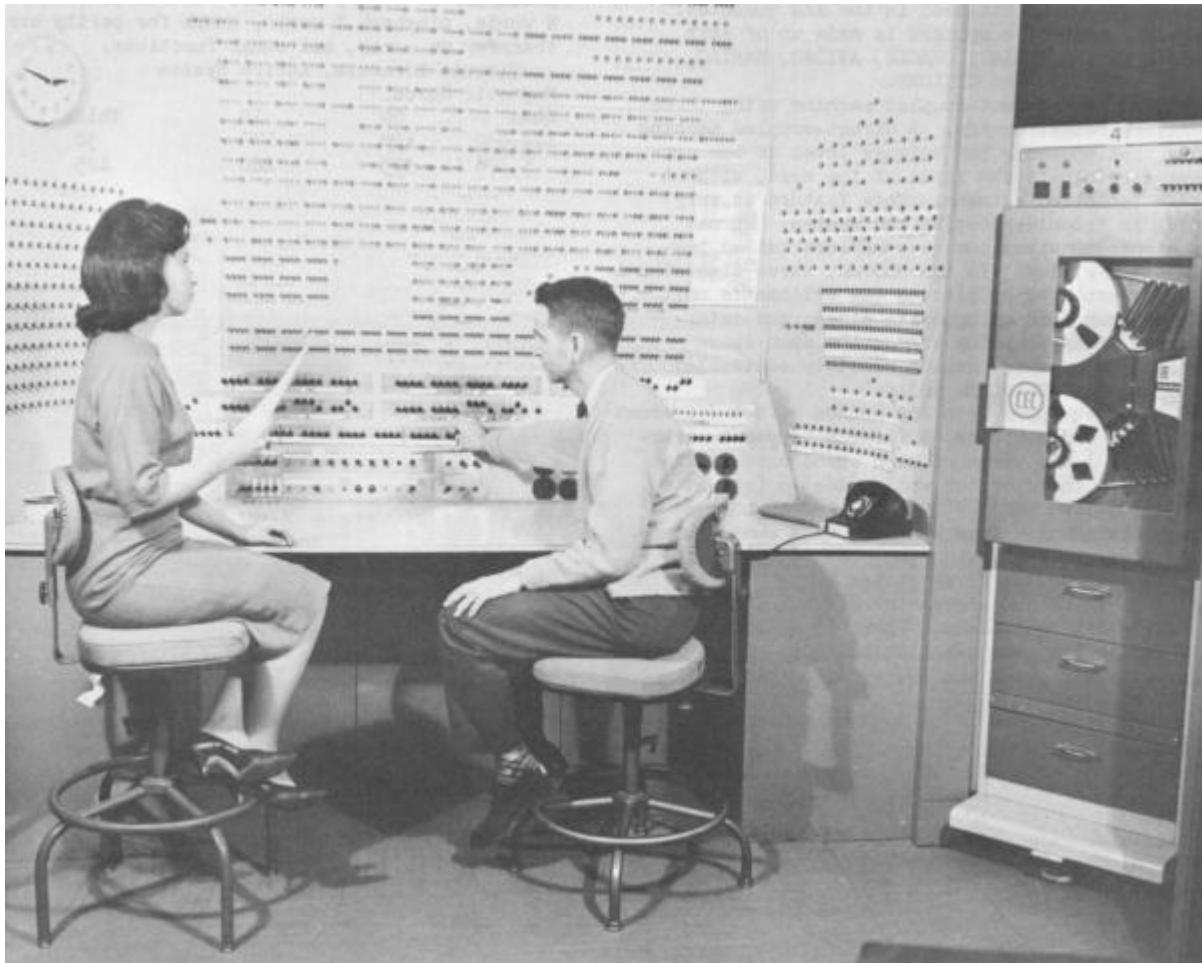
www.safecreative.org/work

(1) Para consultar los usos autorizados de este trabajo o contactar con el autor:
<http://www.safecreative.org/work/1401289956290>

Introducción a la Ingeniería del Software

1.0

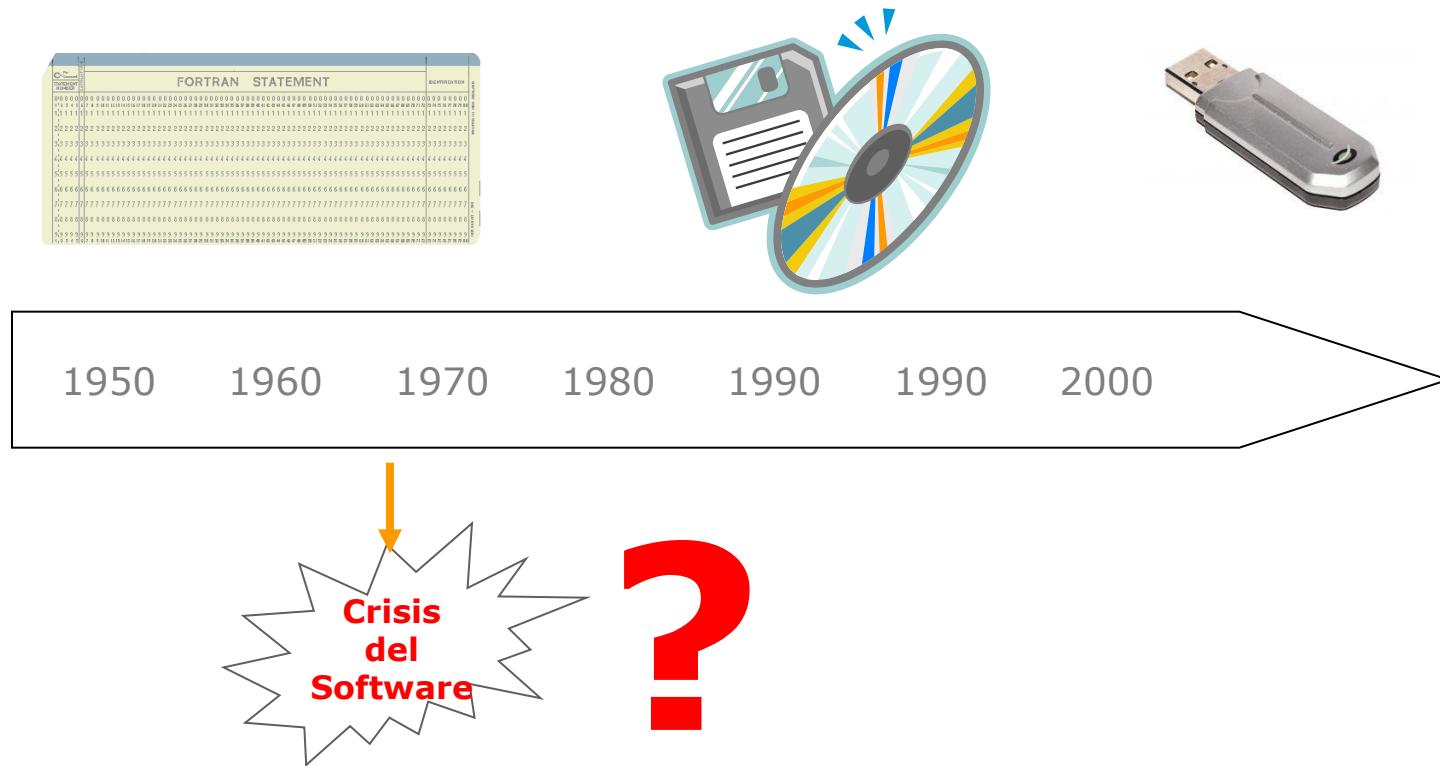


**1952**



1965

Crisis del software



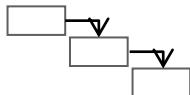
Crisis del software

Este problema se identificó por primera vez en 1968, año en el que la organización NATO desarrolló la primera conferencia sobre desarrollo de software, y en la que se acuñaron los términos “**crisis del software**” para definir a los problemas que surgían en el desarrollo de sistemas de software, e “**ingeniería del software**” para describir el conjunto de conocimientos que existían en aquel estado inicial.

Algunas referencias útiles para comprender cuáles eran los conocimientos estables para el desarrollo de software en 1968 son:

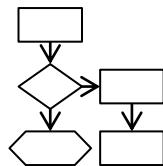
- En 1962 se publicó el primer algoritmo para búsquedas binarias.
- C. Böhm y G. Jacopini publicaron en 1966 el documento que creaba una fundación para la eliminación de “GoTo” y la creación de la programación estructurada.
- En 1968 los programadores se debatían entre el uso de la sentencia GoTo, y la nueva idea de programación estructurada; ese era el caldo de cultivo en el que Edsger Dijkstra escribió su famosa carta “GoTo Statement Considered Harmful” en 1968.
- La primera publicación sobre programación estructurada no vio la luz hasta 1974, publicada por Larry Constantine, Glenford Myers y Wayne Stevens.
- El primer libro sobre métrica de software fue publicado en 1977 por Tom Gilb.
- El primero sobre análisis de requisitos apareció en 1979

LA PRIMERA SOLUCIÓN A LA CRISIS DEL SOFTWARE



INGENIERÍA DEL SOFTWARE

Aplicación de un enfoque sistemático, disciplinado y cuantificable en el desarrollo, operación y mantenimiento de software.



INGENIERÍA DE PROCESOS

(Producción basada en procesos / mejora continua del proceso)

Aplicación del principio de calidad de Jurán empleado en la producción industrial a la producción de software: "La calidad del resultado depende de la calidad del proceso" -> modelos de procesos: CMMI, ISO 15504...



GESTIÓN DE PROYECTOS

(Aplicación del modelo de gestión de proyectos predictivo en los proyectos de software)

Aplicación de procesos de planificación, ejecución y control para alcanzar un objetivo final en un plazo de tiempo determinado con el coste y nivel de calidad previstos



1980

**1986**

**1990**

**2000**



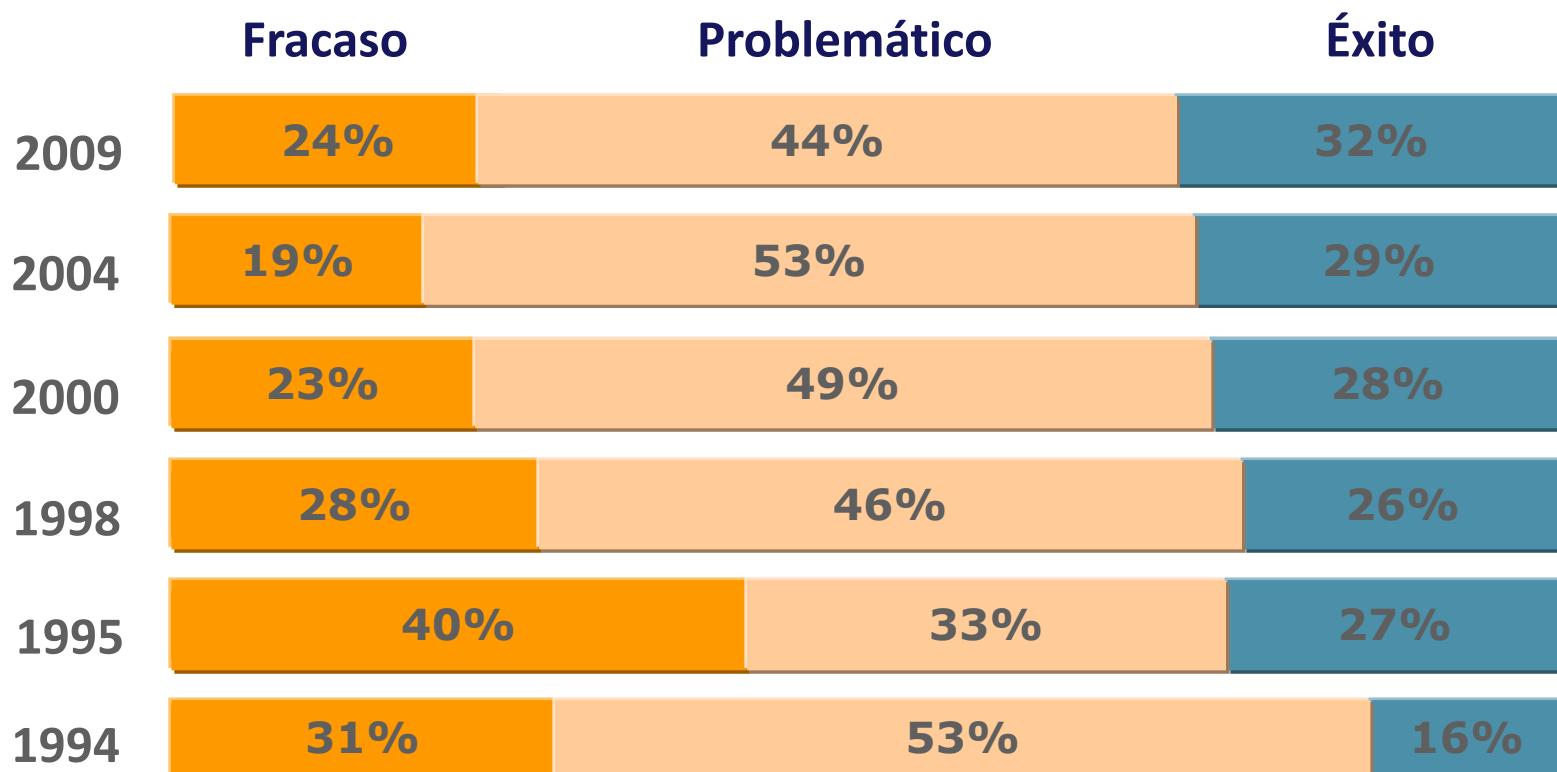
2005



2010

Crisis del software

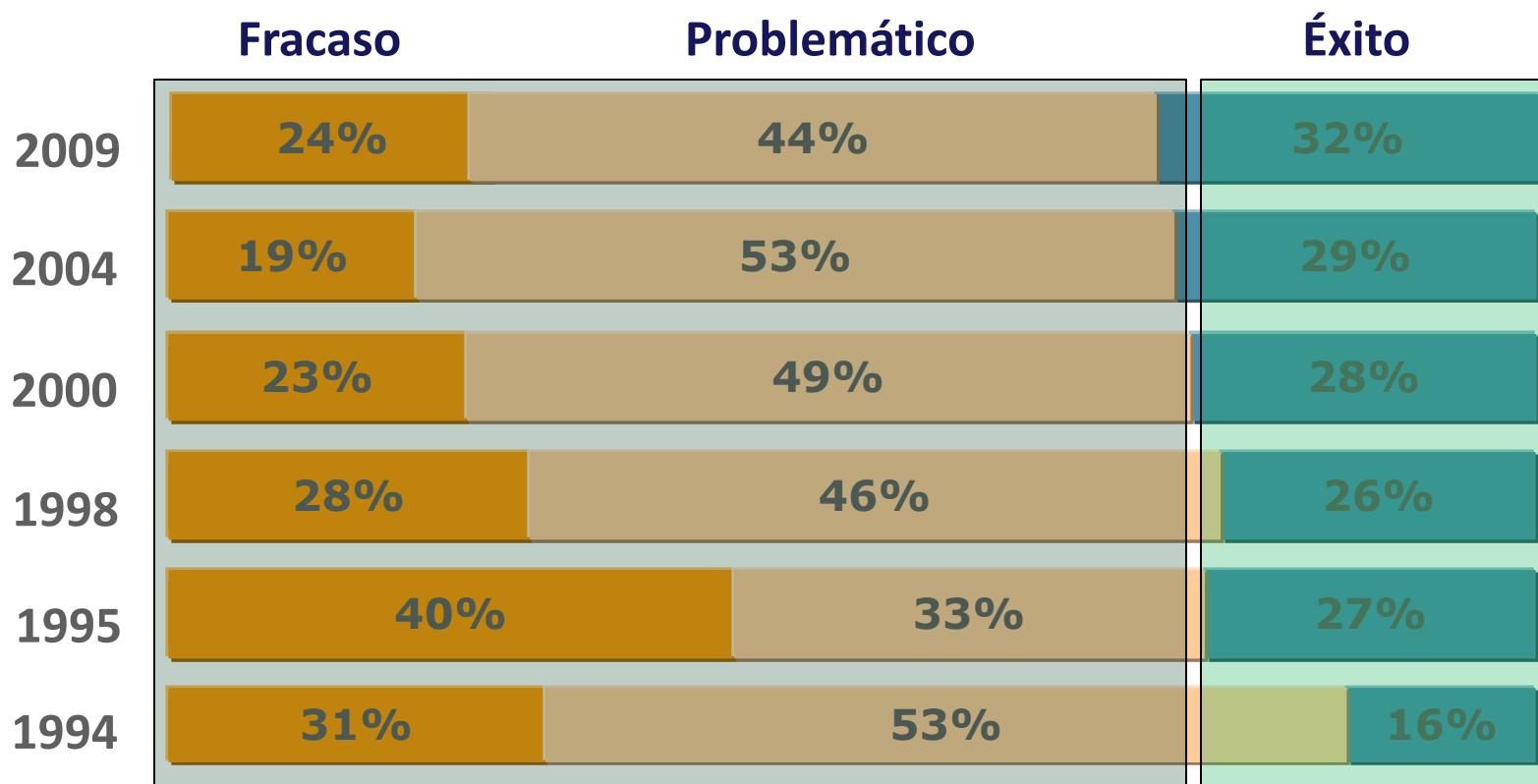
Proyectos para desarrollo de sistemas de software



Fuente: Standish Group Survey,

Crisis del software

Proyectos para desarrollo de sistemas de software



Fuente: Standish Group Survey,

Ingeniería del software

Definición original:

"Establecimiento y uso de principios de ingeniería para obtener software económico que trabaje de forma eficiente en máquinas reales".

Fritz Bauer, 1968 (conferencia NATO)

Otras definiciones

"Disciplina para producir software de calidad desarrollado sobre las agendas y costes previstos y satisfaciendo los requisitos".

S. Schach 1990, Software Engineering

**"(1) La aplicación de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software; esto es, la aplicación de la ingeniería al software.
(2) El estudio de (1)".**

IEEE 1993

Ingeniería del software

Desde 1968 hasta la fecha han sido muchos los esfuerzos realizados por los departamentos de informática de las universidades, y por organismos de estandarización (SEI, IEEE, ISO) para identificar las causas del problema y definir pautas estándar para la producción y mantenimiento del software.

Los esfuerzos se han encaminado en tres direcciones principales.

- Identificación de los factores clave que determinan la calidad del software.
- Identificación de los procesos necesarios para producir y mantener software.
- Acotación, estructuración y desarrollo de la base de conocimiento necesaria para la producción y mantenimiento de software.

El resultado ha sido la **necesidad de profesionalizar el desarrollo, mantenimiento y operación de los sistemas de software**, introduciendo métodos y formas de trabajo sistemáticos, disciplinados y cuantificables.

La forma de trabajo de programadores individuales surgida por la necesidad de los primeros programas, ha creado una cultura de la programación heroica, para el desarrollo de software que es la principal causa de los problemas apuntados, y en la actualidad una de las principales resistencias a la implantación de técnicas de ingeniería para el desarrollo de sistemas

Ingeniería del software

La Ingeniería del Software es una ingeniería muy joven que necesitaba:

- **Definirse a sí misma:** ¿Cuáles son las áreas de conocimiento que la comprenden?
- **Definir los procesos que intervienen en el desarrollo, mantenimiento y operación del software**
- **De las mejores prácticas, extraer modelos de cómo ejecutar esos procesos para evitar los problemas de la “crisis del software”**
- **Definir criterios unificadores para las tareas de requisitos, pruebas, gestión de la configuración, etc.**

Los estándares son útiles porque:

- Agrupan lo mejor y más apropiado de las buenas prácticas y usos del desarrollo de software.
- Engloban los “conocimientos”.
- Proporcionan un marco para implementar procedimientos de aseguramiento de la calidad.
- Proporcionan continuidad y entendimiento entre el trabajo de personas y organizaciones distintas.

Ingeniería del software

Desde la identificación del fenómeno “crisis del software”, han sido muchas las organizaciones que han abordado, con mayor o menor rigor, el análisis de problemas en el desarrollo de sistemas de software. Sus trabajos se han encaminado a la localización de las causas; y a la exposición en textos didácticos, normativos o estándares de procesos o prácticas necesarias para abordar el desarrollo, mantenimiento y operación con las mayores garantías de éxito.

Han sido muchos los departamentos de universidades, organismos de normalización o investigación nacionales o internacionales, sociedades de profesionales, departamentos de defensa, departamentos de calidad y procesos de empresas los que han ido generando normas y estándares.

Este compendio considera como entidades de mayor reconocimiento internacional, por sus trabajos y esfuerzos realizados para la normalización, y reconocimiento de la Ingeniería del software a: **ISO, IEEE- Computer Society y SEI.**

Ingeniería del software

■ ISO

Organización Internacional para la Estandarización. Fundada en 1947

Son miembros 87 países.



En 1987 la Organización Internacional para la Estandarización (ISO) y la Comisión Internacional Electrotécnica (IEC), establecieron un Comité Internacional (JTC1) para las Tecnologías de la Información. La misión del JTC1 es la "estandarización en el campo de campo de los sistemas de tecnologías de la información, incluyendo microprocesadores y equipos.

Los estándares o instrucciones técnicas más importantes para la Ingeniería del Software:

- **ISO/IEC 12207**
- **ISO/IEC TR 15504**

■ SEI

Instituto de Ingeniería del software. (SEI <http://www.sei.cmu.edu/>).



Integrado en la Universidad Carnegie Mellon.

Los trabajos y aportaciones realizadas por el Instituto de Ingeniería del Software a la Ingeniería del software son también referente mundial de primer orden, siendo la aportación más significativa los modelos de madurez de las capacidades: CMM y CMMI; que en sus casi 15 años de implantación efectiva en entornos de producción de software han demostrado su efectividad en las dos finalidades que cubren: como marco de referencia para mejora de procesos, y como criterio de evaluación para determinar la madurez, y por tanto fiabilidad de resultados previsibles de una organización de software.

Ingeniería del software

■ IEEE Computer Society

IEEE Es el Instituto de Ingenieros en electricidad y electrónica (Institute of Electrical and Electronics Engineers).

Su misión es preservar, investigar y promover la información de las tecnologías eléctricas y electrónicas.

Surgió en 1963 con la fusión del AIEE (Instituto Americano de Ingenieros Eléctricos) y el Instituto de Ingenieros de Radio (IRE).

La IEEE Computer Society (www.computer.org) es una sociedad integrada en IEEE, formada en la actualidad por más de 100.000 miembros en todo el mundo.

Su finalidad es avanzar en la teoría, práctica y aplicación de las tecnologías de la información. Realiza conferencias, publicaciones, cursos de formación, y desarrolla estándares.

Estándares para la Ingeniería del Software

IEEE ha desarrollado estándares para todas las áreas de Ingeniería del Software.

Algunos de ellos, correspondientes a las principales áreas específicas de la Ingeniería del Software son:

- IEEE Std. 830 Prácticas recomendadas para las especificaciones de software.
- IEEE Std. 1362 Guía para la especificación del documento de requisitos "ConOps"
- IEEE Std. 1063 Estándar para la documentación de usuario de software.
- IEEE Std. 1012 Estándar para la verificación y validación de software.
- IEEE Std. 1219 Estándar para el mantenimiento del software



Ingeniería del software

La Ingeniería del Software es una ingeniería muy joven que necesitaba:

- **Definirse a sí misma:** ¿Cuáles son las áreas de conocimiento que la comprenden?

→ **SWEBOK: Software Engineering Body of knowledge**

- **Definir los procesos que intervienen en el desarrollo, mantenimiento y operación del software**

→ **ISO/IEC 12207: Procesos del ciclo de vida del software**

- **De las mejores prácticas, extraer modelos de cómo ejecutar esos procesos para evitar los problemas de la “crisis del software”**

→ **CMM / CMMI
ISO/IEC TR 15504**

- **Definir estándares menores para dibujar criterios unificadores en requisitos, pruebas, gestión de la configuración, etc.**

→ **IEEE 830 - IEEE 1362 - ISO/IEC 14764 ...**

SWEBOK

El proyecto SWEBOK (Software Engineering Body of Knowledge) comenzó sus actividades de manera efectiva dentro del SWECC¹ en 1997 (aunque el comité SWECC se creó en 1993).

En el proyecto también están representados:

los dos principales organizaciones de estandarización en Ingeniería del Software: IEEE e ISO/IEC JTC1/SC/.

Los autores de las tres principales obras de Ingeniería del Software: Steve Mc Connell, Roger Pressman e Ian Sommerville.

Universidad de Québec (Montreal)

Empresas y organizaciones como: [Rational](#), [SAP](#), [Boeing](#), [Construx](#), [MITRE](#), [Raytheon](#),

En 2001 el proyecto publicó ya una definición consensuada del cuerpo de conocimiento aceptado en la ingeniería del software (<http://www.swebok.org>).

Las fuentes de información para la identificación de las áreas de conocimiento han sido los índices de textos genéricos sobre la Ingeniería del Software, los currícula para licenciatura y postgrado en Ingeniería de Software, y los criterios de admisión que se utilizan en el postgrado. Todos estos datos se han organizado siguiendo el estándar ISO/IEC 12207.

El cuerpo de conocimiento identificado por el proyecto SWEBOK se ha configurado como el estudio más relevante y como la referencia de más autoridad en toda la comunidad informática para la acotación y descripción de los conocimientos que configuran la Ingeniería del software.

¹ Software, Engineering Coordinating Committee”, Comisión creada por IEEE Computer Society y ACM (Association for Computer Machinery) para definir el cuerpo de la Ingeniería del Software

Ingeniería del software

SWEBOK da el primer paso necesario para constituir a la Ingeniería del Software como profesión: la delimitación del cuerpo de conocimiento que comprende la profesión. Sin esta delimitación no es posible validar de forma universal exámenes de licenciatura, no es posible la preparación para acceder a la profesión, y no hay un consenso sobre el contenido de su currículo.

El proyecto parte de la suposición de que es necesario establecer cuál es el cuerpo de conocimiento que deben conocer los ingenieros del software, y en su desarrollo ha agrupado este conocimiento en 10 áreas

- **Requisitos**
- **Diseño**
- **Construcción**
- **Pruebas**
- **Mantenimiento**
- **Gestión de la configuración**
- **Gestión**
- **Procesos**
- **Herramientas y métodos**
- **Calidad**

Es importante resaltar que estas áreas no incluyen aspectos importantes de las tecnologías de la información, tales como lenguajes específicos de programación, bases de datos relacionales o redes o tecnología de redes y comunicaciones.

Esta es una consecuencia de la distinción que entre “esencia” y “accidente” se establece desde un enfoque de ingeniería.

Por supuesto que un Ingeniero de Software debe conocer las técnicas de cada momento, pero la definición de procesos y metodología de trabajo es la “esencia” de la profesión. Así por ejemplo, el área de conocimiento de requisitos, sí que puede considerarse como “esencia” de la profesión. Los problemas que pueden derivarse en un proyecto por una mala obtención o gestión de los requisitos son indistintos del hardware o lenguaje de programación empleado. Eran los mismos hace dos décadas que ahora, y todo nos hace suponer que seguirán siendo idénticos dentro de otros cuatro lustros.

ISO 12207: PROPÓSITO

Establecer un estándar para evitar una situación de Torre de Babel en la gestión e ingeniería del software, proporcionando un marco y un lenguaje común en la disciplina del software

Establece un marco común para el ciclo de vida del software para

- Adquisición, suministro, desarrollo, operación y mantenimiento del software
- Gestionar, controlar y mejorar el marco
- Como base de referencia para el trabajo e intercambio entre organizaciones de software

Ciclo de vida del software

Periodo de tiempo que comienza al concebir la idea de un nuevo sistema de software, y termina cuando este se retira y deja de funcionar.

ISO 12207: PROPÓSITO

El estándar no prescribe:

- Que deba emplearse ningún tipo de documentación específica.
- Que deba emplearse un tipo específico de ciclo de desarrollo.
- Métodos concretos para el desarrollo, mantenimiento u operación del software.

Define el QUÉ, no el CÓMO.

Dice cuáles son los procesos, actividades y tareas implicados en el desarrollo, mantenimiento y operación de los sistemas de software, asentando un marco estándar de referencia internacional, pero no se ocupa ni prescribe técnicas específicas.

El estándar sirve de referencia desde dos perspectivas diferentes:

Para la adquisición de sistemas y servicios de software.

Para el suministro, desarrollo, mantenimiento y operación de productos de software.

El estándar no cubre el desarrollo de productos de software para distribución comercial masiva (productos "en caja").

No se trata de un estándar de certificación, tipo ISO 9000, sino de un estándar para la normalización.

ISO 12207 (2005¹): PROCESOS

5. Procesos primarios

5.1 Adquisición

5.2 Suministro

5.3
Desarrollo

5.3
Operación

5.3
Mantenimiento

6.- Procesos de soporte

6.1 Documentación

6.2 Gestión de la configuración

6.3 Control de calidad

6.4 Verificación

6.5 Validación

6.6 Reuniones

6.7 Auditoría

6.8 Resolución de problemas

7. Procesos organizacionales

7.1 Gestión

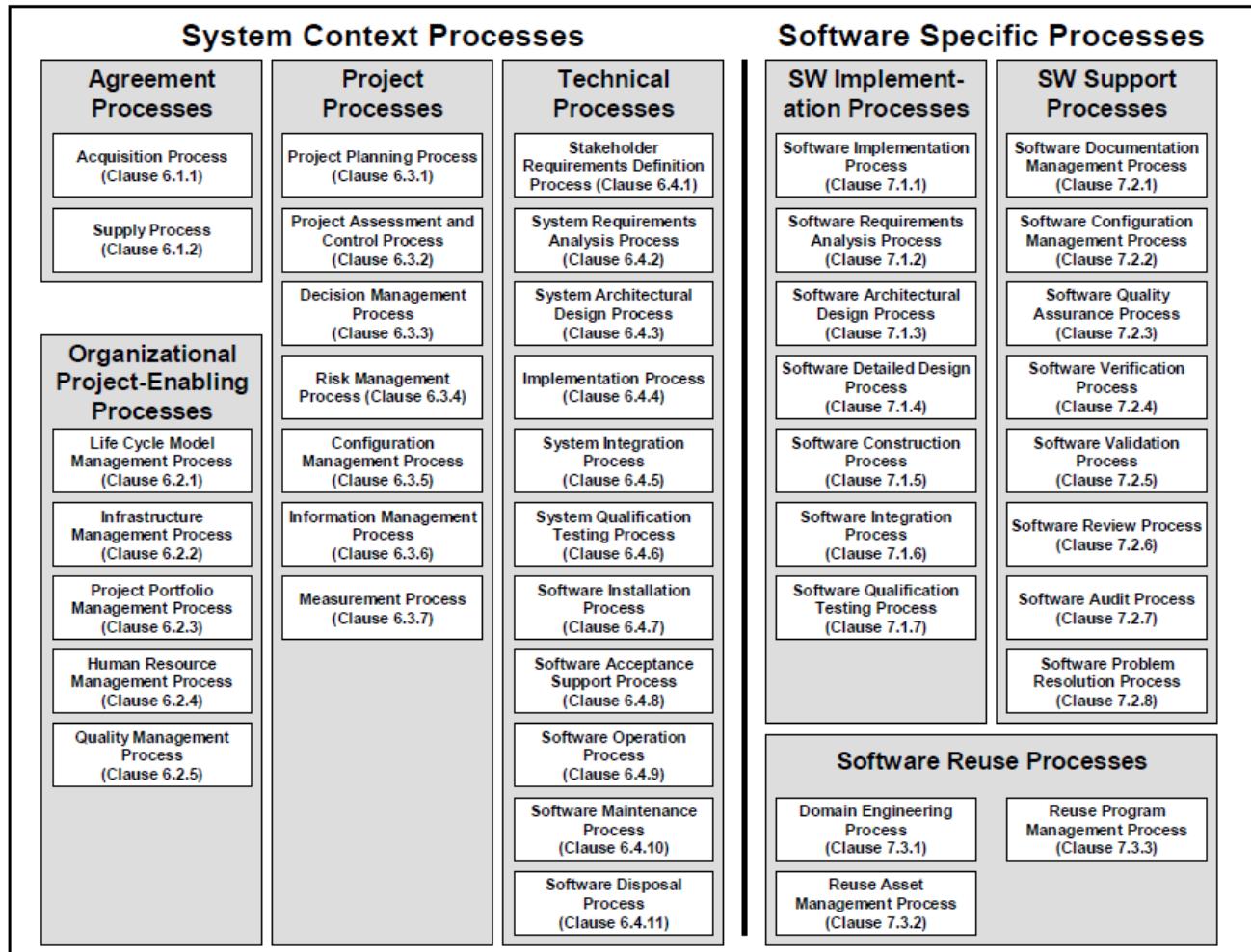
7.2 Infraestructura

7.3 Mejora

7.4 Formación

(1) La versión actual de este estándar ha modificado el esquema de procesos. Para el ámbito de estos apuntes se mantiene esta versión de 2005 por ofrecer una estructura más comprensible.

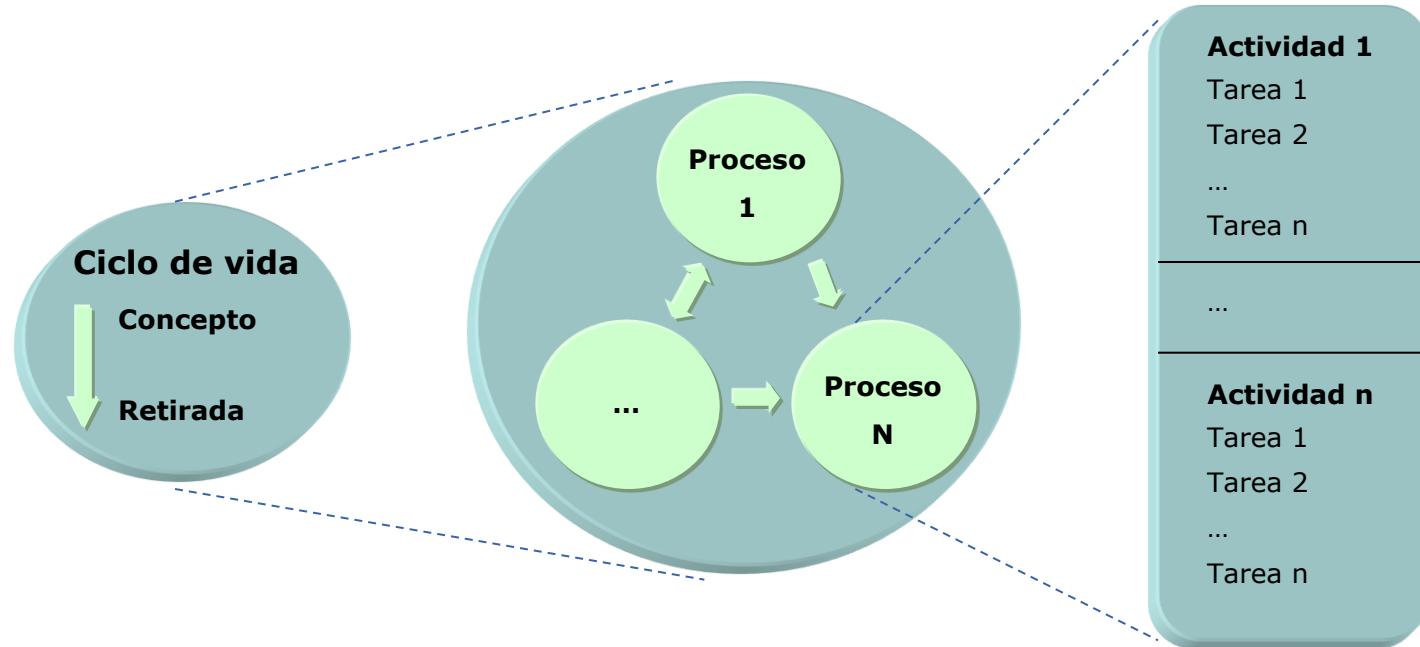
ISO 12207 (2008¹): PROCESOS



(1) Para el ámbito y finalidad de estos apuntes, manejamos mejor la versión previa de 2005 por ofrecer una estructura más comprensible

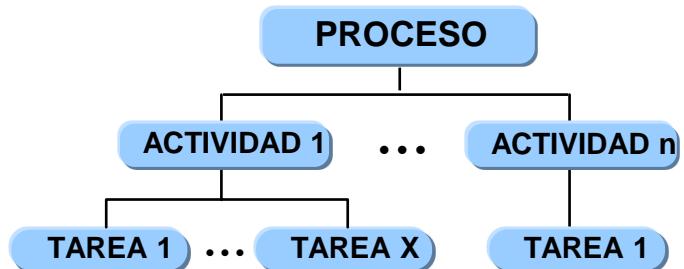
ISO 12207

- ISO 1227 define los procesos que componen el ciclo de vida del software

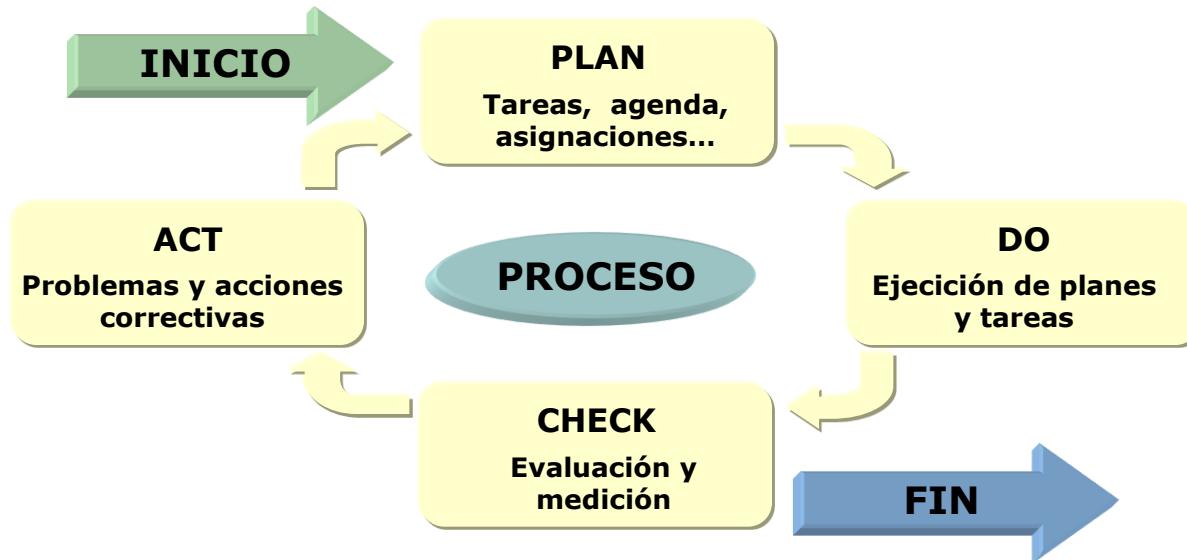


ISO 12207

- Un proceso está compuesto por actividades.
- Una actividad está compuesta de tareas.



- La descomposición del proceso en actividades y tareas se realiza sobre el concepto de ciclo de mejora PDCA "Plan – Do – Chek – Act" (Planificación, ejecución, medición y mejora)



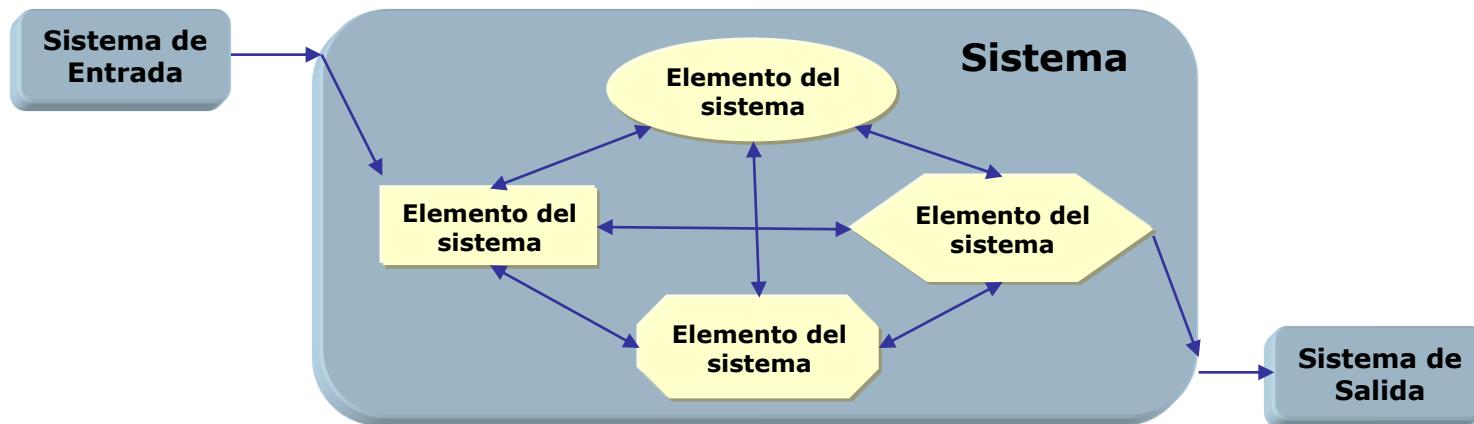
INGENIERÍA DE SISTEMAS

- ISO 12207 establece un nexo con la Ingeniería de sistemas al considerar al **software como parte de un sistema**.
- Desde esta perspectiva se establece a la **Ingeniería de sistemas como fundamento de la Ingeniería del Software**.

¿Qué es un sistema?

"Colección de componentes organizados para cumplir una función o conjunto de funciones específicas".

IEEE Standard 610.12-1990



"Colección de elementos relacionados de forma que puedan realizar un objetivo tangible".

Pressman 1982

INGENIERÍA DE SISTEMAS

Sistema

conjunto de elementos de hardware, software, personas, procedimientos, herramientas y otros factores organizativos, organizados para llevar a cabo un objetivo común.

Sistema de software

Sistema o sub-sistema formado por una colección de programas y documentación que de forma conjunta satisfacen unos determinados requisitos.

Un sistema de software puede ser en sí mismo un sistema independiente que, por ejemplo, realiza su objetivo en un ordenador independiente. A este tipo de sistemas se les denomina también “**sistema intensivo de software**”, porque el sistema es prácticamente software.

Un sistema de software puede ser también una parte de un sistema mayor. En cuyo caso se trata en realidad de un “sub-sistema de software”.

Por ejemplo, el sistema de software de un avión de combate es en realidad el sub-sistema de software del avión.

Ingeniería de sistemas

El término “Ingeniería de sistemas” surgió por primera vez en 1956, y fue propuesto por H. Hitch, presidente del departamento de Ingeniería Aeronaútica de la Universidad de Pensilvania, para intentar desarrollar una disciplina de ingeniería que pudiera abarcar el desarrollo de grandes sistemas que empleaban diversas disciplinas de ingenierías específicas: construcción de bombarderos, submarinos, etc.

Los principios de Ingeniería de sistemas desarrollados en los 60 y 70 se aplicaron en programas como el Apolo, o el programa de misiles balísticos USAF/USN.

INGENIERÍA DE SISTEMAS

Algunas definiciones

Ingeniería de sistemas comprende la función de gestionar todo el esfuerzo de desarrollo para conseguir un balance óptimo entre todos los elementos del sistema. Es el proceso que transforma la necesidad operacional en la descripción de los parámetros del sistema, e integra esos parámetros para mejorar la eficiencia general del sistema.

Defense Systems Management College, 1989

Los procesos de ingeniería de sistemas integran las secuencias de actividades y decisiones que transforman la definición de una necesidad en un sistema, que con un ciclo de vida optimizado, consigue un balance óptimo de todos sus componentes.

USAF, 1985

La principal función de la ingeniería de sistemas es garantizar que el sistema satisface los requisitos durante todo el ciclo de vida. Todas las demás consideraciones se alinean sobre esta función.

Wymore 1993

La ingeniería de sistemas define el plan para gestionar las actividades técnicas del proyecto. Identifica el ciclo de desarrollo y los procesos que será necesario aplicar. Desde la Ingeniería de sistemas se desarrolla la línea base técnica para todo el desarrollo, tanto de hardware como de software.

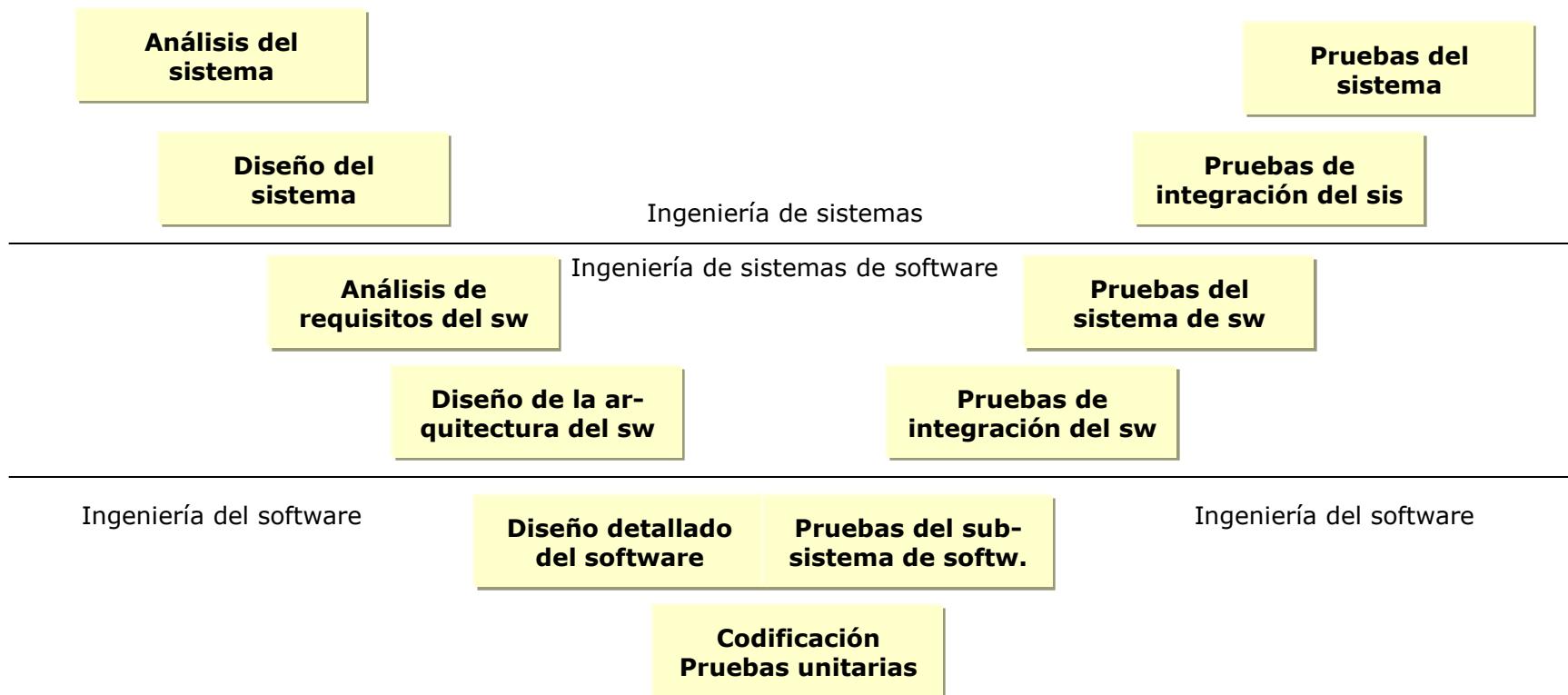
INGENIERÍA DE SISTEMAS

Funciones de la Ingeniería de sistemas

- **Definición del problema:** Determinación de las expectativas hacia el producto, necesidades y restricciones obtenidas y analizadas en los requisitos del sistema. Trabaja cerca del cliente para establecer las necesidades operacionales.
- **Análisis de la solución:** Determinar las opciones posibles para satisfacer los requisitos y las restricciones. Estudiar y analizar las posibles soluciones. Seleccionar la mejor, sopesando las necesidades inmediatas, opciones de implementación, utilidad, evolución del sistema...
- **Planificación de los procesos:** Determinar los grupos de tareas técnicas que se deben realizar, el esfuerzo requerido para cada una, su prioridad y los riesgos que implican para el proyecto.
- **Control de los procesos:** Determinar los métodos para controlar las actividades técnicas del proyecto y los procesos; la medición del progreso, revisión de los productos intermedios y ejecución de las acciones correctivas, cuando corresponda.
- **Evaluación del producto:** Determinar la calidad y cantidad de los productos elaborados, a través de evaluaciones, pruebas, análisis, inspecciones...

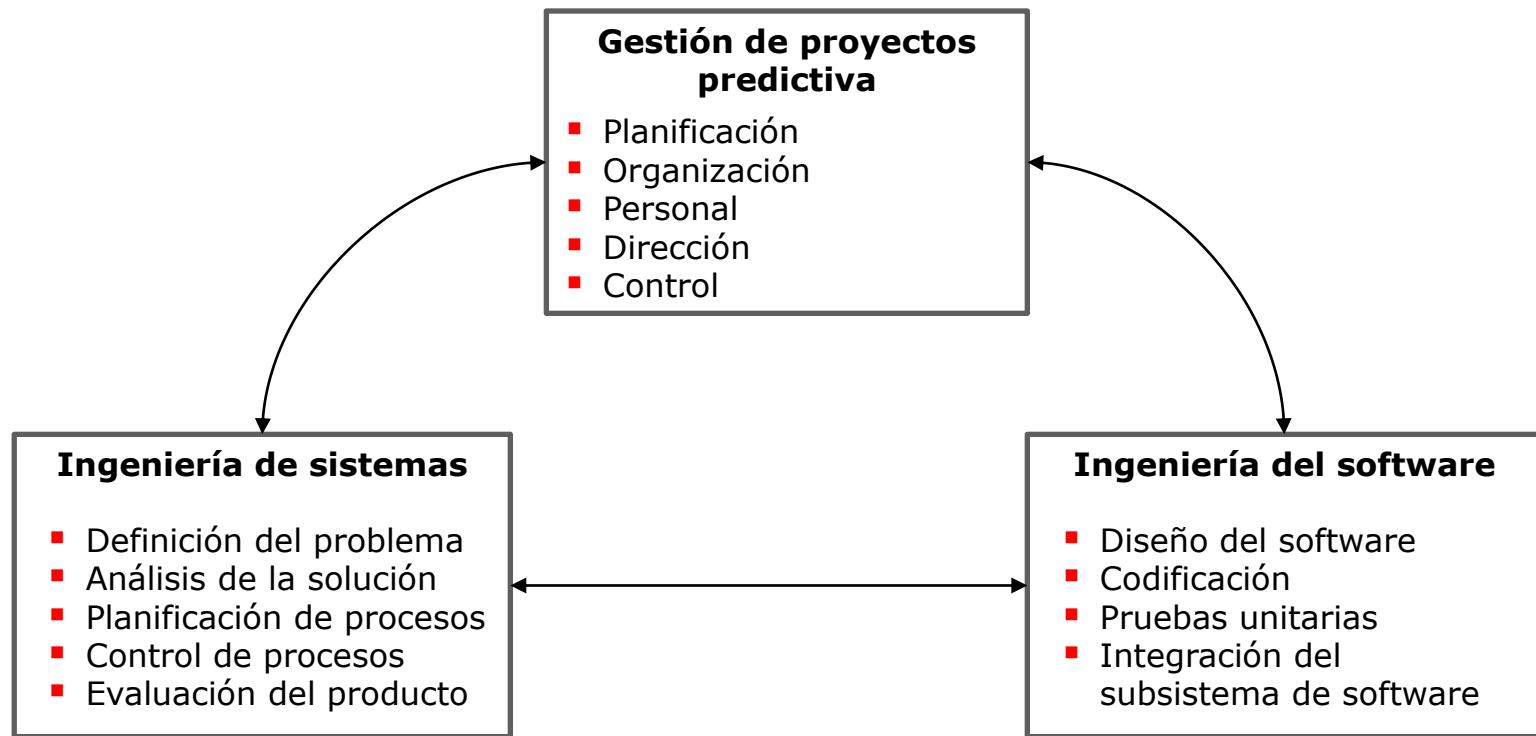
INGENIERÍA DE SISTEMAS

Ingeniería de sistemas – Ingeniería de sistemas de software – Ingeniería del software

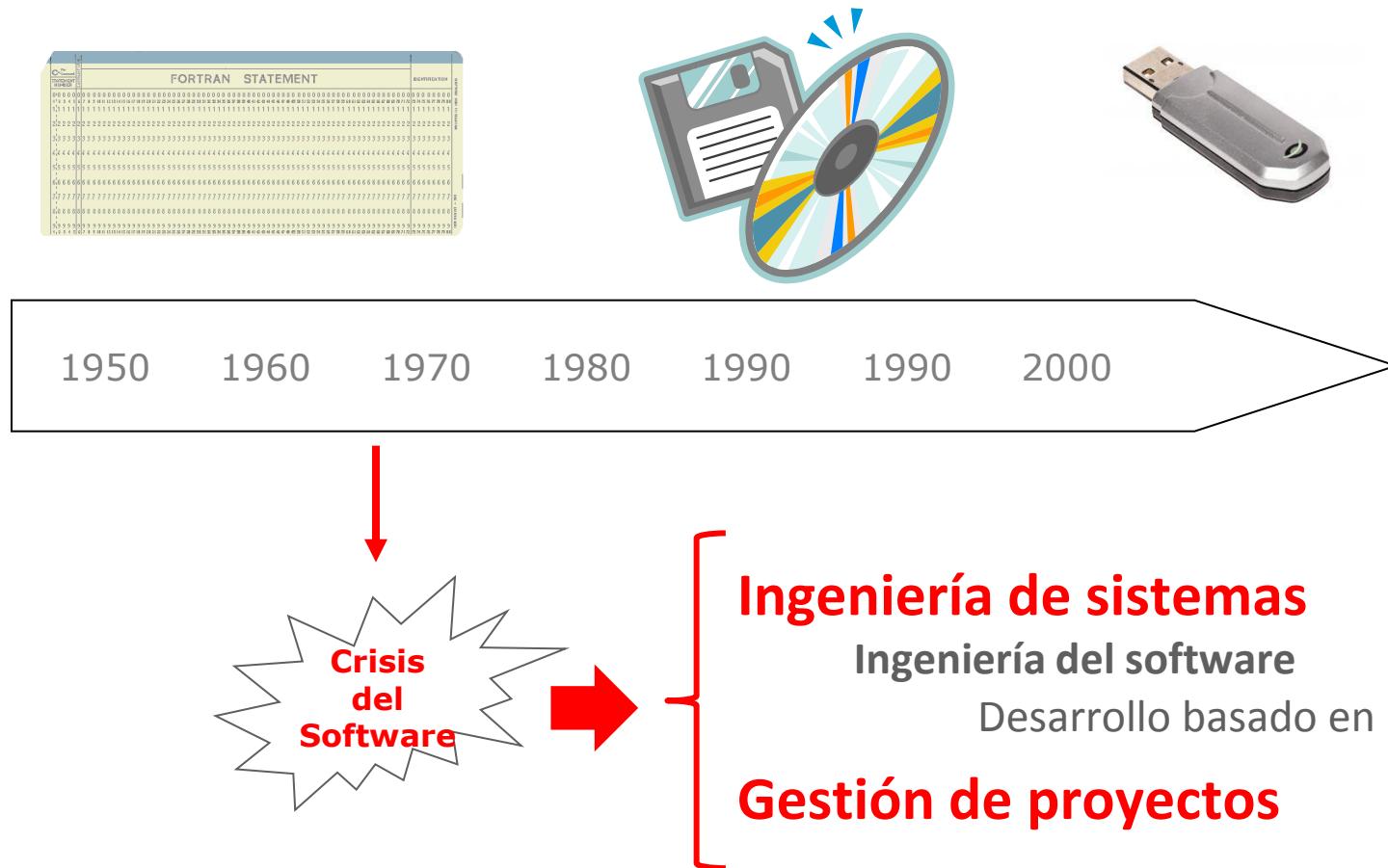


INGENIERÍA DE SISTEMAS

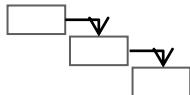
Ing. de sistemas – Gestión de proyectos predictiva – Ing. del Software



LA PRIMERA SOLUCIÓN A LA CRISIS DEL SOFTWARE

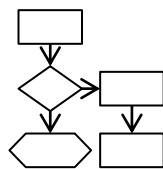


LA PRIMERA SOLUCIÓN A LA CRISIS DEL SOFTWARE



INGENIERÍA DEL SOFTWARE

Aplicación de un enfoque sistemático, disciplinado y cuantificable en el desarrollo, operación y mantenimiento de software.



INGENIERÍA DE PROCESOS

(Producción basada en procesos / mejora continua del proceso)

Aplicación del principio de calidad de Jurán empleado en la producción industrial a la producción de software: "La calidad del resultado depende de la calidad del proceso" -> modelos de procesos: CMMI, ISO 15504...

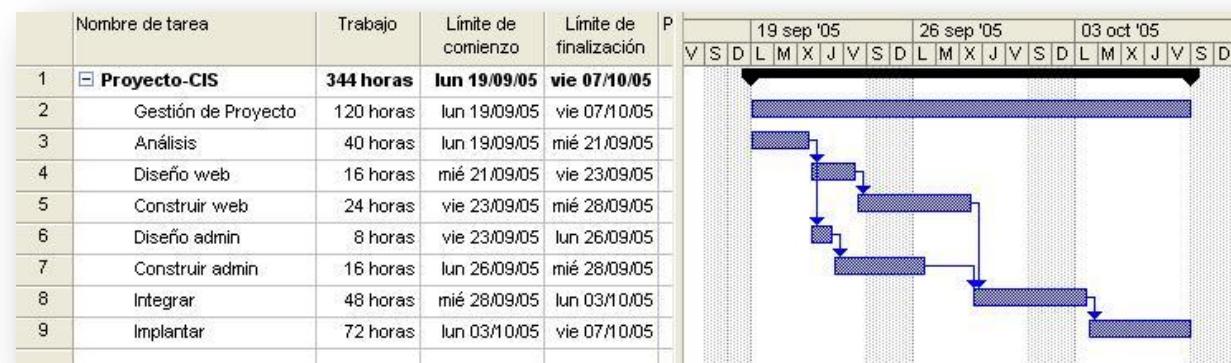


(Aplicación del modelo de gestión de proyectos predictivo en los proyectos de software)

Aplicación de procesos de planificación, ejecución y control para alcanzar un objetivo final en un plazo de tiempo determinado con el coste y nivel de calidad previstos.

Gestión de proyectos predictiva

1.0



PROYECTOS



Conjunto único de actividades necesarias para producir un resultado definido, en un rango de fechas determinado y con una asignación específica de recursos

GESTIÓN DE PROYECTOS (predictiva)



Gestión basada en **PLANIFICACIÓN** **SEGUIMIENTO**

1

¿Qué hacer?

2

Planificación del trabajo

3

Ejecución y control

PROYECTOS



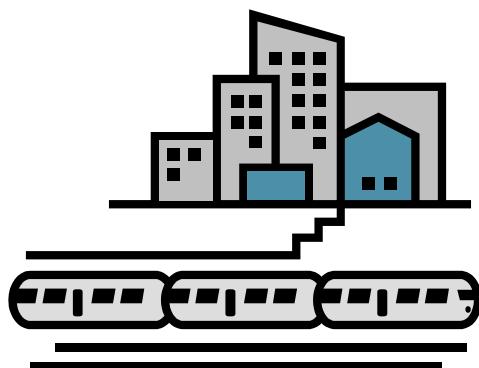
1965.- IPMA

1969.- PMI

1989.- PRINCE 2



PROYECTOS



A **tiempo**

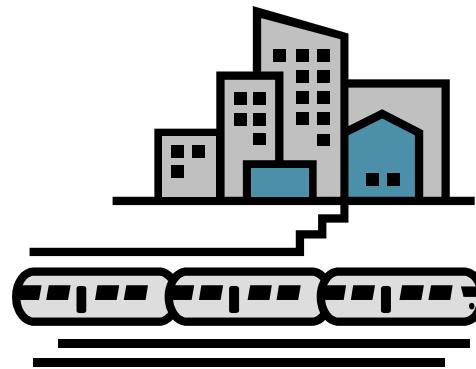
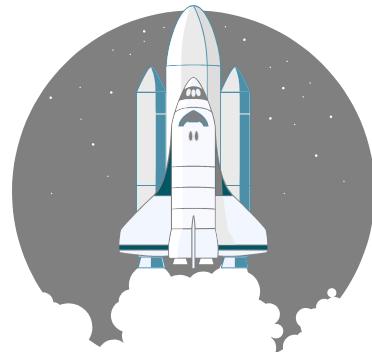
En costes

Calidad



ÉXITO

NUEVA PROFESIÓN

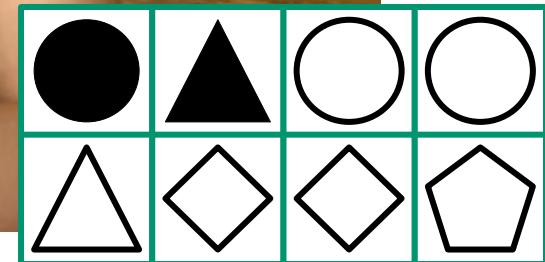


Professionalización del cuerpo de conocimiento para desarrollo de proyectos

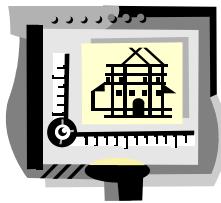
PRÁCTICA: GESTIÓN PREDICTIVA



cc-by: [LizMarie](#)



PRÁCTICA: GESTIÓN PREDICTIVA



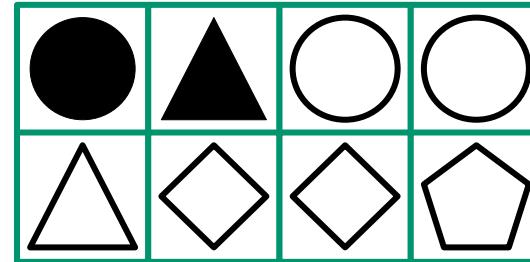
Lo previsto



A tiempo



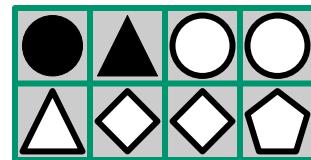
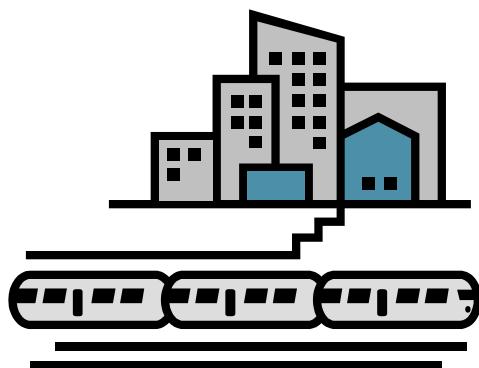
En costes



15 Turnos



PRÁCTICA: GESTIÓN PREDICTIVA



Lo previsto



A tiempo



En costes



ÉXITO

PRÁCTICA: GESTIÓN PREDICTIVA



Gestión basada en **PLANIFICACIÓN** **SEGUIMIENTO**

1

¿Qué hacer?

2

Planificación del trabajo

3

Ejecución y control

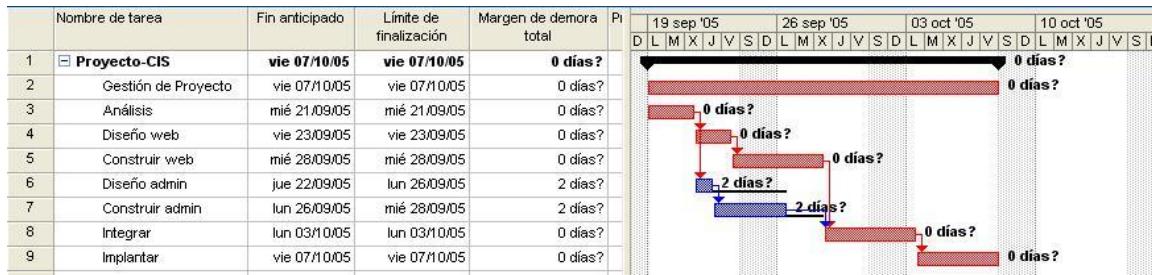
La forma más eficiente de hacer un trabajo es hacerlo bien a la primera

Watts S. Humphrey

GESTIÓN PREDICTIVA<->PRODUCCIÓN BASADA EN PROCESOS

**LA FORMA MÁS EFICIENTE DE
HACER UN TRABAJO, ES
HACERLO BIEN A LA PRIMERA**

Watts S. Humphrey
Creador de los modelos CMM - CMMI

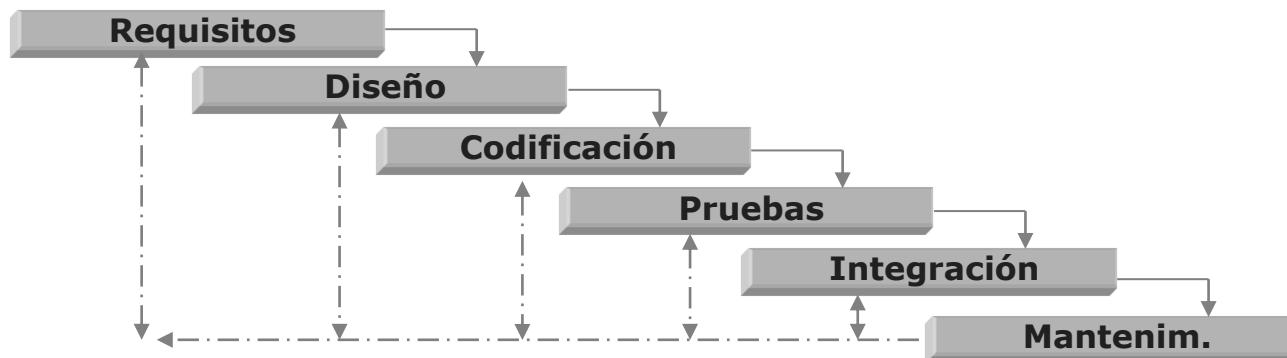


GESTIÓN PREDICTIVA<->PRODUCCIÓN BASADA EN PROCESOS

**LA FORMA MÁS EFICIENTE DE
HACER UN TRABAJO, ES
HACERLO BIEN A LA PRIMERA**

Watts S. Humphrey

Creador de los modelos CMM - CMMI



OPERACIONES Y PROYECTOS

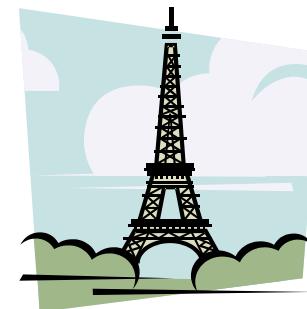
Las empresas y las organizaciones en general llevan a cabo su trabajo bajo la forma de proyectos o de operaciones.

Características comunes a las operaciones y los proyectos

- Son realizados por personas
- Disponen de recursos limitados
- Su ejecución se controla y responde a una planificación.

Diferencias entre operaciones y proyectos

Los proyectos son temporales y únicos, mientras que las operaciones realizan siempre los mismos procesos de forma continua.



PROYECTOS

Características de los proyectos

- Son realizados por personas
- Disponen de recursos limitados
- Su ejecución se controla y responde a una planificación.
- **Tienen un inicio y fin definidos**
- **Tienen como finalidad producir un producto o servicio único**
- **Ciclo de vida planificado**

Temporalidad

Cada proyecto tiene un inicio y fin definidos. El fin se alcanza cuando se consiguen los objetivos del proyecto por razones objetivas se decide abortar su ejecución.

Producto único

La finalidad del proyecto es realizar algo que no se piensa repetir de forma sistemática.

Ciclo de vida planificado

Característica derivada de la temporalidad y el objetivo único. El producto o servicio se lleva a cabo de forma incremental siguiendo unos pasos planificados que constituyen el ciclo de vida del desarrollo.

Áreas de conocimiento de la gestión de proyectos

Gestión de la integración del proyecto

- Desarrollo del plan de proyecto
- Ejecución del plan de proyecto
- Control integrado del cambio

Gestión del ámbito del proyecto

- Inicio
- Planificación del ámbito
- Definición del ámbito
- Verificación del ámbito
- Control de cambio del ámbito

Gestión de agenda

- Definición de la actividad
- Secuencia de la actividad
- Estimación de tiempos
- Desarrollo de la agenda
- Control de la agenda

Gestión de costes

- Plan de recursos
- Estimación de costes
- Presupuesto
- Control de costes

Gestión de la calidad del proyecto

- Plan de calidad
- Aseguramiento de la calidad
- Control de calidad

Gestión de los recursos humanos del proyecto

- Plan de organización
- Incorporación de personas
- Desarrollo del equipo

Gestión de las comunicaciones del proyecto

- Plan de comunicaciones
- Distribución de la información
- Informes de eficiencia
- Cierre administrativo

Gestión de riesgos del proyecto

- Plan de riesgos
- Identificación de riesgos
- Análisis cuantitativo de riesgos
- Análisis cualitativo de riesgos
- Plan de exposición de riesgos
- Monitorización y control de ries.

Gestión de compras

- Plan de necesidades
- Plan de compras
- Compras
- Selección de proveedores
- Contratación administrativa
- Cierre de contrato

Relación de la G. de proyectos con otras áreas de gestión

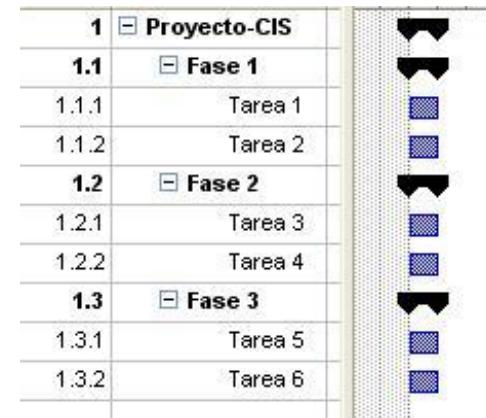


El principal conocimiento que se necesita para gestionar proyectos es exclusivo de la gestión de proyectos, pero para llevar a cabo la gestión debe complementarse con conocimientos que pertenecen al área de “management” en general y con conocimientos específicos del negocio al que va a servir el proyecto.

Gestión de proyectos: principales conceptos y prácticas

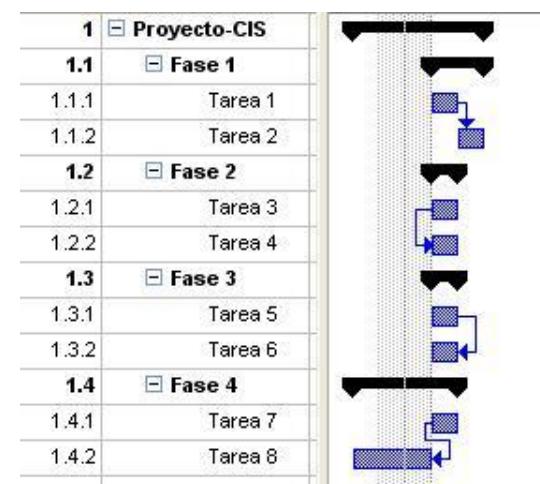
WBS

Work Breakdown Structure: Estructura de las tareas en las que se descompone el proyecto.



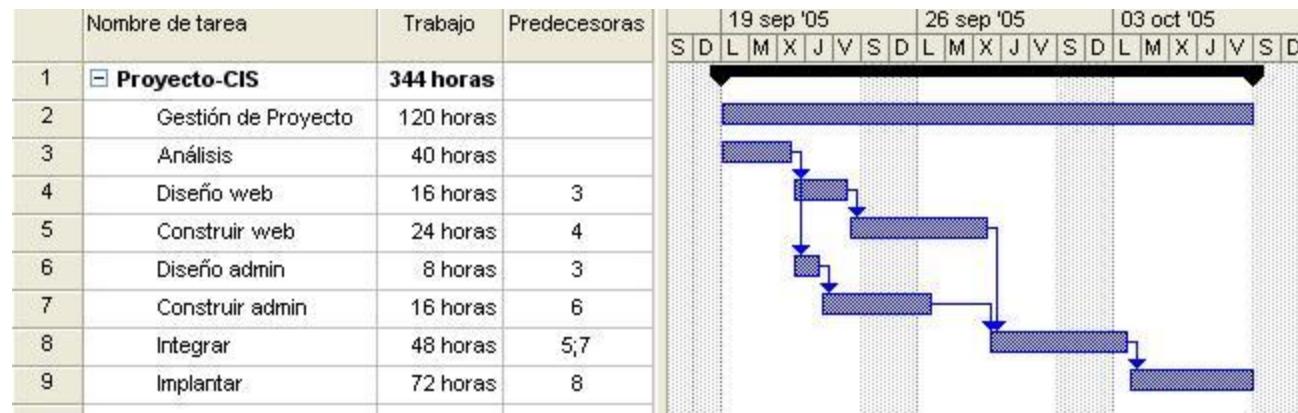
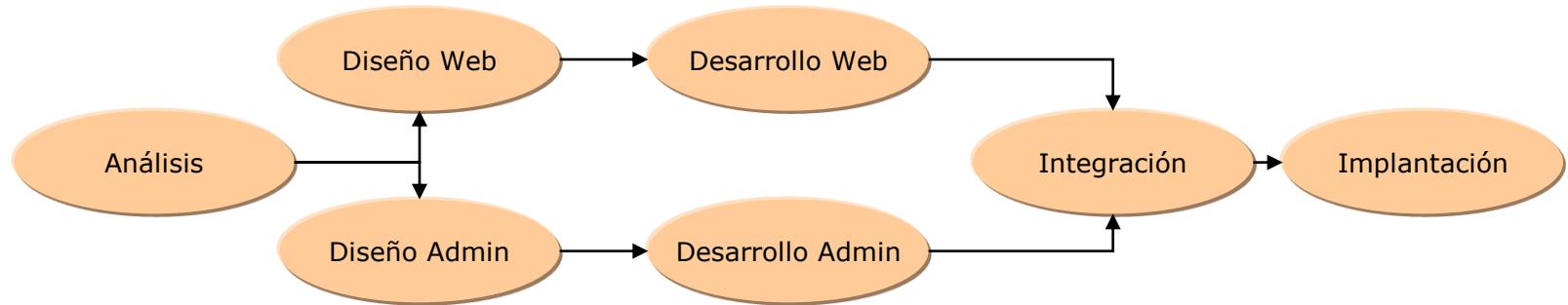
Planificando el proyecto: Tipos de dependencias entre tareas

- FC (de Fin a Comienzo) – la más habitual
- CC (de Comienzo a Comienzo)
- FF (de Fin a Fin)
- CF (de Comienzo a Fin) – suele ser problemática



Gestión de proyectos: principales conceptos y prácticas

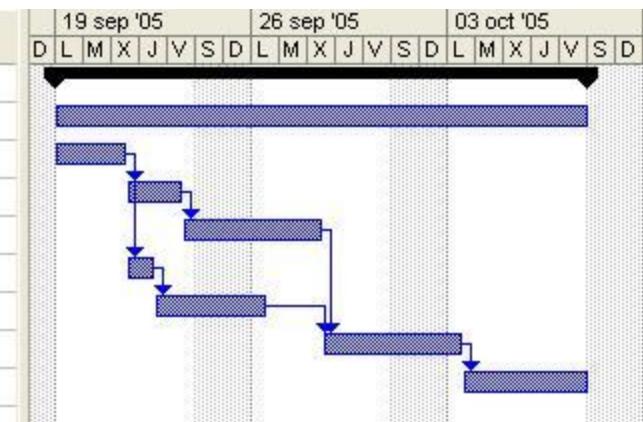
Tipos de dependencias entre tareas



Gestión de proyectos: principales conceptos y prácticas

Asignación de recursos y personas a las tareas

	Nombre de tarea	Trabajo	Predecesoras	Nombres de los recursos
1	■ Proyecto-CIS	344 horas		
2	Gestión de Proyecto	120 horas		Gestor
3	Análisis	40 horas		Programador1;Programador2
4	Diseño web	16 horas	3	Programador1
5	Construir web	24 horas	4	Programador1
6	Diseño admin	8 horas	3	Programador2
7	Construir admin	16 horas	6	Programador2
8	Integrar	48 horas	5,7	Programador2;Programador1
9	Implantar	72 horas	8	Programador2;Programador1



Optimización

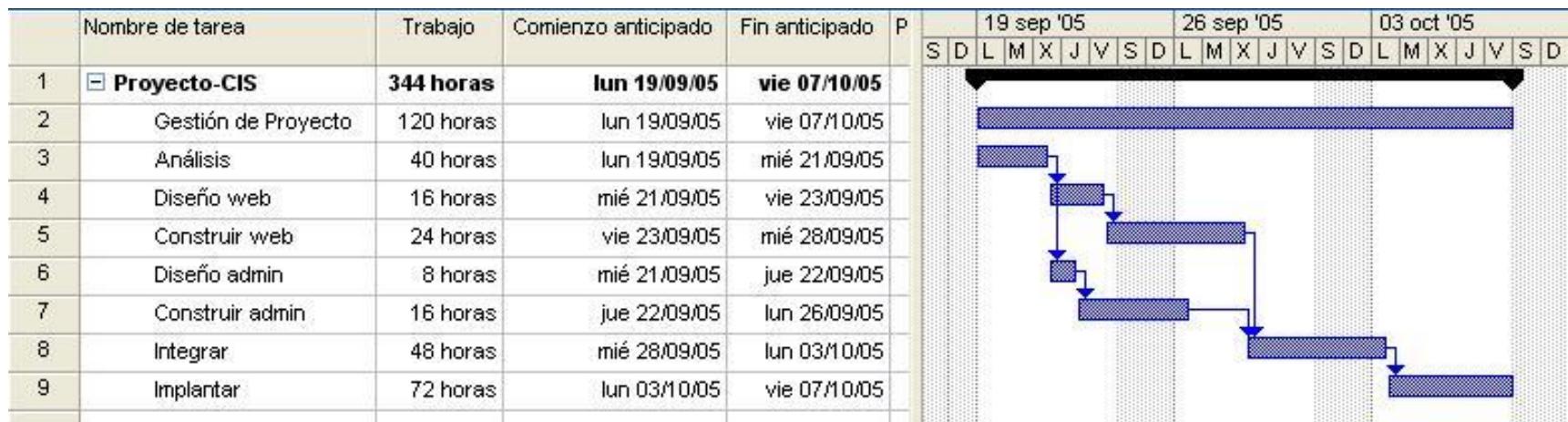
Conceptos clave para la optimización del proyecto

- Paso adelante
- Paso atrás
- Ruta crítica
- Reasignación de recursos

Gestión de proyectos: principales conceptos y prácticas

Paso adelante

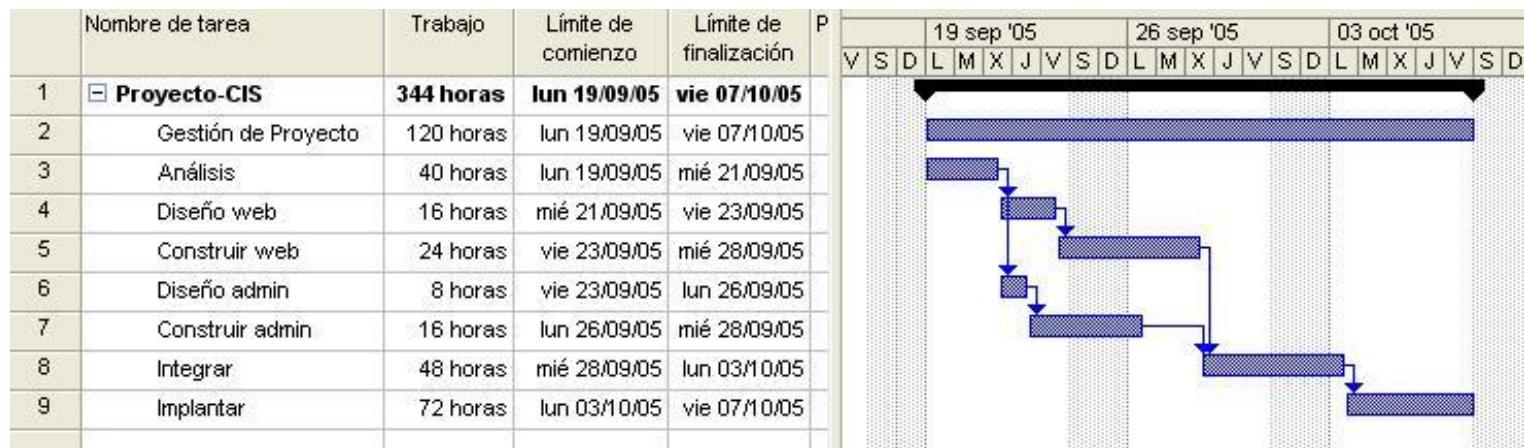
- Estimar la fecha más temprana para comenzar y terminar cada tarea
- Comenzando por la fecha de inicio del proyecto
- Estima la fecha de fin más optimista



Gestión de proyectos: principales conceptos y prácticas

Paso atrás

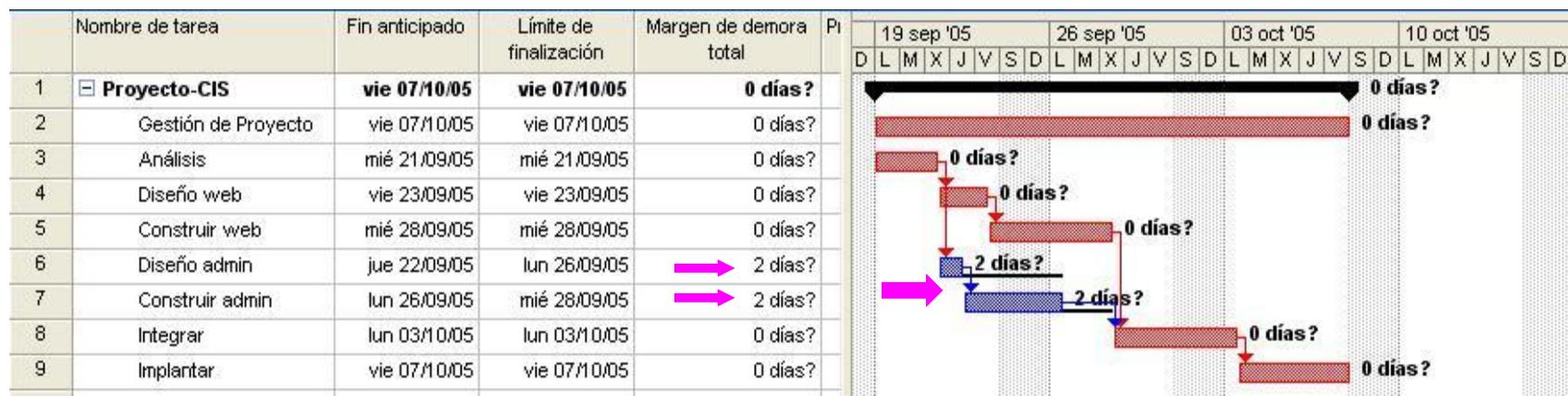
- Estimar cuales pueden ser las fechas mas retrasadas para el inicio y fin de cada tarea sin causar retrasos en el proyecto.
- Se puede estimar con la fecha de entrega calculada por “paso adelante”, o con la fecha de entrega deseada
 - Al calcular con la fecha de entrega por “paso adelante” se debe obtener a la misma fecha de inicio.
 - Al calcular con la fecha de entrega esperada se obtiene la fecha límite para comenzar el proyecto



Gestión de proyectos: principales conceptos y prácticas

Demora total

- Diferencia entre las fechas calculadas con “paso adelante” y “paso atrás” para cada tarea
- Retraso máximo para una tarea sin retrasar sin afectar a la fecha de entrega del proyecto
- La demora total se comparte entre las tareas en una cadena. Si se emplea en una tarea ya no queda disponible para otras.

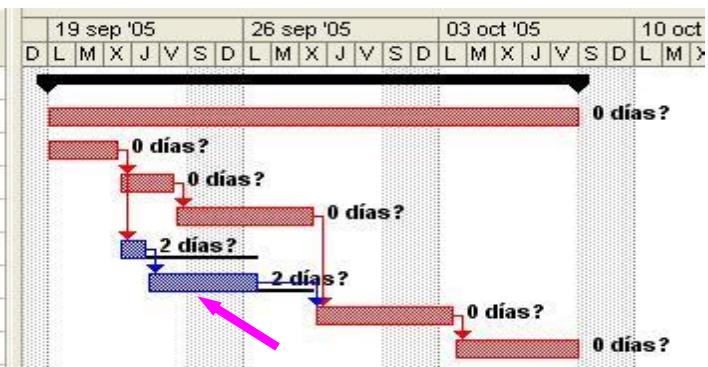


Gestión de proyectos: principales conceptos y prácticas

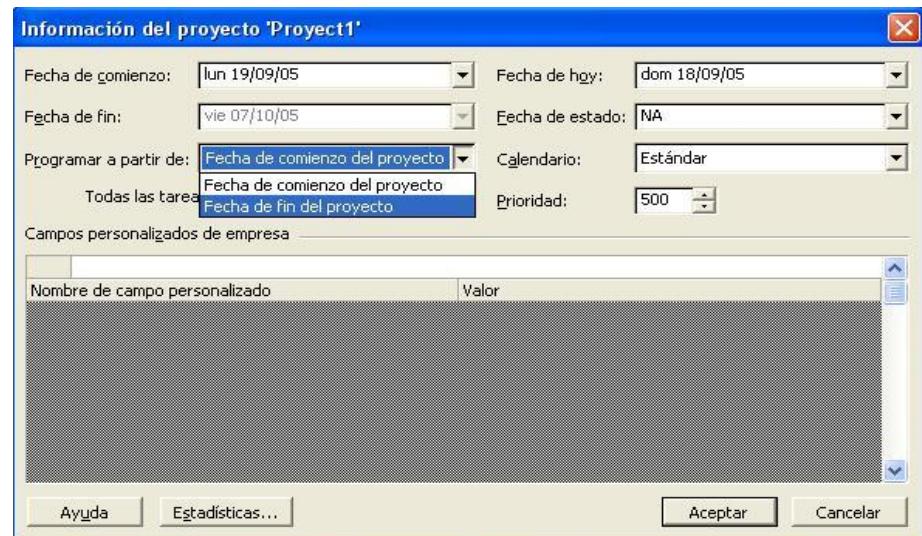
Demora permisible

Tiempo que puede retrasarse una tarea sin afectar a la agenda del proyecto

	Nombre de tarea	Fin anticipado	Límite de finalización	Demora permisible
1	■ Proyecto-CIS	vie 07/10/05	vie 07/10/05	0 días?
2	Gestión de Proyecto	vie 07/10/05	vie 07/10/05	0 días?
3	Análisis	mié 21/09/05	mié 21/09/05	0 días?
4	Diseño web	vie 23/09/05	vie 23/09/05	0 días?
5	Construir web	mié 28/09/05	mié 28/09/05	0 días?
6	Diseño admin	jue 22/09/05	lun 26/09/05	0 días?
7	Construir admin	lun 26/09/05	mié 28/09/05	2 días?
8	Integrar	lun 03/10/05	lun 03/10/05	0 días?
9	Implantar	vie 07/10/05	vie 07/10/05	0 días?



Algunos programas como MS Project
pueden hacer los cálculos de forma
automática



Gestión de proyectos: principales conceptos y prácticas

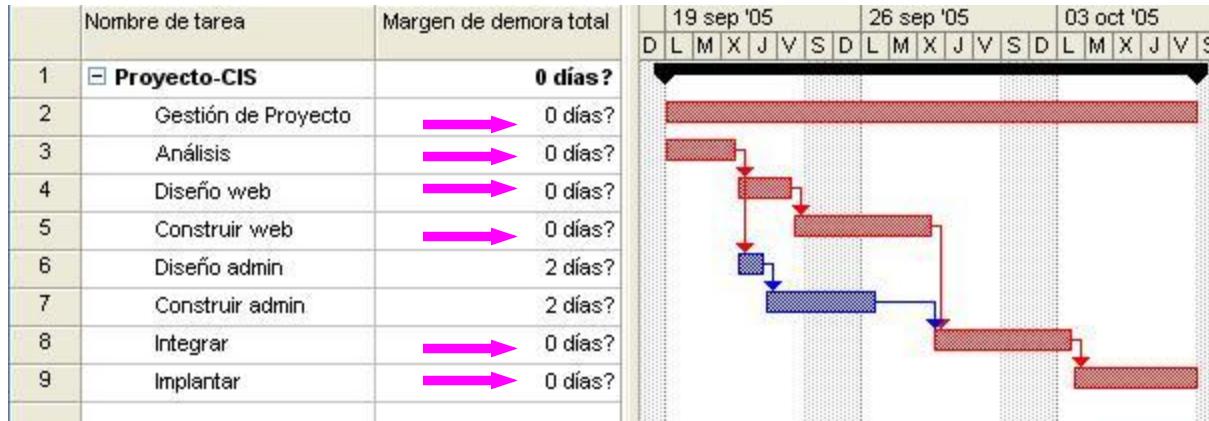
Ruta crítica

Es la ruta más larga en el plan del proyecto, y delimita la fecha de entrega más temprana posible



Actividades críticas

Actividades que están en la ruta crítica y que no tienen demora permisible. Sus retrasos afectan al proyecto

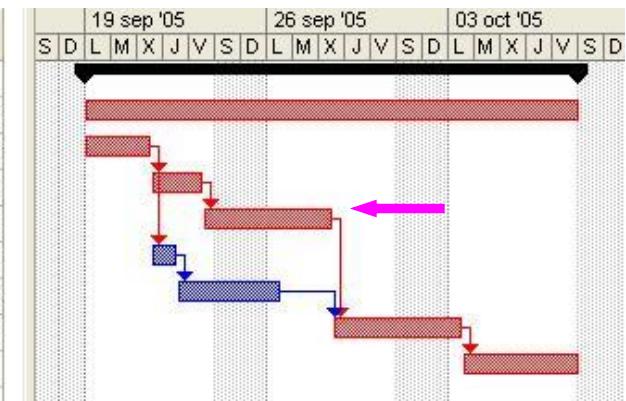


Gestión de proyectos: principales conceptos y prácticas

Optimización de agenda

1.- Dirigir el esfuerzo de trabajo sobre las actividades de la ruta crítica

	Nombre de tarea	Trabajo	Fin	Nombres de los recursos
1	Proyecto-CIS	344 horas	vie 07/10/05	
2	Gestión de Proyecto	120 horas	vie 07/10/05	Gestor
3	Análisis	40 horas	mié 21/09/05	Programador1;Programador2
4	Diseño web	16 horas	vie 23/09/05	Programador1
5	Construir web	24 horas	mié 28/09/05	Programador1
6	Diseño admin	8 horas	jue 22/09/05	Programador2
7	Construir admin	16 horas	lun 26/09/05	Programador2
8	Integrar	48 horas	lun 03/10/05	Programador2;Programador1
9	Implantar	72 horas	vie 07/10/05	Programador2;Programador1



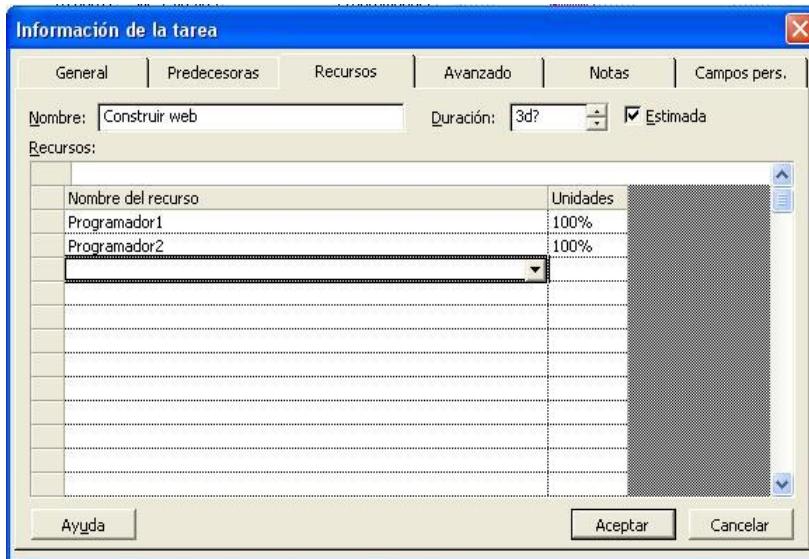
2.- Revisar la asignación de personas

	1	Nombre del recurso	Trabajo	Detalles	octubre				
					12/09	19/09	26/09	03/10	1
1		+ Gestor	120 horas	Trabajo			40h	40h	40h
2		+ Programador1	120 horas	Trabajo			40h	40h	40h
3		+ Programador2	104 horas	Trabajo			40h	24h	40h
				Trabajo					

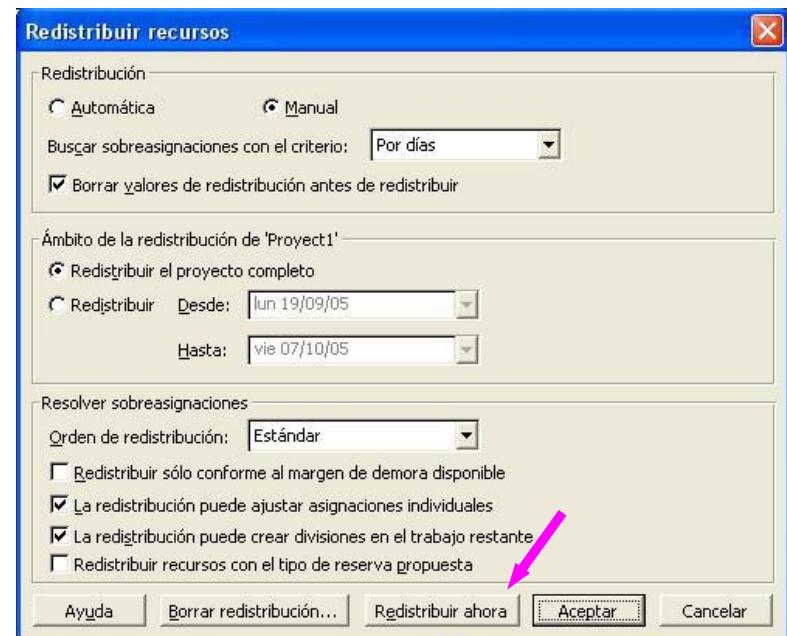
Gestión de proyectos: principales conceptos y prácticas

Optimización de agenda

3.- Modificar la asignación



4.- Redistribuir a las personas

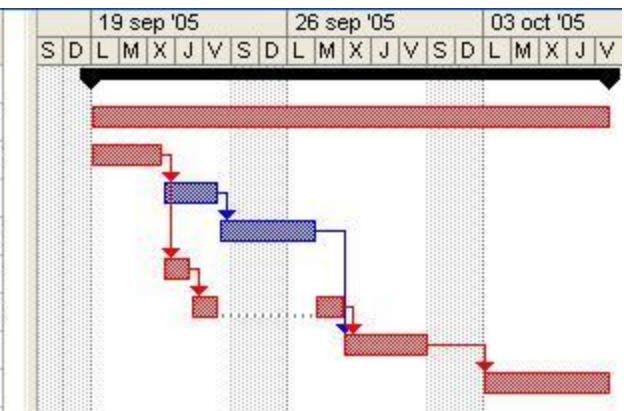


Gestión de proyectos: principales conceptos y prácticas

Optimización de agenda

Nueva ruta crítica

	Nombre de tarea	Trabajo	Fin	Nombres de los recursos
1	✉ Proyecto-CIS	340 horas	vie 07/10/05	
2	Gestión de Proyecto	116 horas	vie 07/10/05	Gestor
3	Análisis	40 horas	mié 21/09/05	Programador1;Programador2
4	Diseño web	16 horas	vie 23/09/05	Programador1
5	Construir web	24 horas	lun 26/09/05	Programador1;Programador2
6	Diseño admin	8 horas	jue 22/09/05	Programador2
7	Construir admin	16 horas	mar 27/09/05	Programador2
8	Integrar	48 horas	vie 30/09/05	Programador2;Programador1
9	Implantar	72 horas	vie 07/10/05	Programador2;Programador1



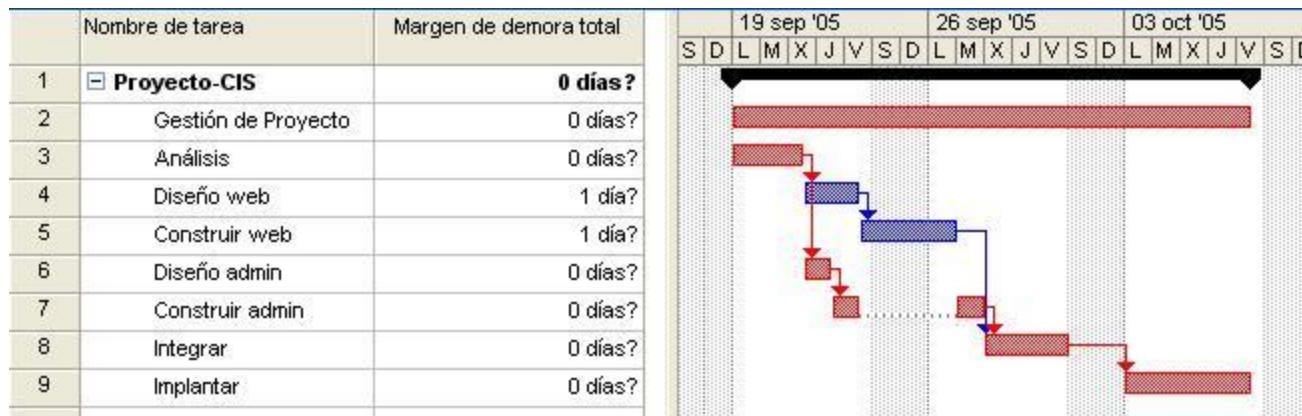
Asignación mas eficiente

	ℹ	Nombre del recurso	Trabajo	octubre		
				19/09	26/09	03/10
1		+ Gestor	116 horas		40h	40h
2		+ Programador1	108 horas		40h	32h
3		+ Programador2	116 horas		40h	36h

Gestión de proyectos: principales conceptos y prácticas

Optimización de agenda

Reducción de la demora total



Recomendaciones

- Duplicar la asignación de personas no significa "la mitad de tiempo"
- Menos tiempo de entrega no significa menor presupuesto
- Cada tarea debe tener un resultado cuantificable para comprobar que se ha realizado
- Usar el sentido común

GESTIÓN DE PROYECTOS PREDICTIVA



cc-by: [Betsy Fletcher](#)

Vídeo: Construcción Plaza Carso y Museo Soumaya (Mexico)



GESTIÓN DE PROYECTOS PREDICTIVA

Gestión de la integración del proyecto

- Desarrollo del plan de proyecto
- Ejecución del plan de proyecto
- Control integrado del cambio

Gestión del ámbito del proyecto

- Inicio
- Planificación del ámbito
- Definición del ámbito
- Verificación del ámbito
- Control de cambio del ámbito

Gestión de agenda

- Definición de la actividad
- Priorización de la actividad
- Estimación de tiempos
- Desarrollo de la agenda
- Control de la agenda

Gestión de costes

- Plan de recursos
- Estimación de costes
- Presupuesto
- Control de costes

Gestión de la calidad del proyecto

- Plan de calidad
- Seguimiento de la calidad
- Control de calidad

Gestión de los recursos humanos del proyecto

- Plan de organización
- Incorporación de personas
- Desarrollo del equipo

Gestión de las comunicaciones del proyecto

- Plan de comunicación
- Distribución de la información
- Informes de ejecución
- Cierre administrativo

Gestión de riesgos del proyecto

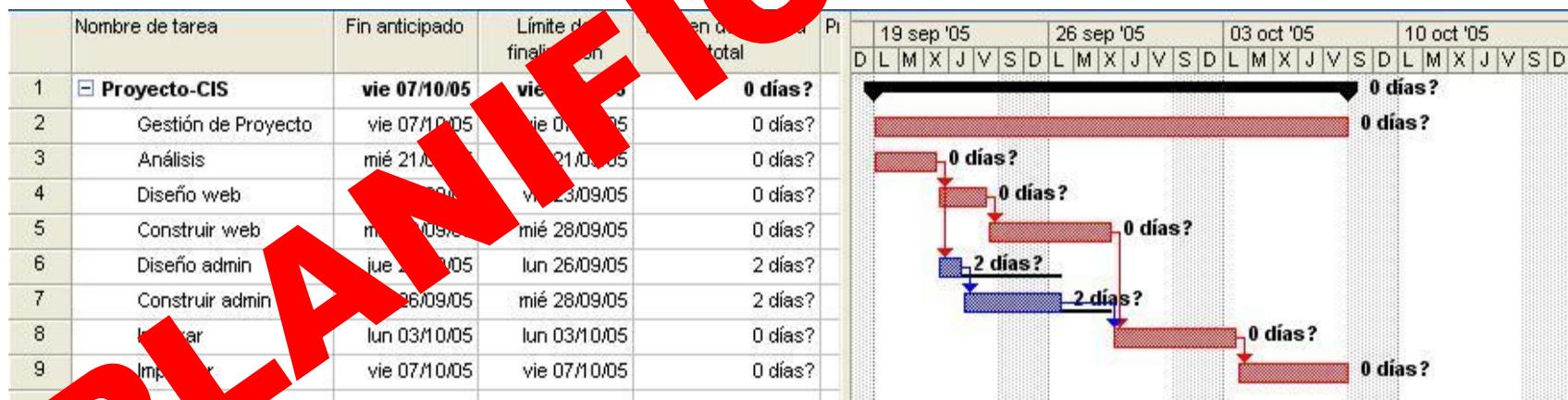
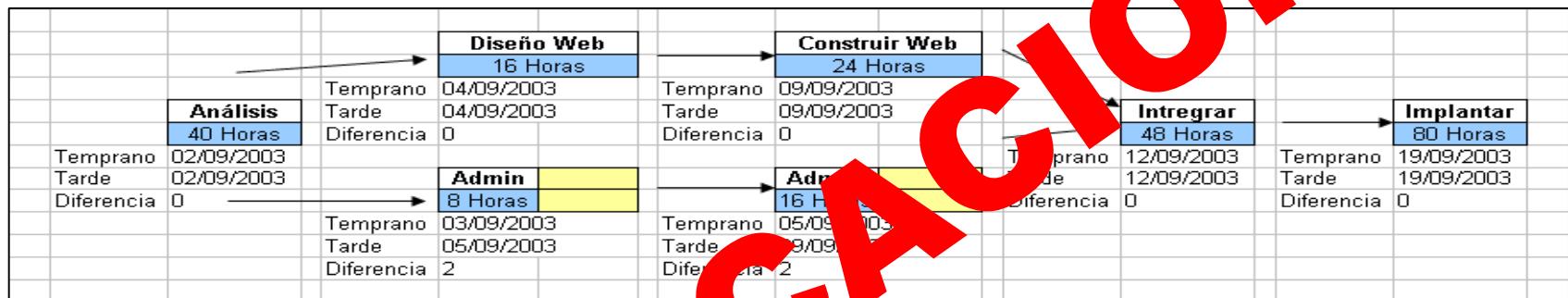
- Plan de riesgos
- Identificación de riesgos
- Análisis cuantitativo de riesgos
- Análisis cualitativo de riesgos
- Plan de exposición de riesgos
- Monitorización y control de riesgos

Gestión de compras

- Plan de necesidades
- Plan de compras
- Compras
- Selección de proveedores
- Contratación administrativa
- Cierre de contrato

GESTIÓN

GESTIÓN DE PROYECTOS PREDICTIVA

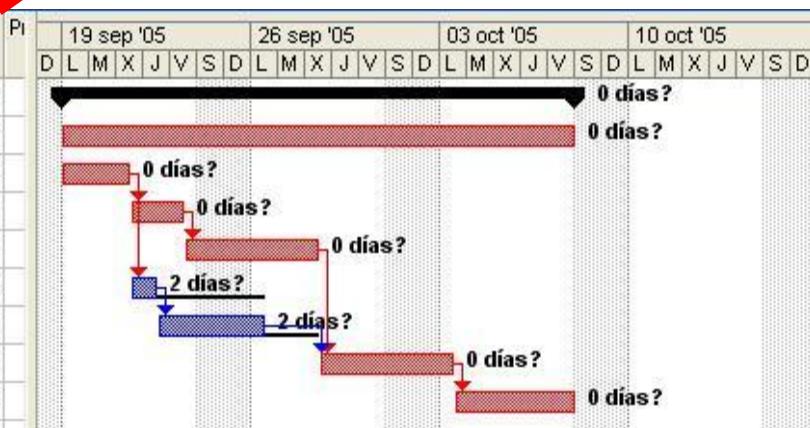


PLANIFICACIÓN

GESTIÓN DE PROYECTOS PREDICTIVA



	Nombre de tarea	Fin anticipado	Límite de finalización	Demora de tarea	Prioridad
1	Proyecto-CIS	vie 07/10/05	07/10/05	0 días?	
2	Gestión de Proyecto	vie 07/10/05	07/10/05	0 días?	
3	Análisis	mié 21/09/05	mié 21/09/05	0 días?	
4	Diseño web	viernes 23/09/05	23/09/05	0 días?	
5	Construir web	miércoles 28/09/05	28/09/05	0 días?	
6	Diseño admin	lunes 26/09/05	26/09/05	2 días?	
7	Construir admin	lun 28/09/05	28/09/05	2 días?	
8	Implantar	lun 03/10/05	03/10/05	0 días?	
9	Finalizar	vie 07/10/05	07/10/05	0 días?	



SEGUIMIENTO

GESTIÓN DE PROYECTOS PREDICTIVA

OBJETIVO



A tiempo

En costes

Calidad

GESTIÓN DE PROYECTOS CARACTERÍSTICAS

		Diseño Web			Construir Web			Integrar			Implantar		
		16 Horas			24 Horas			48 Horas			80 Horas		
		Temprano	04/09/2003	Tarde	04/09/2003	Diferencia	0	Temprano	09/09/2003	Tarde	09/09/2003	Diferencia	0
		Temprano	02/09/2003	Tarde	02/09/2003	Diferencia	0	Temprano	12/09/2003	Tarde	19/09/2003	Diferencia	0
		Admin	8 Horas	Admin	16 Horas	Admin	16 Horas	Admin	12/09/2003	Admin	19/09/2003	Admin	80 Horas
1	Projecto-CIS		vie 07/10/05										
2	Gestión de Proyecto		vie 07/10/05										
3	Análisis		mí 05/10/05										
4	Diseño web		vie 07/10/05		vie 23/09/05								
5	Construir web		vie 07/10/05		mié 28/09/05								
6	Diseño admin		vie 07/10/05		lun 26/09/05								
7	Construir admin		vie 07/10/05		mié 28/09/05								
8	Integrar		vie 07/10/05		lun 03/10/05								
9	Implantar		vie 07/10/05		viernes 10/10/05								

GESTIÓN DE PROYECTOS CARACTERÍSTICAS



**AGENDA Y COSTE
SON
RESULTADO DE LA
PLANIFICACIÓN**

CONCLUSIONES

Características de los proyectos

Obra única

Inicio y fin definidos

Objetivo de la gestión de proyectos

Realizar la obra definida, en la fecha prevista y por el coste estimado

Puntos de apoyo de la gestión de proyectos

Definir qué hacer

Planificar el trabajo

Gestionar el cumplimiento

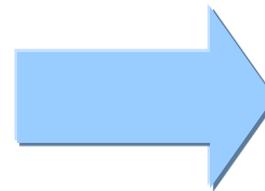
Necesita requisitos estables

La agenda y los costes son cálculos realizados sobre el plan del proyecto

Gestión de riesgos: principales conceptos y prácticas

GESTIÓN DE RIESGOS

- IDENTIFICACIÓN
- ANÁLISIS
- TRATAMIENTO



**Plan de gestión
de riesgos**

Gestión de riesgos: principales conceptos y prácticas

Principales estándares para gestión de riesgos

Estándar	Descripción
ISO 31000	Estándar publicado por ISO
British Standard BS 31100	Estándar publicado por el instituto británico de estandarización
COSO ERM	Marco para riesgos corporativos desarrollado por el “Comitee of Sponsoring Organizations of the Treadway Commission” (coso.org)
ISO IEC 73	Glosario estándar para la gestión de riesgos.

Gestión de riesgos: principales conceptos y prácticas

Principios de la gestión de riesgos

Principio	Descripción
Proporcional	La gestión de riesgos debe ser proporcional al nivel de criticidad del proyecto.
Exhaustiva	Para que resulte no debe dejar áreas generadoras de riesgos sin cubrir.
Institucionalizada	Las actividades de gestión de riesgos deben formar parte de las prácticas propias de la organización.
Dinámica	Las actividades de gestión de riesgos deben estar incluidas en ciclos institucionalizados de actualización y mejora continua.
Alineada	Las actividades de gestión de riesgos en los proyectos deben estar alineadas con la cultura y madurez de la organización.

Gestión de riesgos: principales conceptos y prácticas

Términos incluidos en ISO IEC 73

- COMMUNICATION & CONSULTATION
- CONSEQUENCE
- CONTROL
- ESTABLISHING THE CONTEXT
- EVENT
- EXPOSURE
- EXTERNAL CONTEXT
- FREQUENCY
- HAZARD
- INTERNAL CONTEXT
- LEVEL OF RISK
- LIKELIHOOD
- MONITORING
- PROBABILITY
- RESIDUAL RISK
- RESILIENCE
- REVIEW
- RISK
- RISK ACCEPTANCE
- RISK AGGREGATION
- RISK ANALYSIS
- RISK APPETITE
- RISK ASSESSMENT
- RISK ATTITUDE
- RISK AVERSION
- RISK AVOIDANCE
- RISK CRITERIA
- RISK DESCRIPTION
- RISK EVALUATION
- RISK FINANCING
- RISK IDENTIFICATION
- RISK MANAGEMENT
- RISK MANAGEMENT AUDIT
- RISK MANAGEMENT FRAMEWORK
- RISK MANAGEMENT PLAN
- RISK MANAGEMENT POLICY
- RISK MANAGEMENT PROCESS
- RISK MATRIX
- RISK OWNER
- RISK PERCEPTION
- RISK PROFILE
- RISK REGISTER
- RISK REPORTING
- RISK RETENTION
- RISK SHARING
- RISK SOURCE
- RISK TOLERANCE
- RISK TREATMENT
- STAKEHOLDER
- VULNERABILITY

Gestión de riesgos: principales conceptos y prácticas

Gestión de riesgos



Gestión de riesgos: principales conceptos y prácticas

Análisis clasificación

■ MÉTODOS CUALITATIVOS

Los más utilizados cuando:

- El nivel de riesgo es bajo y no justifica el esfuerzo de realizar análisis completos.
- Los datos numéricos (análisis cuantitativos) resultan inapropiados.

Algunos métodos cualitativos: Tormenta de ideas, "what if", cuestionarios y entrevistas estructuradas, juicio de expertos (Técnica Delphi), checklists...

■ MÉTODOS CUANTITATIVOS

Permiten asignar valores objetivos de probabilidad y ocurrencia para realizar análisis de probabilidades, consecuencias o simulaciones computacionales.

■ MÉTODOS SEMICUANTITATIVOS

Utilizan rangos de estimación como "alto" "medio" "bajo" en una escala apropiada para estimar el nivel de riesgo o sus consecuencias.

Gestión de riesgos: principales conceptos y prácticas

Identificación

Método sistemático y organizado para descubrir los riesgos
Pueden identificarse directa o indirectamente.

DIRECTA: Identificando los orígenes (causas) que pueden ser:

- **Programación:** agendas impuestas, excesivas restricciones contractuales, riesgos políticos...
- **Requisitos:** inconsistencias, TBD's, crecientes...
- **Técnicos:** requisitos de rendimiento, seguridad, plataforma tecnológica
- **Calidad:** deficiencias en las prácticas de desarrollo
- **Logísticos:** mantenibilidad, operación, disponibilidad... (el impacto se produce cuando el sistema está en uso.)
- **Despliegue:** integración, instalación, distribución

INDIRECTA: Identificando el impacto y buscando las causas. Las áreas de impacto son:

- **Agenda**
- **Coste**
- **Calidad**
- **Operación**

Gestión de riesgos: principales conceptos y prácticas

Identificación: ejemplos

Riesgos de costes

Los principales factores que influyen en los riesgos de costes son:

- **Incertidumbre en los requisitos**
- **Estimación incorrecta de costes**
- **Requisitos crecientes**
- **Presión de agendas**
- **Costes irrazonables**

Riesgos de requisitos

Las principales causas de riesgo por los requisitos son:

- **Requisitos incorrectos**
- **Requisitos incompletos**
- **Requisitos inconsistentes**
- **Requisitos de dificultad imprevista**
- **Requisitos imposibles**
- **Requisitos ambiguos**
- **Volatilidad**

Gestión de riesgos: principales conceptos y prácticas

Análisis

Examen de los riesgos para determinar sus dos características principales:

- Probabilidad
- Impacto

El análisis de los riesgos es una tarea de ejecución cíclica que debe realizarse y revisarse periódicamente de forma adecuada a las características del proyecto.

El resultado del análisis se plasma en un registro de los riesgos con la identificación de su descripción y características, con apoyo de herramientas para su consulta, revisión, priorización, etc.

Gestión de riesgos: principales conceptos y prácticas

Gestión de riesgos: probabilidad e impacto



Gestión de riesgos: principales conceptos y prácticas

Tratamiento

El tratamiento de los riesgos es el tercer paso del proceso de gestión de riesgos y comprende la implementación de métodos y técnicas para reducir el impacto o la probabilidad.

Las técnicas se clasifican en 5 categorías en función de su finalidad:

1.- Evitar.

No proceder con esa actividad o escoger una alternativa. Para riesgos con ratio probabilidad / impacto no admisible.

2.- Prevenir.

Para reducir los parámetros de probabilidad y/o impacto. Actividades más utilizadas:
Formación / información, mantenimiento preventivo, disminución del nivel de exposición,

3.- Aceptación o asunción del riesgo.

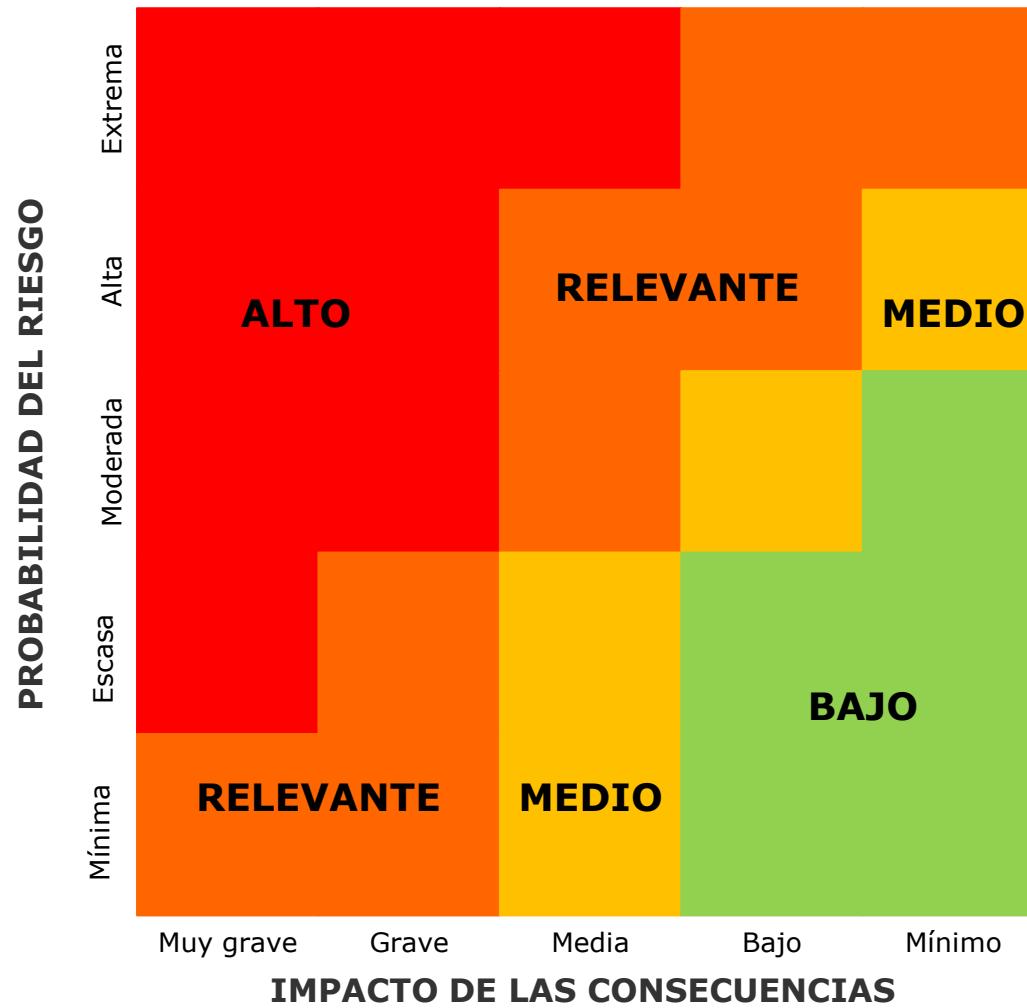
Puede resultar suficiente para riesgos de escaso impacto y probabilidad no usar medidas preventivas, si se dispone de reactivas en el caso de que se produzca, y su uso estadístico global supone menor coste global que la prevención.

4.- Proteger. Medidas que reducen

- La probabilidad (Discos redundantes, sistemas eléctricos redundantes, SAI...)
- Reducen las consecuencias (Backup, extintor...)

5.- Transferencia del riesgo. Se traspasa el riesgo a otra compañía, departamento o ámbito. ejemplo: contratación de un seguro, outsourcing

Gestión de riesgos: principales conceptos y prácticas



PRÁCTICA: ANÁLISIS DE RIESGOS



cc-by: [LizMarie](#)

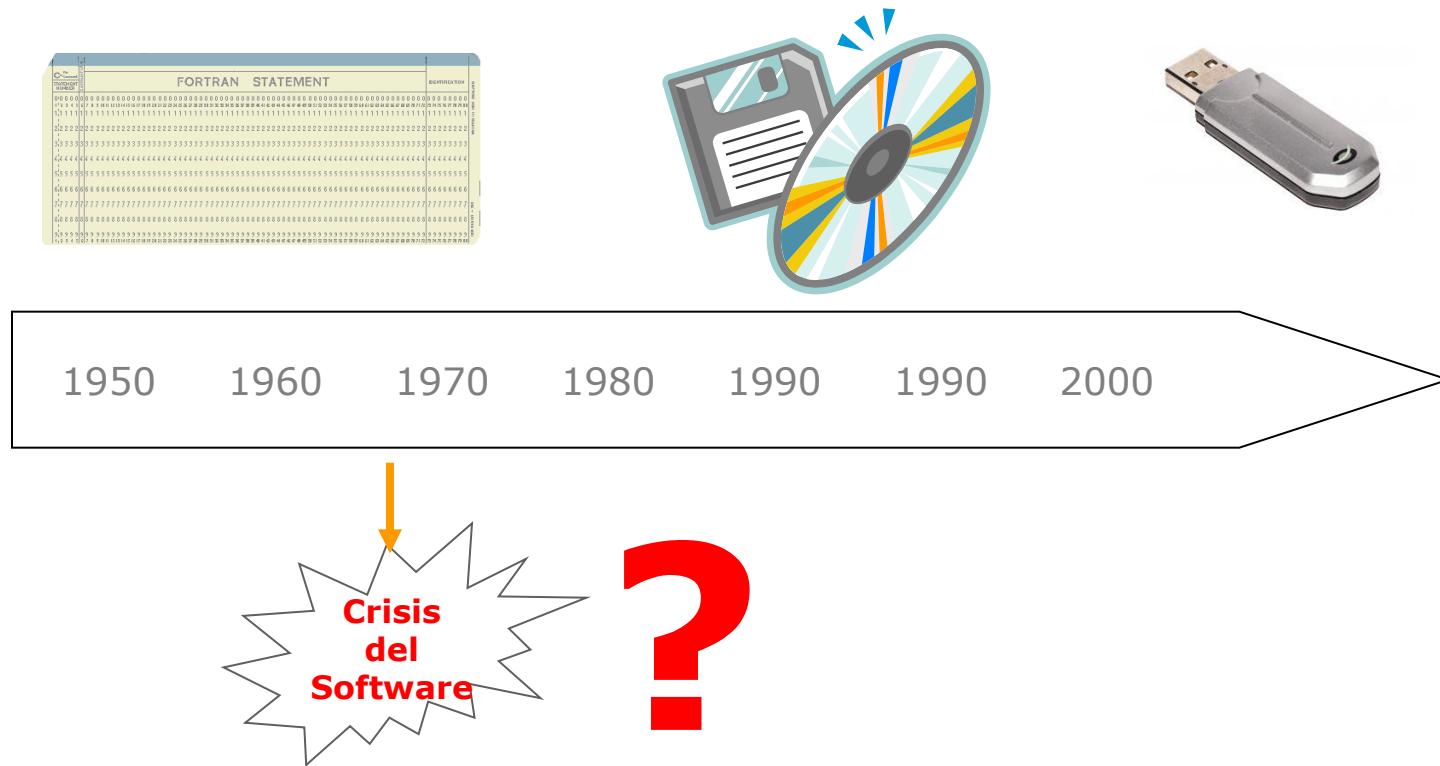


Producción basada en procesos

1.0



EL PROBLEMA



LAS PRIMERAS SOLUCIÓN: PRODUCCIÓN BASADA EN PROCESOS



Gestión de
proyectos
predictiva



Producción
basada en
procesos

ESCENARIO DE LOS 80: PRODUCCIÓN BASADA EN PROCESOS

**LA CALIDAD DEL RESULTADO
DEPENDE DE LA CALIDAD DE
LOS PROCESOS**

TQM - CMM - CMMI
Jurán / Humphrey



ESCENARIO DE LOS 80: PRODUCCIÓN BASADA EN PROCESOS

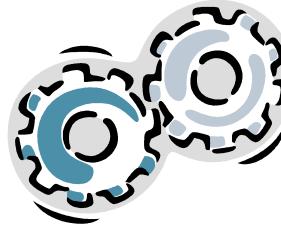


Eficiencia



Calidad

PROCESOS



Repetibilidad

PRÁCTICA: PRODUCCIÓN BASADA EN PROCESOS



cc-by: [LizMarie](#)

ESCENARIO DE LOS 80: PRODUCCIÓN BASADA EN PROCESOS

**LA CALIDAD DEL RESULTADO
DEPENDE DE LA CALIDAD DE
LOS PROCESOS**

TQM - CMM - CMMI
Jurán / Humphrey



ESCENARIO DE LOS 80: PRODUCCIÓN BASADA EN PROCESOS

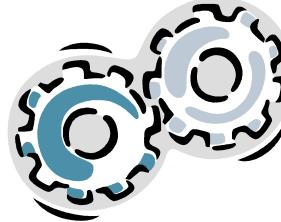


Eficiencia



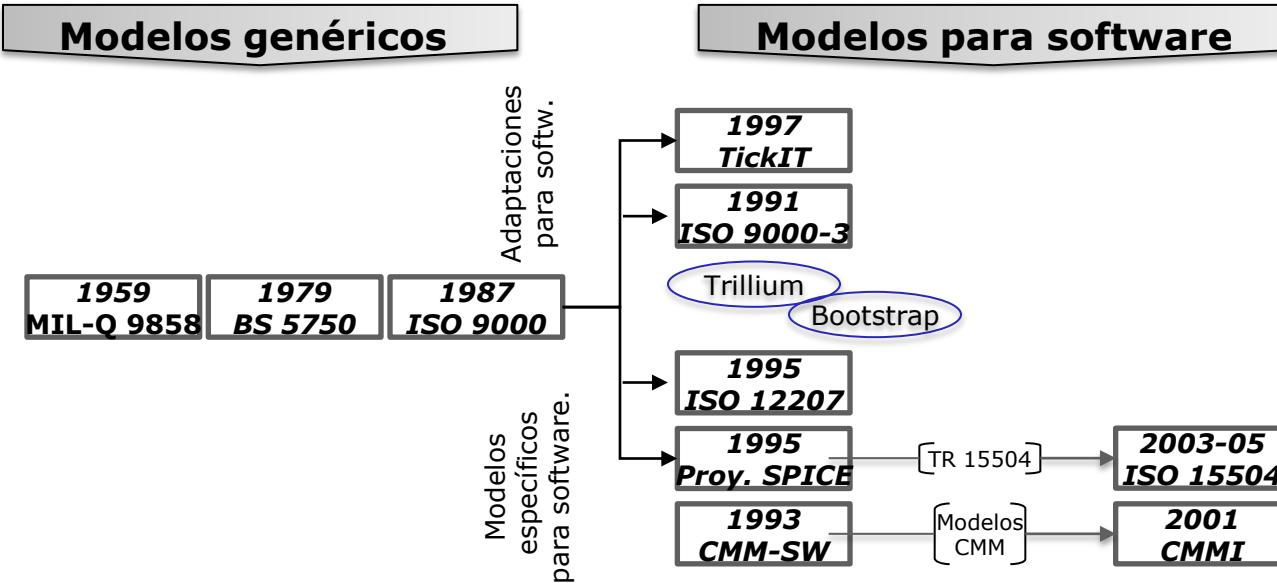
Calidad

PROCESOS



Repetibilidad

MODELOS DE PROCESOS



REFERENTES DE LA PRIMERA SOLUCIÓN

Ingeniería
del software

Ingeniería de
procesos

Gestión
Predictiva



ISO 9000-3
ISO/IEC 12207
ISO/IEC 15504



ESTÁNDARES PARA LA
INGENIERÍA DEL
SOFTWARE

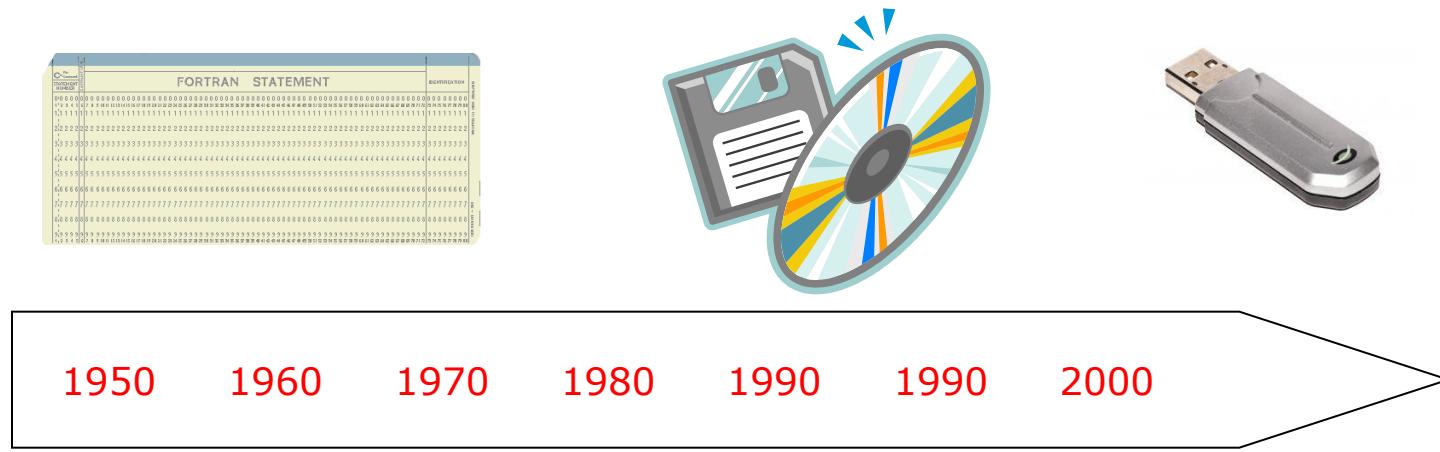


CMM-SW
CMMI



PMBOK

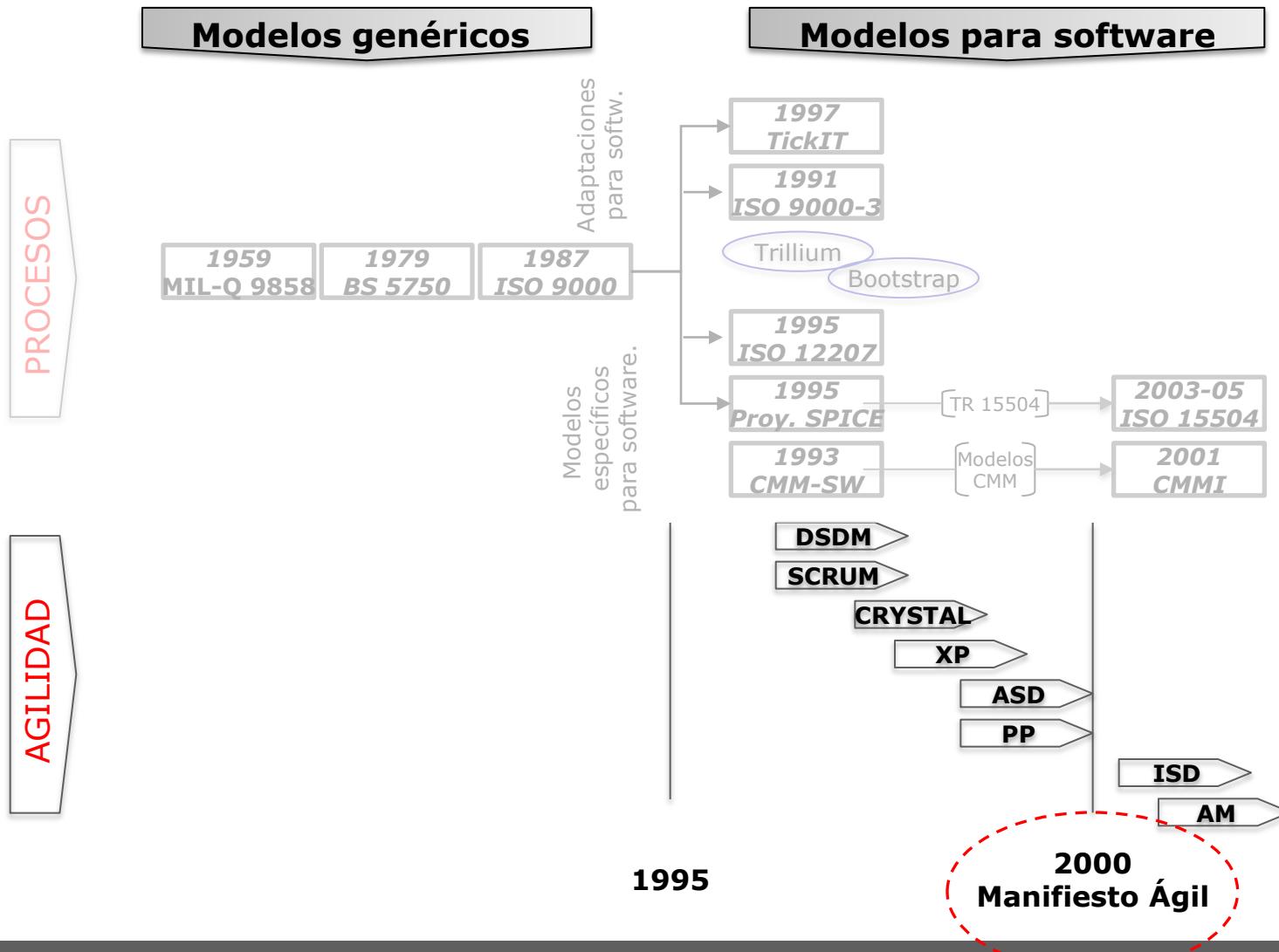
LA PRIMERA SOLUCIÓN A LA CRISIS DEL SOFTWARE



Ingeniería del software – Ingeniería de procesos
Gestión de proyectos predictiva

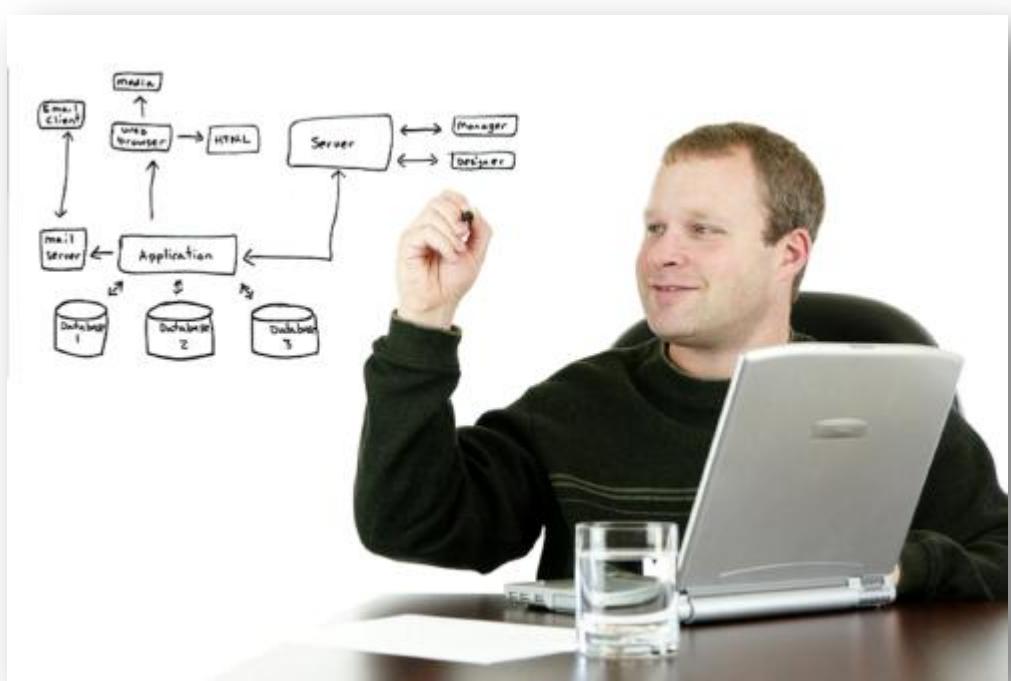


POSTERIOR ANTÍTESIS A LOS MODELOS DE PROCESOS: AGILIDAD



Ingeniería de procesos de software

1.0



PROCESOS: CONCEPTOS GENERALES

■ ¿Qué es un proceso?

- **Conjunto repetitivo ...**
- De actividades **interrelacionadas** ...
- Que se realizan **sistemáticamente** ...
- Mediante las cuales ...
- Un **entrada** se **convierte** en una **salida** o resultado, ... después de **añadirle valor**

Ingeniería de procesos de software: definición

■ Finalidad de los procesos de Ingeniería del Software

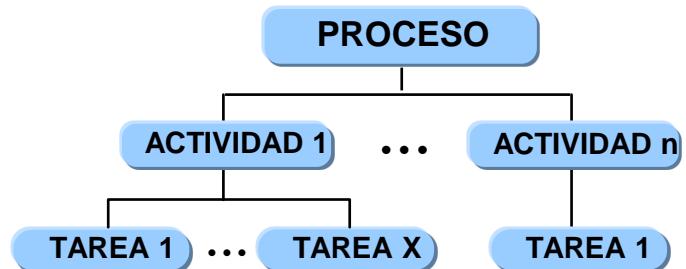
- Facilitar el entendimiento y comunicación
- Dar soporte a la gestión y mejora de procesos
- Proporcionar un soporte / guía "automatizado"

■ Los procesos deben ser adecuados a la organización y tipo de proyectos

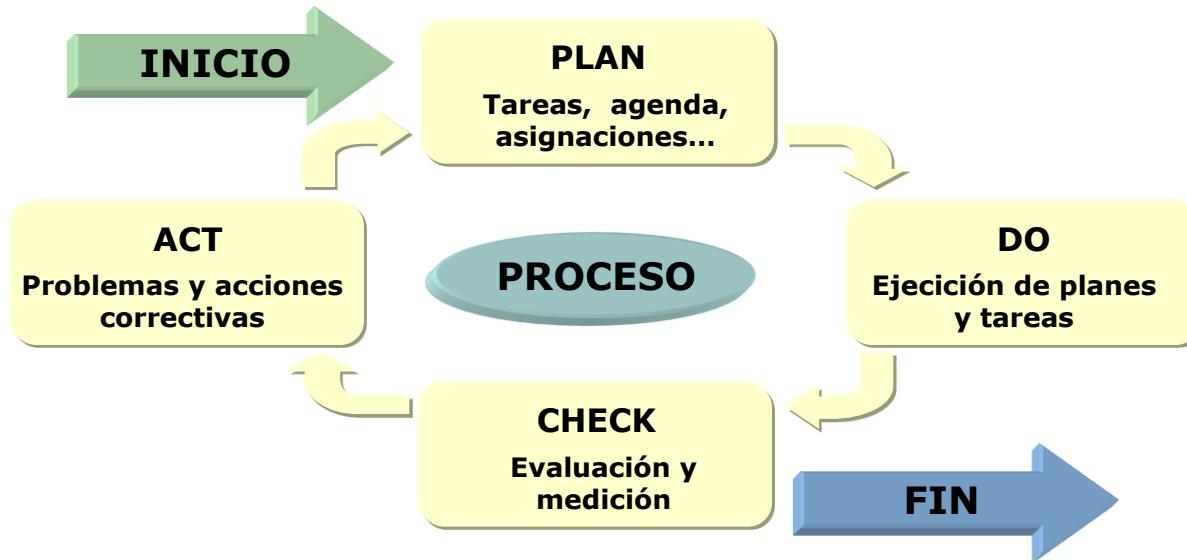
- Naturaleza del trabajo (Mto. / Desarrollo)
- Dominio de aplicación
- Estructura del proceso de entrega (incremental, evolutivo, cascada...)
- Madurez de la organización

Ejemplo: estructura de procesos en ISO 12207

- Un proceso está compuesto por actividades.
- Una actividad está compuesta de tareas.



- La descomposición del proceso en actividades y tareas se realiza sobre el concepto de ciclo de mejora PDCA "Plan – Do – Chek – Act" (Planificación, ejecución, medición y mejora)



Modelos de documentación de procesos

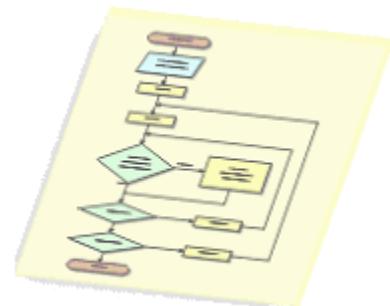
Hay diferentes **MODELOS** para definir y documentar procesos:

- Diferentes **niveles de abstracción** - Diferentes **tipos de definición**
- Modelos de **marco del ciclo de vida del software**
Los más comunes son: evolutivo / incremental, cascada, prototipado, prototipado evolutivo, espiral, software reutilizable.
- Modelos de **proceso del ciclo de vida del software**
Las principales referencias son:
 - **ISO/IEC 12207 IT – Software Life Cycle Processes**
 - **ISO/IEC 15504 IT – Software Process Assesment**

Notaciones

Lenguajes de modelado gráfico que sirven para representar a los procesos.

- **Diagramas de flujo:** Técnica para especificar los detalles de un proceso en términos de actividades, tareas, entradas, salidas, condiciones.
- **IDEF0:** Estándar para desarrollar representaciones gráficas estructuradas de un sistema o entorno empresarial. Permite la construcción de modelos que comprenden funciones del sistema (actividades, acciones, operaciones, procesos), relaciones y datos.
- **UML...**



Proceso de ingeniería del software

■ Niveles del proceso de Ingeniería del Software

El proceso de ingeniería del software puede examinarse en **2 niveles**:

- **Actividades técnicas y de gestión** ejecutadas durante la adquisición, desarrollo, mantenimiento y retirada

- Meta-nivel concerniente a la **definición, implementación, medición y mejora de los procesos**



Ingeniería de procesos del software

Ingeniería de procesos del software (del ciclo de vida del software)

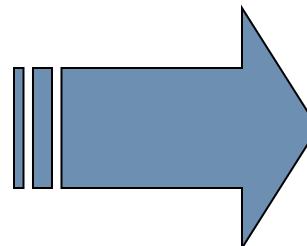
■ Modelos más relevantes de ingeniería de procesos

PDCA

IDEALSM

ISO 15504, Parte 7

ISO 9004:2000



- Modelos genéricos
- Basados en ciclos de mejora continua
- Tienen similitudes

Ingeniería de procesos del software (del ciclo de vida del software)

■ PDCA

PLAN

- Identificar el problema
- Analizar el problema

DO

- Desarrollar soluciones
- Implementar una solución

CHECK

- Evaluar los resultados

ACT

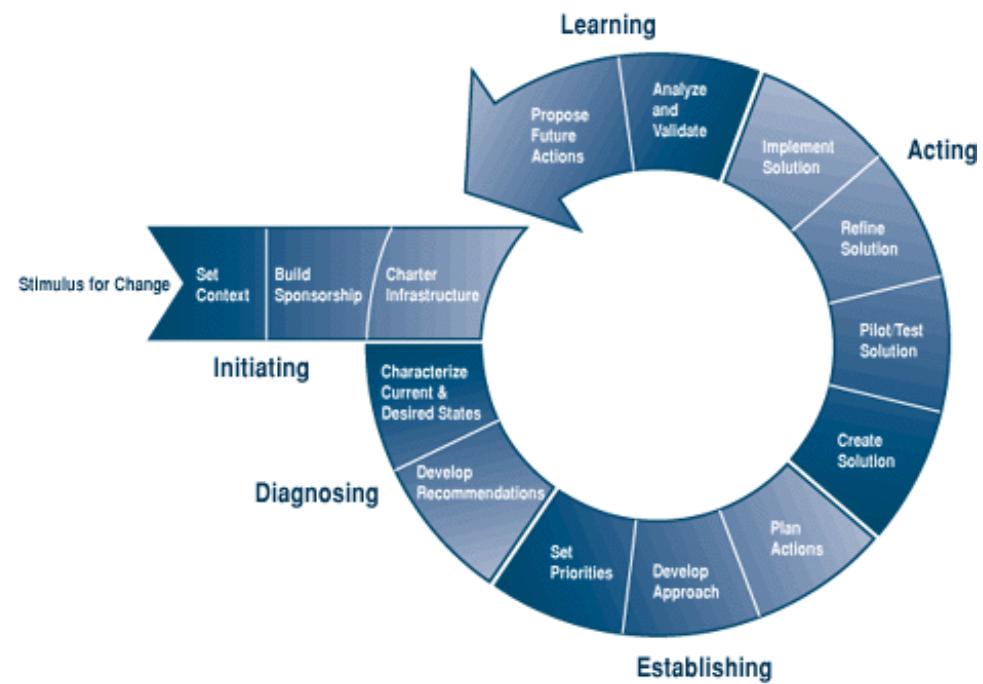
- Determinar los siguientes pasos



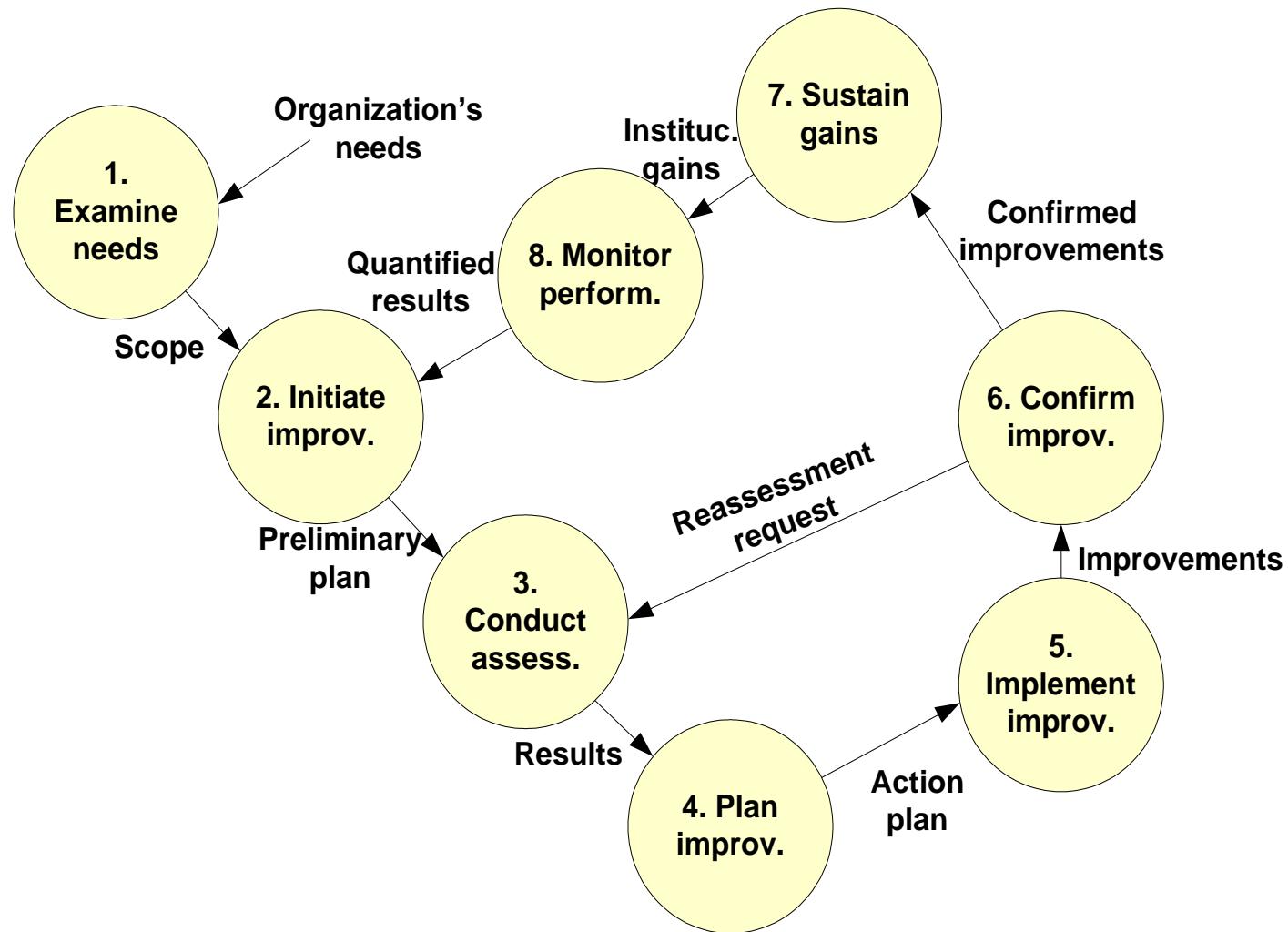
Ingeniería de procesos del software (del ciclo de vida del software)

■ IDEAL

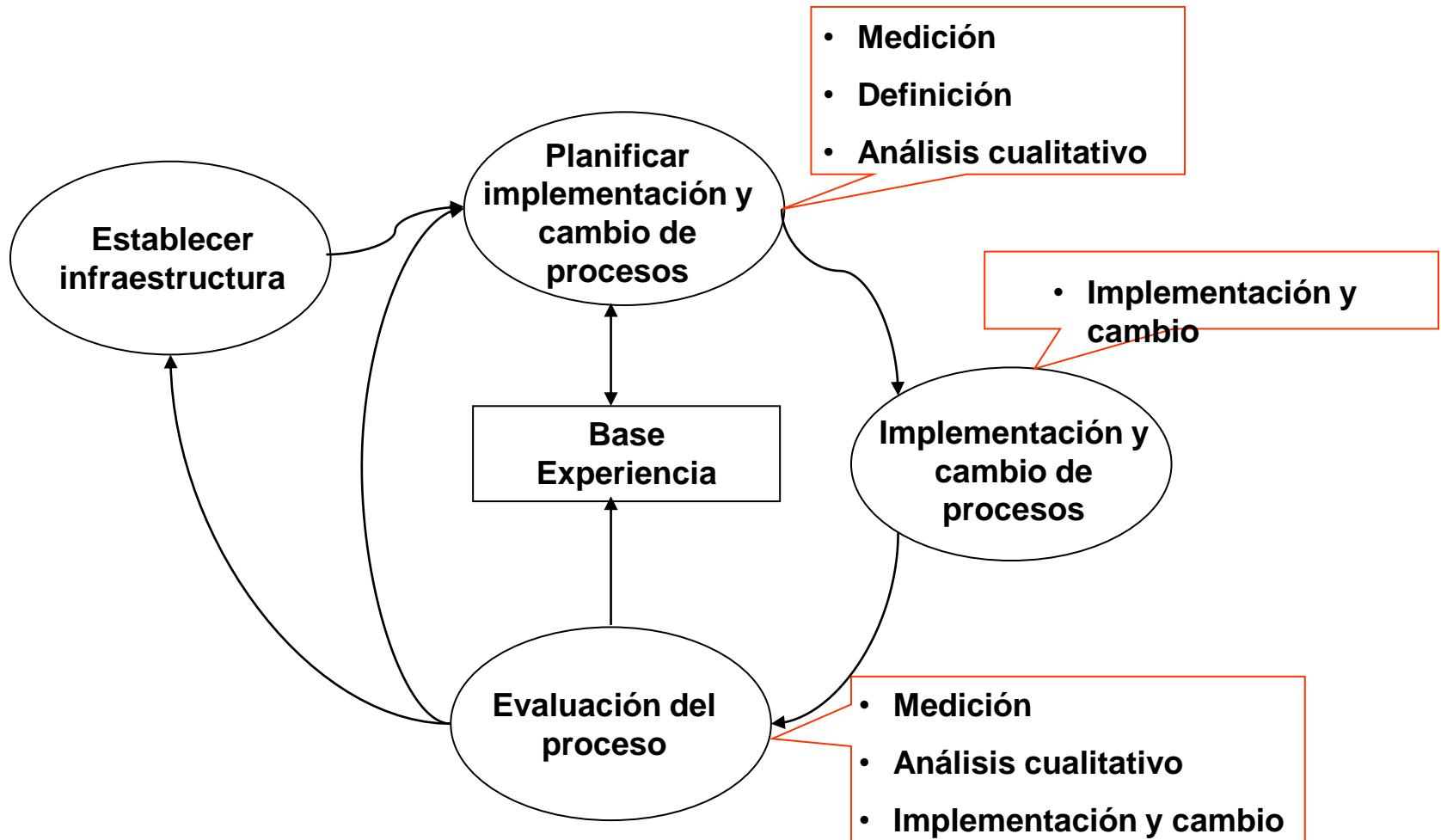
- Fase Inicial
- Fase de Diagnóstico
- Fase de establecimiento
- Fase de acción
- Fase de aprendizaje



ISO / IEC 15504



PATRONES COMUNES



INGENIERÍA DE PROCESOS: Patrones comunes



DEFINICIÓN



MEDICIÓN



ANÁLISIS



IMPLEMENTACIÓN Y CAMBIO

Medición

Obtener, analizar e interpretar información **CUANTITATIVA** para:

- **CARACTERIZAR**

Entender el proceso actual, entorno, etc.

Disponer de información de la situación anterior al cambio

- **EVALUAR**

Determinar la consecución de objetivos

Identificar fortalezas y debilidades del proceso

- **PREDECIR**

Entender las relaciones entre procesos y productos

Establecer objetivos alcanzables de calidad, costes y agendas

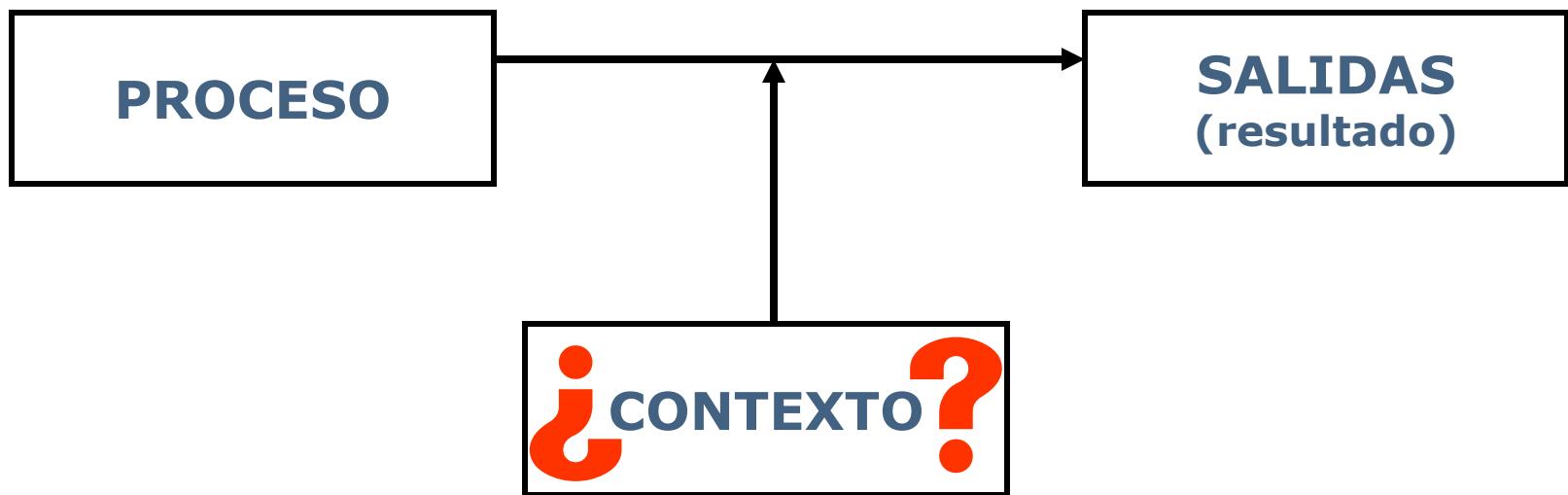
- **MEJORAR**

Identificar causas y oportunidades de mejora

Seguir el rendimiento de los cambios y compararlo con líneas base

Medición

Se puede medir la calidad del **proceso** (en si mismo) o la calidad de sus **salidas**.



Medición

■ Qué medir

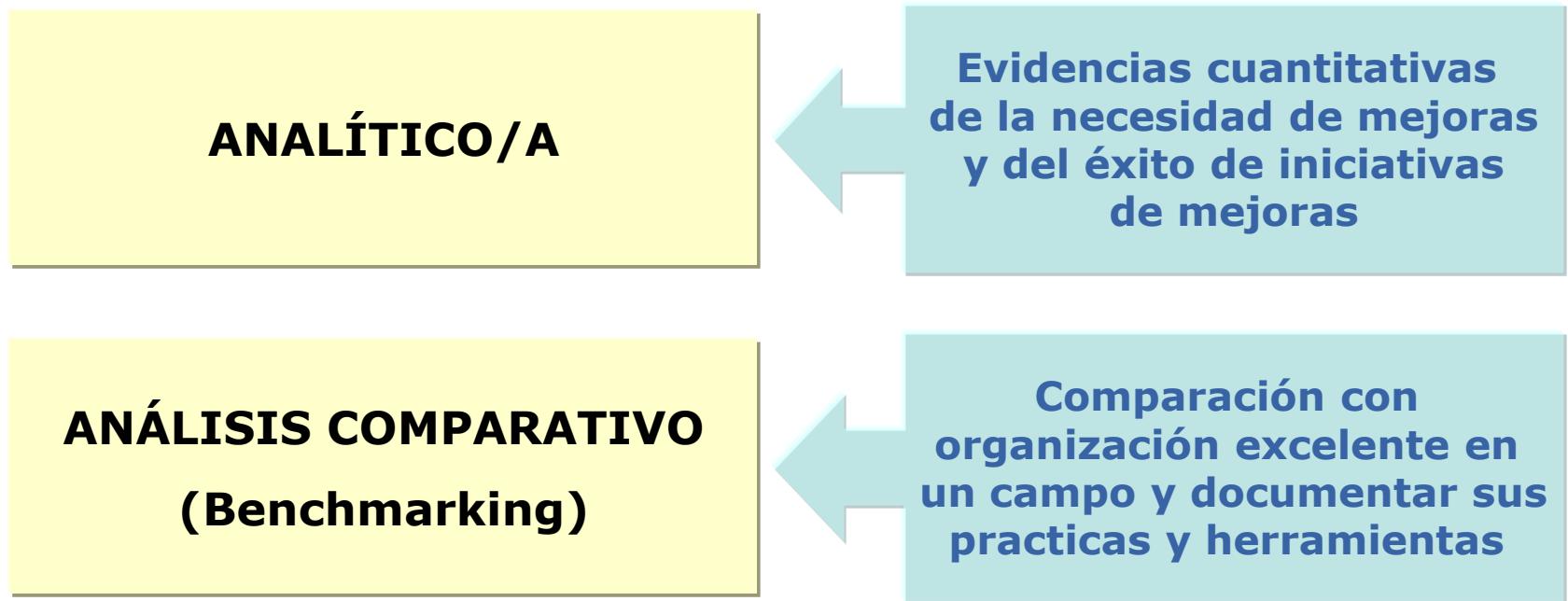
La forma de identificar qué medir y cómo hacerlo más eficiente es basándose en los objetivos del proceso (goal-oriented / goal driven)

- 1) Empezar por especificar las **necesidades de información**
- 2) Después, identificar las **medidas** que satisfacen esas necesidades

- **Fiabilidad:** Margen de error de la medición
- **Validez:** Capacidad de medir realmente lo que creemos que mide

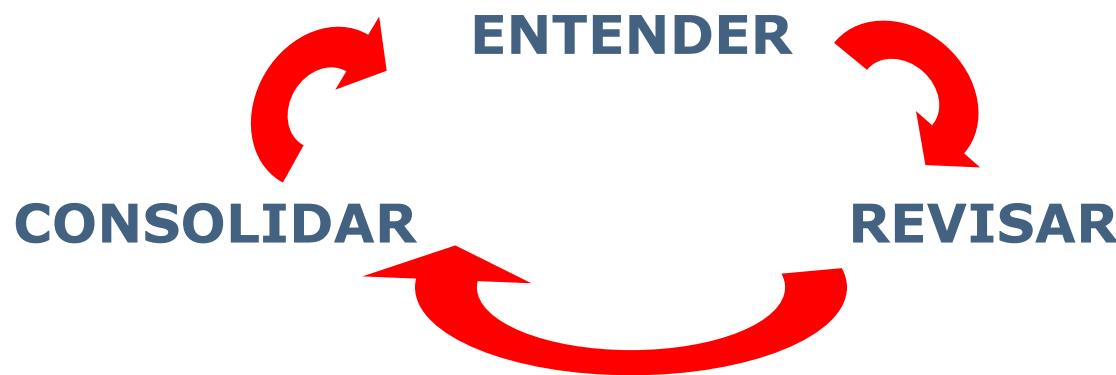
Medición

■ Paradigmas



Medición

El modelo **ANALÍTICO** consiste en:



Ejemplos:

- Estudios experimentales y de observación
- Simulación de procesos
- Clasificación ortogonal de defectos
- Control estadístico de procesos

Medición

- El modelo **ANÁLISIS COMPARATIVO** consiste en medir:
 - La **madurez** de una organización
 - o la **capacidad** de sus procesos
- Los **modelos de evaluación de procesos** recogen lo que se consideran “**best practices**”
 - SW-CMM y CMMI → CBA IPI y SCAMPI
 - ISO 15504 → ISO 15504, Part 8
 - ISO 9001
- **2 arquitecturas** generales con distintas asunciones en cuanto al orden en el que medir los procesos: **continua** y **escalonada**

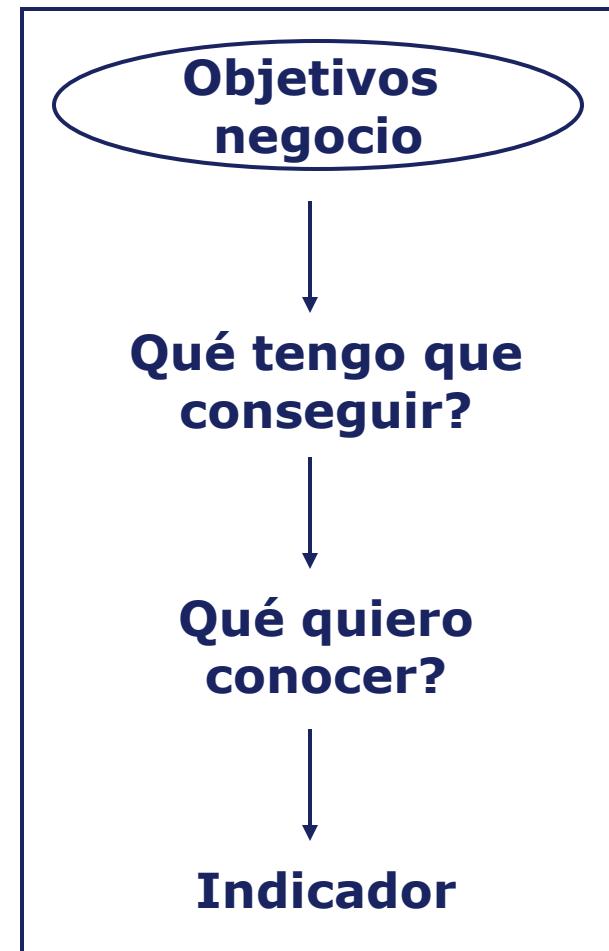
Medición Orientada a Objetivos (GQM) (Goal Question Metric)

OBJETIVO	Cada métrica debe estar dirigida a cubrir / medir un objetivo. La idea es que debemos tener buenas razones para recoger datos.
REQUISITO	Cada objetivo debe contestarse con uno o varios requisitos (preguntas). El requisito debe establecerse de forma que la métrica pueda responderlo claramente.
MÉTRICA INDICADOR	El indicador debe ser una entidad cuantitativa que da respuesta a un requisito específico.

Medición Orientada a Objetivos (GQM) (Goal Question Metric)

Resumen de los **pasos de GQM**

1. Establecer los **objetivos** / expectativas de la colección de datos
2. Desarrollar una lista de **requisitos** / criterios de interés
3. Establecer los **indicadores**
4. Diseñar los medios para recoger los datos
5. Recoger y validar los datos
6. Analizar los datos



Medición Orientada a Objetivos (GQM) (Goal Question Metric)

Ayuda para identificar requisitos: listar entidades y cuestiones relacionadas con los objetivos.

Entidad 1	<ul style="list-style-type: none">•••
Entidad 2	<ul style="list-style-type: none">•••
...	

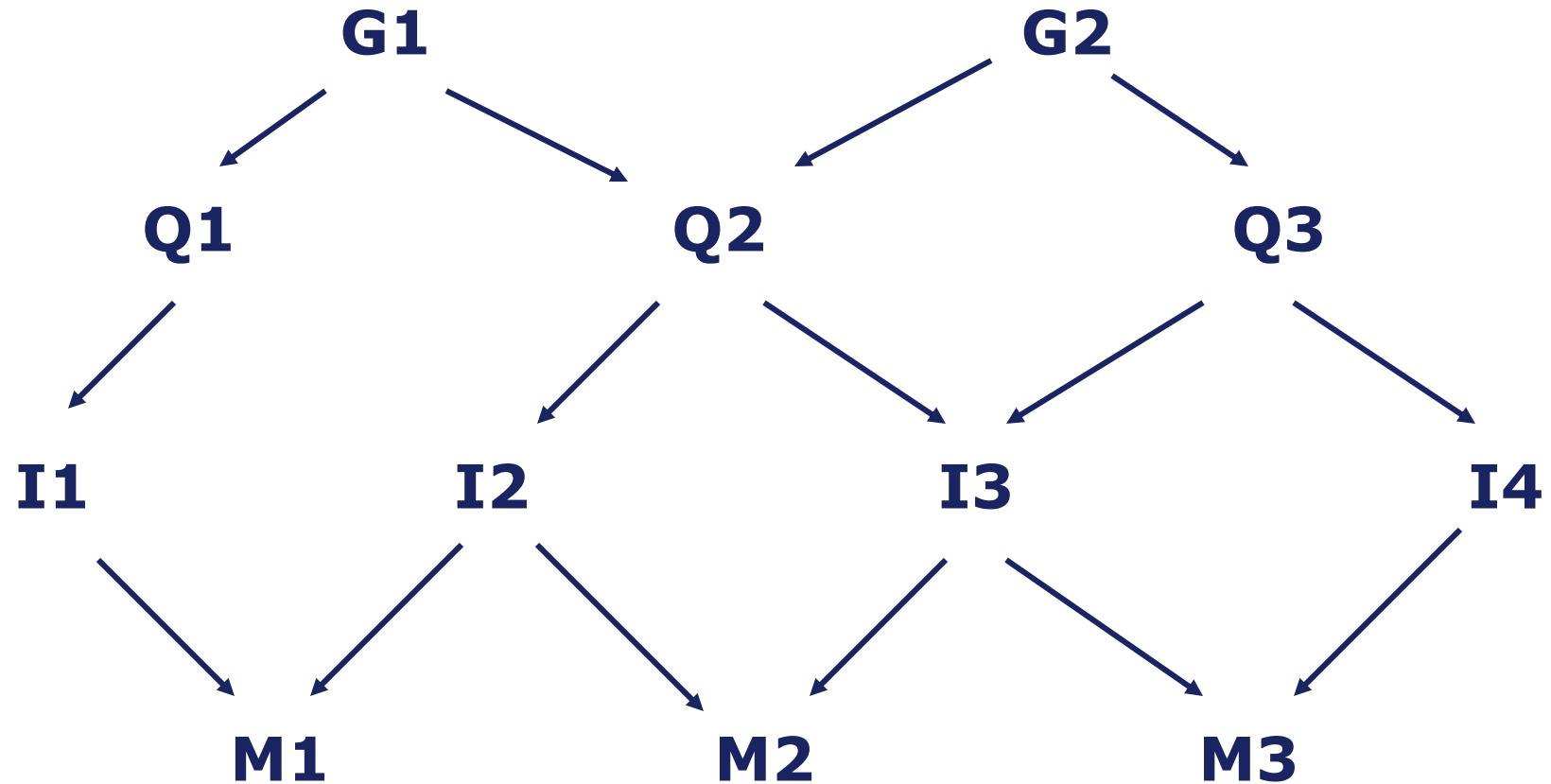
Descartar según sean comprensibles, cuantificables, conscientes, coherentes.

Medición Orientada a Objetivos (GQM) (Goal Question Metric)

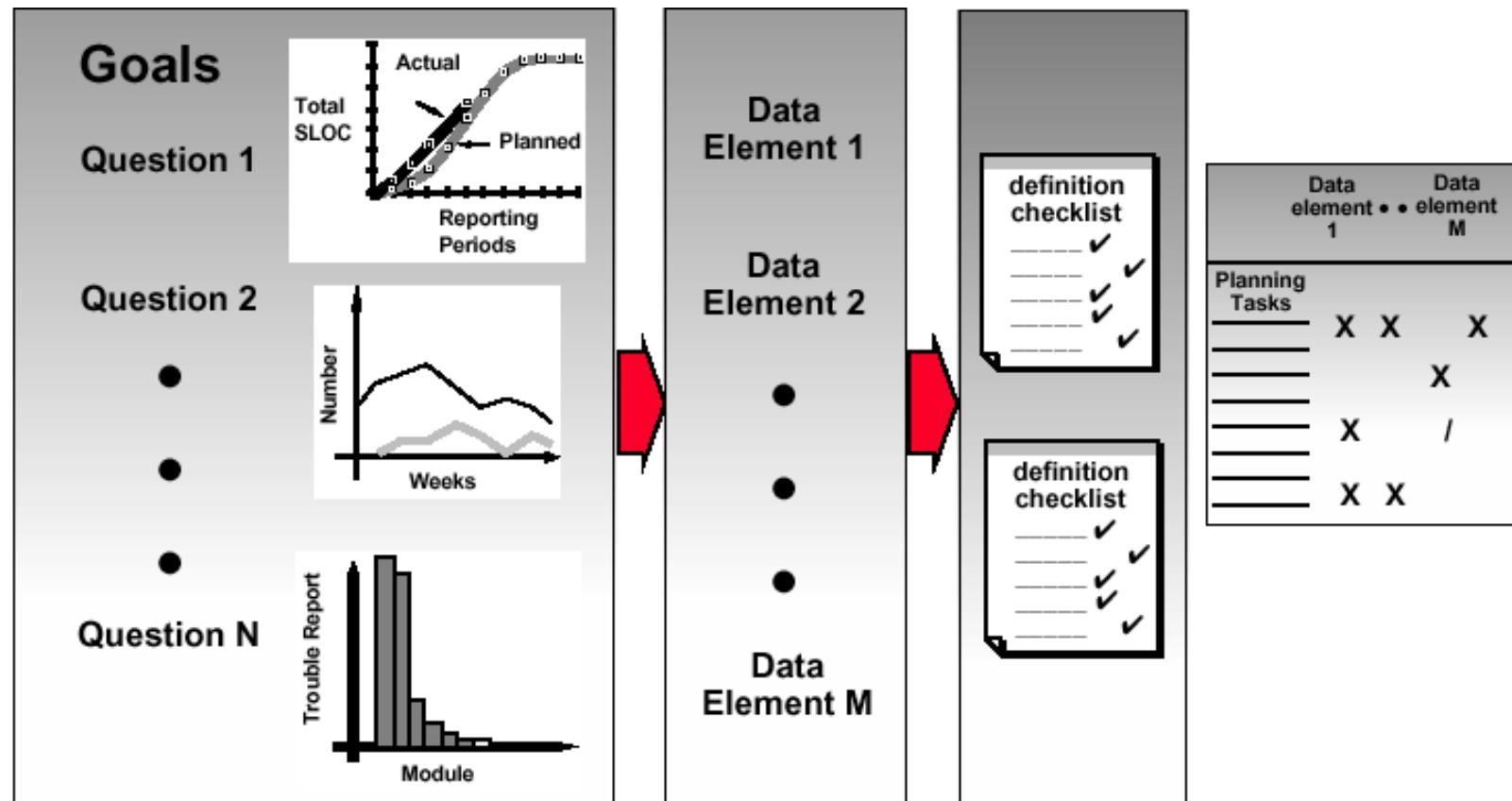
Ejemplos de requisitos:

Entidad Financiera	"Fiabilidad"	<ul style="list-style-type: none">• Sin cargos o abonos improc.• Conformidad importe p- importe c• ...
Fabricantes de coches	"Sin averías"	<ul style="list-style-type: none">• Sin reparación no accidental antes de 2 años• ...
Establecimiento hotelero	"Rapidez"	<ul style="list-style-type: none">• Recepción de mensajes < 15 min.• Chequeo inmediato• ...

Medición Orientada a Objetivos (GQM) (Goal Question Metric)



Medición Orientada a Objetivos (GQM) (Goal Question Metric)



Modelo de procesos

CMMI

1.0



CMM (Capabiliti Maturity Model)

- Deficiencias en las metodologías ⇒ Incapacidad para manejar el proceso de software
- En 1986, SEI (Software Engineering Institute): marco de trabajo sobre madurez de procesos
- En 1991, SEI desarrolló Capability Maturity Model (CMM)
 - Conjunto de prácticas recomendadas en determinadas áreas clave de proceso
 - **Mejora la capacidad del proceso de software**
 - Guía en la selección de estrategias de mejora de proceso
 - **Establecer la madurez de los procesos**
 - Determina cuestiones críticas para la calidad y la mejora del proceso

Organizaciones de software maduras / inmaduras

Idea principal: distinción entre empresas maduras/inmaduras

✗ En una organización **inmadura**:

- Procesos de software: improvisados o no respetados (si existen)
- Planificación en función de los problemas
- Presupuestos y planificación incumplidos
- Sin base objetiva para evaluar la calidad o para resolver problemas
- Inexistencia o reducción de las actividades de mejora de la calidad

✓ En una organización **madura**:

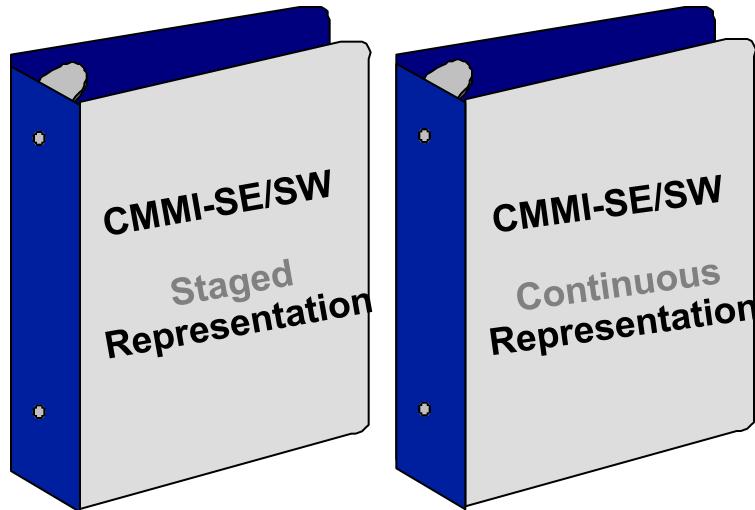
- Capacidad de gestión: desarrollo de software y procesos de mantenimiento
- Proceso de software difundido al equipo y planificado
- Procesos modificables: pruebas piloto controladas y análisis de coste/beneficio
- Roles y responsabilidades establecidos en el proyecto y la organización
- Gestores: monitorización la calidad de los productos y de los procesos
- Planificaciones y presupuestos realistas: rendimientos históricos
- Proceso disciplinado en el que todos los participantes entienden su valor, existiendo además la infraestructura necesaria para soportar el proceso

Proyecto CMMI

- DoD (Departamento de Defensa de los Estados Unidos), SEI (Software Engineering Institute) y NDIA (National Defense Industrial Association).
- **Más de 100 organizaciones involucradas**
 - U.S. Army, Navy, Air Force
 - Federal Aviation Administration
 - National Security Agency
 - Software Engineering Institute
 - ADP, Inc.
 - AT&T Labs
 - BAE
 - Boeing
 - Computer Sciences Corporation
 - EER Systems
 - Ericsson Canada
 - Ernst and Young
 - General Dynamics
 - Harris Corporation
 - Honeywell
 - KPMG
 - Lockheed Martin
 - Motorola
 - Northrop Grumman
 - Pacific Bell
 - Q-Labs
 - Raytheon
 - Reuters
 - Rockwell Collins
 - SAIC
 - Software Productivity Consortium
 - Sverdrup Corporation
 - TeraQuest
 - Thomson CSF
 - TRW



Modelos CMMI



- **Modelo combinado**
System Engineering/Software Engineering
- **Aplicable:**
 - Sólo a proyectos de software engineering
 - Sólo a proyectos de system engineering
 - Ambos

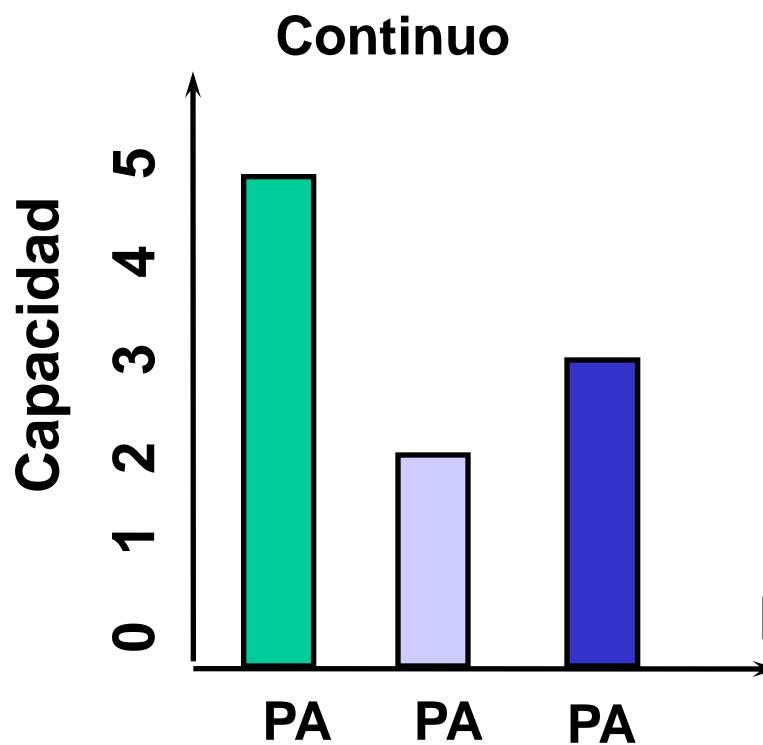
¿Continua o escalonada?

- Ambas incluyen el **mismo contenido** y consiguen **idénticos objetivos**
- La representación **continua** centra su actuación en la **CAPACIDAD DE LOS PROCESOS**
- La representación **escalonada** centra su actuación en la **MADUREZ DE LA ORGANIZACIÓN**

¿Por qué dos representaciones del modelo?

- Heredado de los modelos de origen.
 - Software CMM--Escalonado
 - SECM--Continuo
 - IPD CMM--Híbrido
- En el equipo de desarrollo de CMMI había defensores de cada una de las representaciones.
- Seleccionar una única representación se planteaba como algo “too hard”.
- Compromiso: Inicialmente soportar dos representaciones del modelo con contenidos equivalentes.

Un modelo, dos representaciones



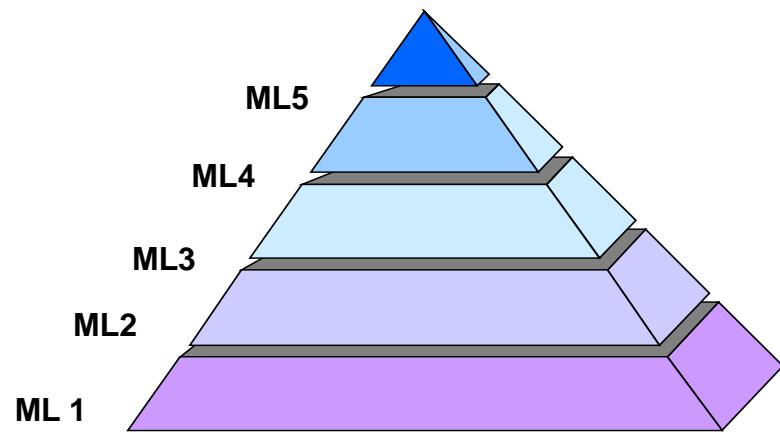
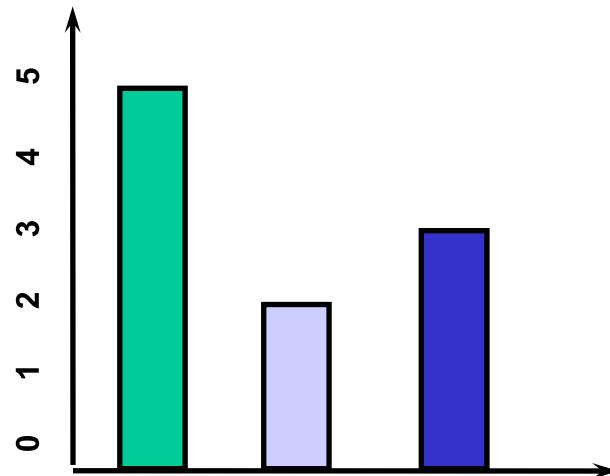
...para un área de proceso
o un conjunto de áreas de proceso



...para un conjunto de
áreas de proceso establecido

Capacidad y madurez

- **Capacidad** es un atributo que se aplica a los procesos y define la eficacia del mismo para conseguir los objetivos previstos.
- **Madurez** es un atributo que se aplica a la organización y define su potencial de calidad en la producción.



Niveles de capacidad

■ 0 – Incompleto

Los procesos no se realizan, o no consiguen sus objetivos

■ 1 – Ejecutado

Los procesos se ejecutan y se logran los objetivos específicos del área

■ 2 – Gestionado

Los procesos que además de considerarse “ejecutados” son también planificados, revisados y evaluados para comprobar que cumplen los requisitos

■ 3 – Definido

Los procesos que además de considerarse “gestionados” se ajustan al conjunto de procesos estándar conforme a las líneas directivas de la organización

■ 4 – Gestión cuantificada

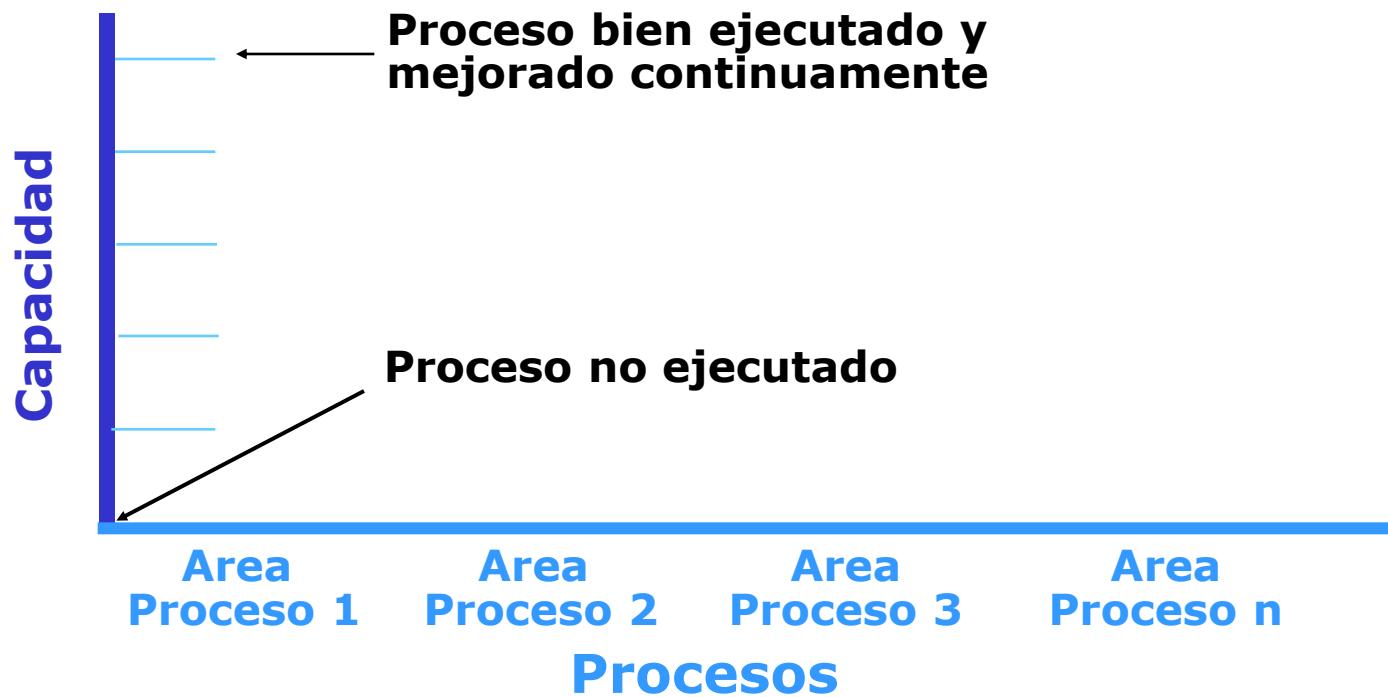
Procesos “definidos” que son controlados utilizando técnicas estadísticas u otras técnicas cuantitativas

■ 5 – Optimizado

Procesos “gestionados cuantificadamente” que son cambiados y adaptados para conseguir objetivos relevantes de negocio

Dimensión de la capacidad

Los valores describen “cómo de bien” se realiza el proceso (nivel de capacidad del proceso).



Niveles de madurez

- **1 – Inicial**

Control deficiente e impredecibilidad de los resultados

- **2 – Gestionado**

Es posible obtener niveles de calidad previamente alcanzados

- **3 – Definido**

Los procesos realizados se encuentran normalizados, son conocidos y comprendidos

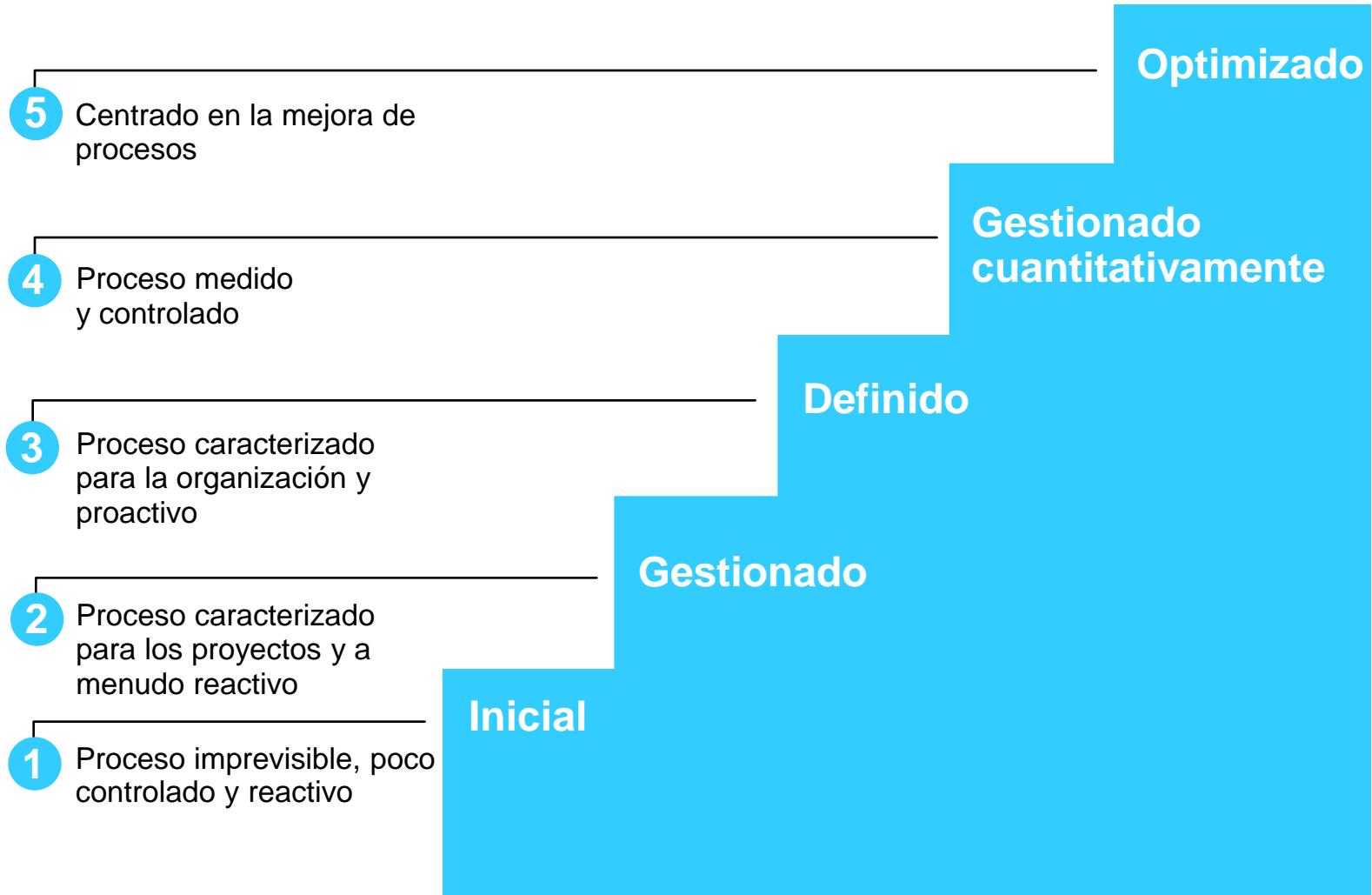
- **4 – Gestionado cuantitativamente**

Los procesos incluyen indicadores de medición y control

- **5 – Optimizado**

Centralización en la mejora de los procesos

Dimensión de la madurez



Áreas de procesos

- CMMI recoge prácticas para **22 áreas de procesos**
- Las áreas de procesos agrupan a las actividades necesarias para la ejecución de los proyectos de ingeniería de sistemas y de software
- El modelo en su representación escalonada clasifica a las 22 áreas de proceso en aquellas cuya gestión es necesaria para lograr cada nivel de calidad
- El modelo en su representación continua las clasifica según a la categoría que pertenecen: Gestión de proyectos, ingeniería, soporte y gestión de procesos

Áreas de procesos en la representación escalonada

NIVEL DE MADUREZ	ÁREAS DE PROCESO
5 OPTIMIZADO	Innovación y desarrollo
4 GESTIONADO CUANTITATIVAMENTE	Gestión cuantificada de proyectos Rendimiento de los procesos de la organización
3 DEFINIDO	Desarrollo de requisitos Solución técnica Verificación Validación Integración de producto Procesos orientados a la organización Definición de los procesos de la organización Formación Gestión integrada de proyecto Gestión de riesgos Análisis y resolución de decisiones
2 GESTIONADO	Gestión de requisitos Planificación de proyecto Monitorización y control de proyectos Gestión y acuerdo con suministradores Medición y análisis Gestión de la calidad de procesos y productos Gestión de la configuración
1 INICIAL	

Áreas de procesos en la representación continua

CATEGORÍA	ÁREAS DE PROCESO
GESTIÓN DE PROYECTOS	Planificación de proyecto Monitorización y control de proyecto Gestión y acuerdo con proveedores Gestión integrada de proyecto Gestión de riesgos Gestión cuantificada de proyecto
SOPORTE	Gestión de la configuración Gestión de la calidad de procesos y productos Medición y análisis Análisis y resolución de decisiones Análisis y resolución de problemas
INGENIERÍA	Desarrollo de requisitos Gestión de requisitos Soluciones técnicas Integración de producto Verificación Validación
GESTIÓN DE PROCESOS	Definición de los procesos de la organización Procesos orientados a la organización Formación Rendimiento de los procesos de la organización Innovación y desarrollo

Cómo usar el modelo

Área de proceso

Conjunto de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos.

Componentes requeridos

■ Objetivo genérico

Los objetivos genéricos asociados a un nivel de capacidad establecen lo que una organización debe alcanzar en ese nivel de capacidad.

El logro de cada uno de esos objetivos en un área de proceso significa mejorar el control en la ejecución del área de proceso

■ Objetivo específico

Los objetivos específicos se aplican a una única área de proceso y localizan las particularidades que describen que se debe implementar para satisfacer el propósito del área de proceso

Cómo usar el modelo

Componentes esperados

- **Práctica genérica**

Una práctica genérica se aplica a cualquier área de proceso porque puede mejorar el funcionamiento y el control de cualquier proceso.

- **Práctica específica**

Una práctica específica es una actividad que se considera importante en la realización del objetivo específico al cual está asociado.

Las prácticas específicas describen las actividades esperadas para lograr la meta específica de un área de proceso.

Componentes informativos

- **Propósito**

- **Notas introductorias**

- **Referencias**

Las referencias son indicadores de otras áreas de proceso relacionadas que pueden aportar información adicional o más detallada

- **Nombres**

- **Tablas de relaciones práctica-objetivo**

- **Prácticas**

Cómo usar el modelo

Componentes informativos

- **Propósito**
- **Notas introductorias**
- **Referencias**

Las referencias son indicadores de otras áreas de proceso relacionadas que pueden aportar información adicional o más detallada

- **Nombres**
- **Tablas de relaciones práctica-objetivo**
- **Prácticas**
- **Productos típicos**
- **Subprácticas**

Una subpráctica es una descripción detallada que sirve como guía para la interpretación de una práctica genérica o específica

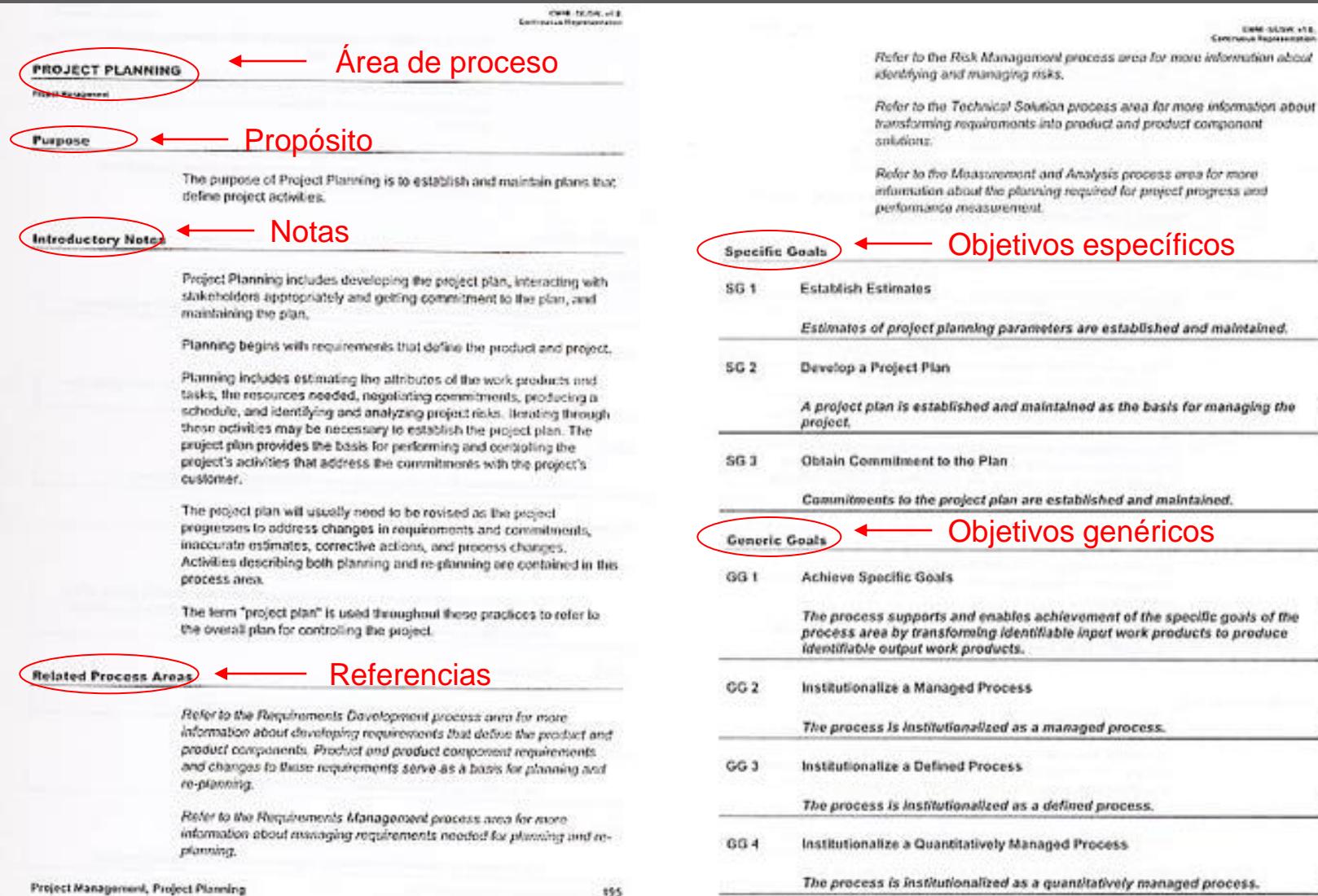
- **Ampliaciones de disciplina**

Las ampliaciones contienen información relevante de una disciplina particular y relacionada con una práctica específica.

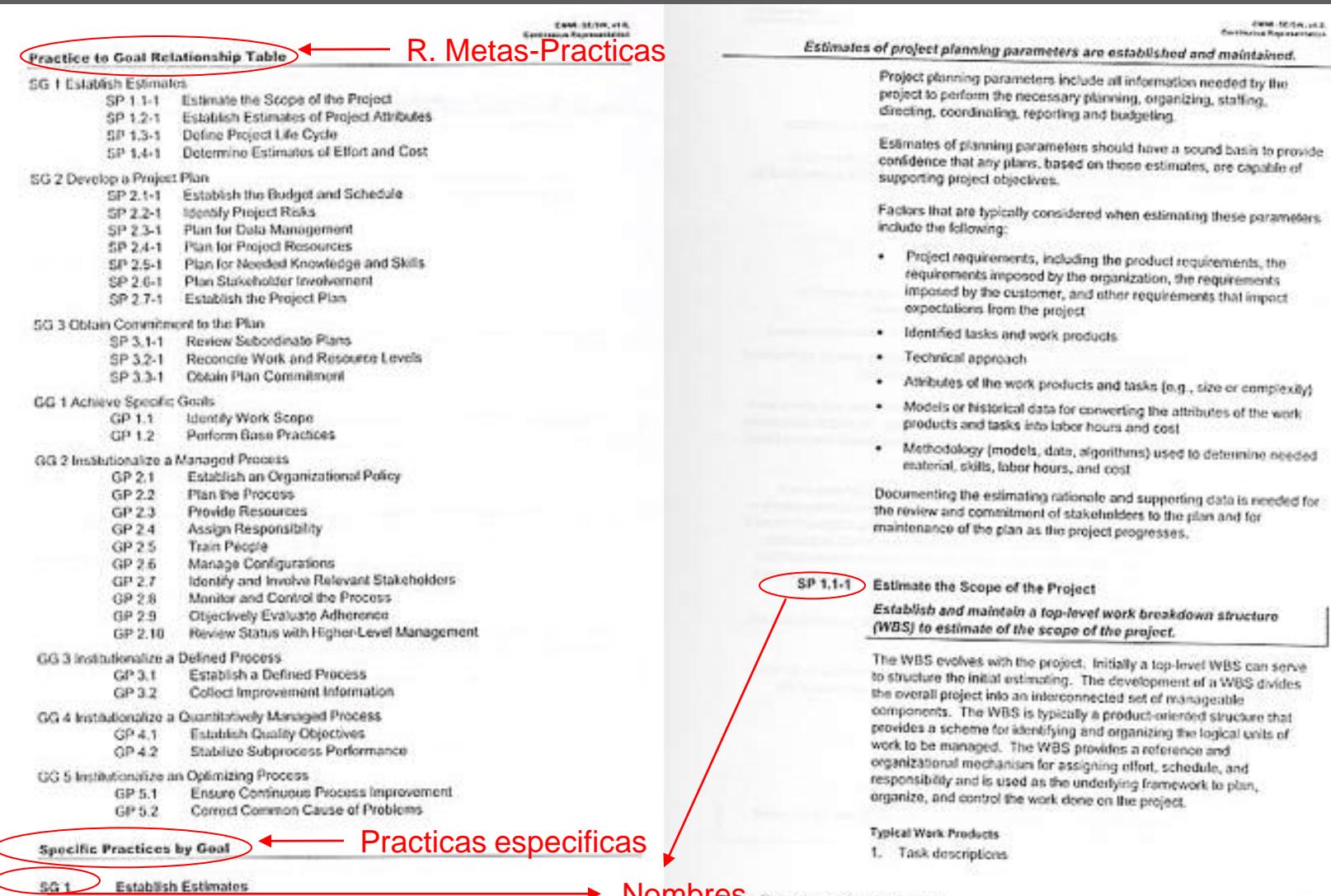
- **Elaboraciones de prácticas genéricas**

Una elaboración de una práctica genérica es una guía de cómo la práctica genérica debe aplicarse al área de proceso.

Mapa del documento



Mapa del documento



Mapa del documento

SG 3 Verify Selected Work Products	
<i>Selected work products are verified against their specified requirements.</i>	
SP 3.1-1 Perform Verification	<p><i>Perform verification according to the verification strategy.</i></p> <p>Verifying products and work products incrementally promotes early detection of problems and can remove defects early. Those resultant verification save considerable cost of fault isolation and rework associated with troubleshooting problems.</p>
<p>Notas</p> <p>Typical Work Products</p> <ul style="list-style-type: none"> 1. Verification results 2. Verification reports 3. Demonstrations 4. "As Verified" procedures log <p>Subpractices</p> <ul style="list-style-type: none"> 1. Verify COTS and reused components to verify that they meet the requirements. 2. Perform product verification against the requirements according to the verification strategy and procedures. 3. Capture the results of verification activities. 4. Identify action items resulting from verification of work products. 5. Document the "as-run" verification method and the deviations from the strategies and procedures made during its performance. 	
SP 3.2-2 Analyze Verification Results and Identify Corrective Action	<p><i>Analyze the results of all verification activities and identify corrective action.</i></p> <p>Actual results must be compared to established verification criteria to determine acceptability.</p> <p>The results of the analysis are recorded as evidence that verification was conducted.</p> <p>Analysis reports or "as-run" method documentation may also indicate that bad verification results are due to method problems, criteria problems, or an infrastructure problem.</p>

Generic Practices by Goal	
GG 1 Achieve Specific Goals	<p><i>The process supports and enables achievement of the specific goals of the process area by transforming identifiable input work products to produce identifiable output work products.</i></p>
GP 1.1 Identify Work Scope	<p><i>Identify the scope of the work to be performed and work products to be produced for verification, and communicate this information to those performing the work.</i></p>
GP 1.2 Perform Base Practices	<p><i>Perform the base practices of the verification process to develop work products and provide services to achieve the specific goals of the process area.</i></p>
GG 2 Institutionalize a Managed Process	<p><i>The process is institutionalized as a managed process.</i></p>
GP 2.1 Establish an Organizational Policy	<p><i>Establish and maintain an organizational policy for planning and performing the verification process.</i></p> <p>Elaboracion:</p> <p>This policy establishes organizational expectations for establishing and maintaining a verification strategy and environment, and performing peer reviews and verifying selected work products.</p>
GP 2.2 Plan the Process	<p><i>Establish and maintain the requirements and objectives, and plans for performing the verification process.</i></p> <p>Elaboracion:</p> <p>These requirements, objectives, and plans are described in the plan for verification. This plan for verification differs from the verification strategy described in the specific practices in this process area. The verification strategy addresses specific actions, resources, and environments required for work product verification, whereas the plan for verification addresses high-level planning for all the verification.</p>

Áreas de proceso

- **CMMI SE/SW** incluye **22 áreas de proceso**
- Las áreas de proceso son **iguales** en ambas representaciones del modelo
- En la representación continua, las áreas de proceso se agrupan por categorías: **Gestión de proyectos, Ingeniería, Soporte y Gestión de procesos**

[Process areas by capability](#)

- En la representación escalonada, las áreas de proceso se agrupan por **niveles de madurez** (2 – 5)

[Process areas by maturity level](#)

Áreas de proceso

Gestión de proyecto

Las áreas de procesos de gestión de proyectos cubren las actividades relacionadas con la planificación, monitorización y control del proyecto.

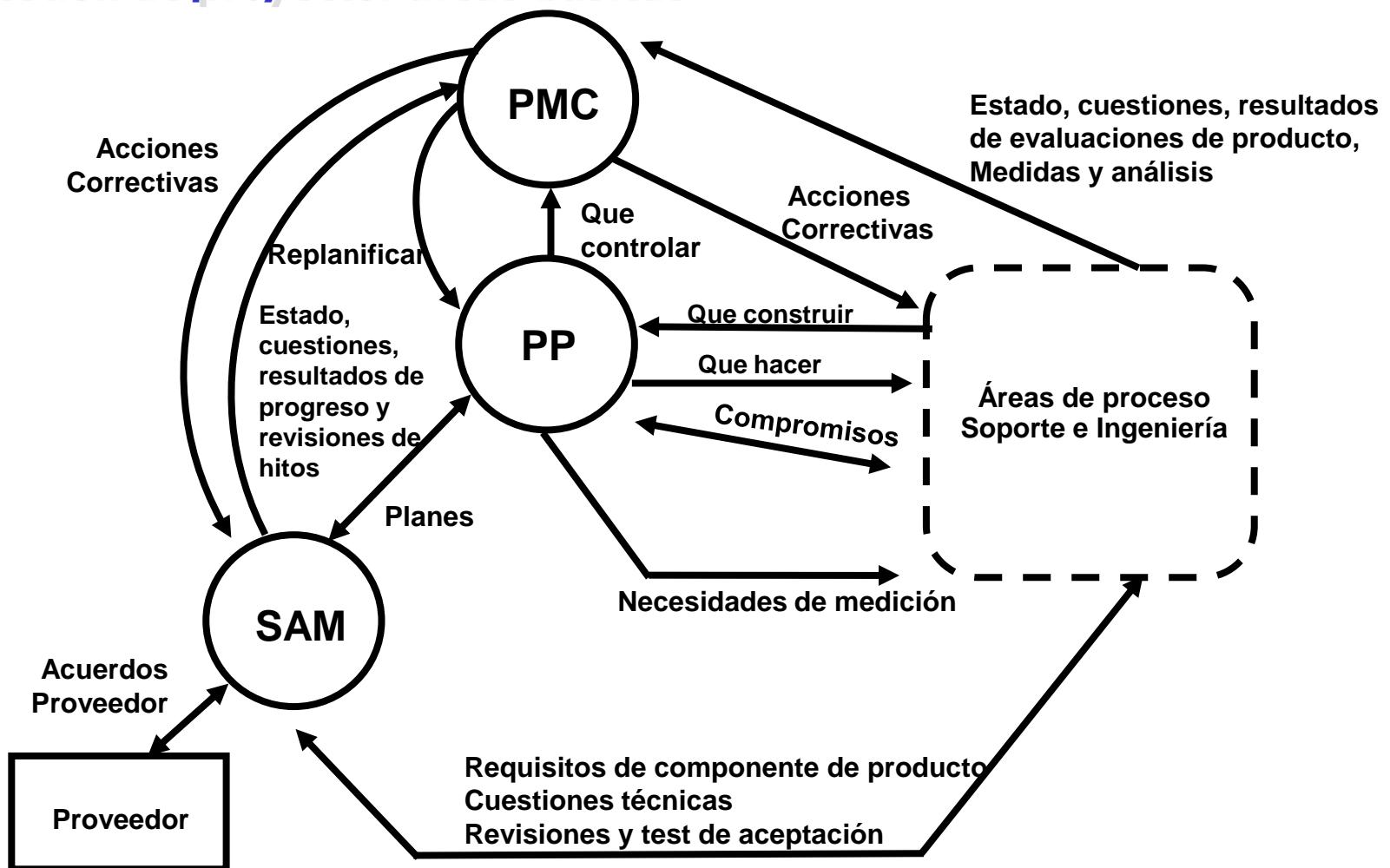
El modelo **CMMI SE/SW** incluye **seis áreas de proceso** en gestión de proyectos:

- **Planificación de proyecto**
- **Monitorización y control de proyecto**
- **Gestión y acuerdos con proveedores**
- **Gestión integrada de proyecto**
- **Gestión de riesgos**
- **Gestión cuantificada de proyecto**



Áreas de proceso

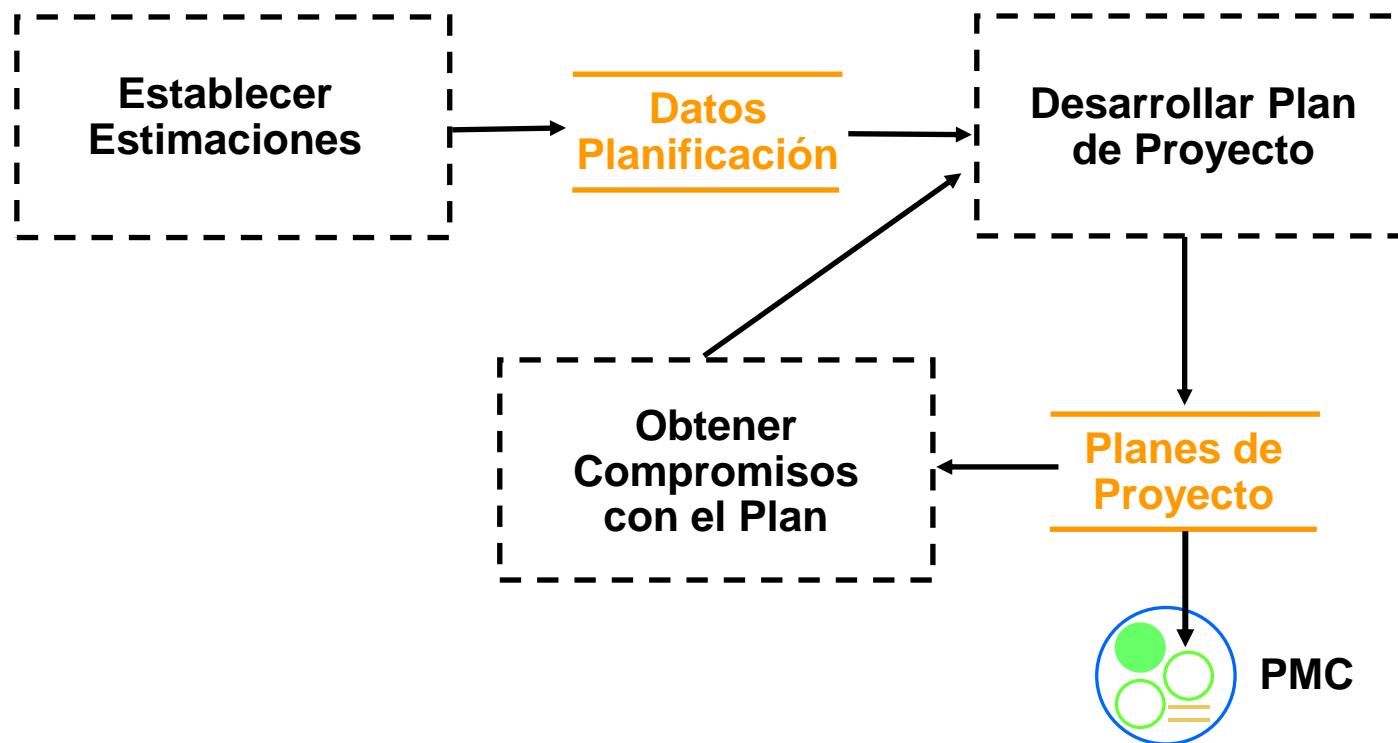
Gestión de proyecto: áreas básicas



Áreas de proceso

Gestión de proyecto: planificación de proyecto

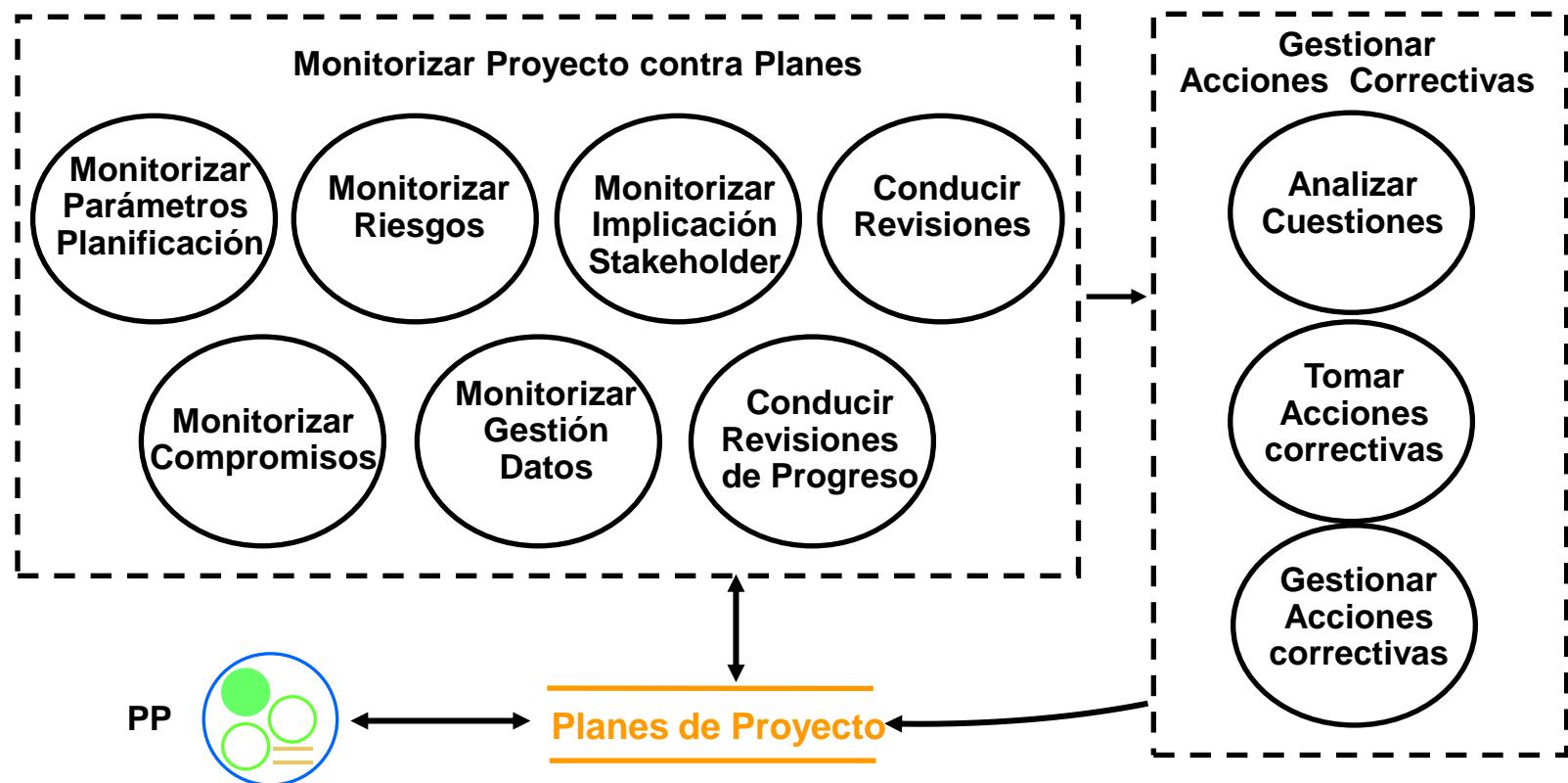
Propósito: establecer y mantener planes que definen las actividades del proyecto



Áreas de proceso

Gestión de proyecto: monitorización y control

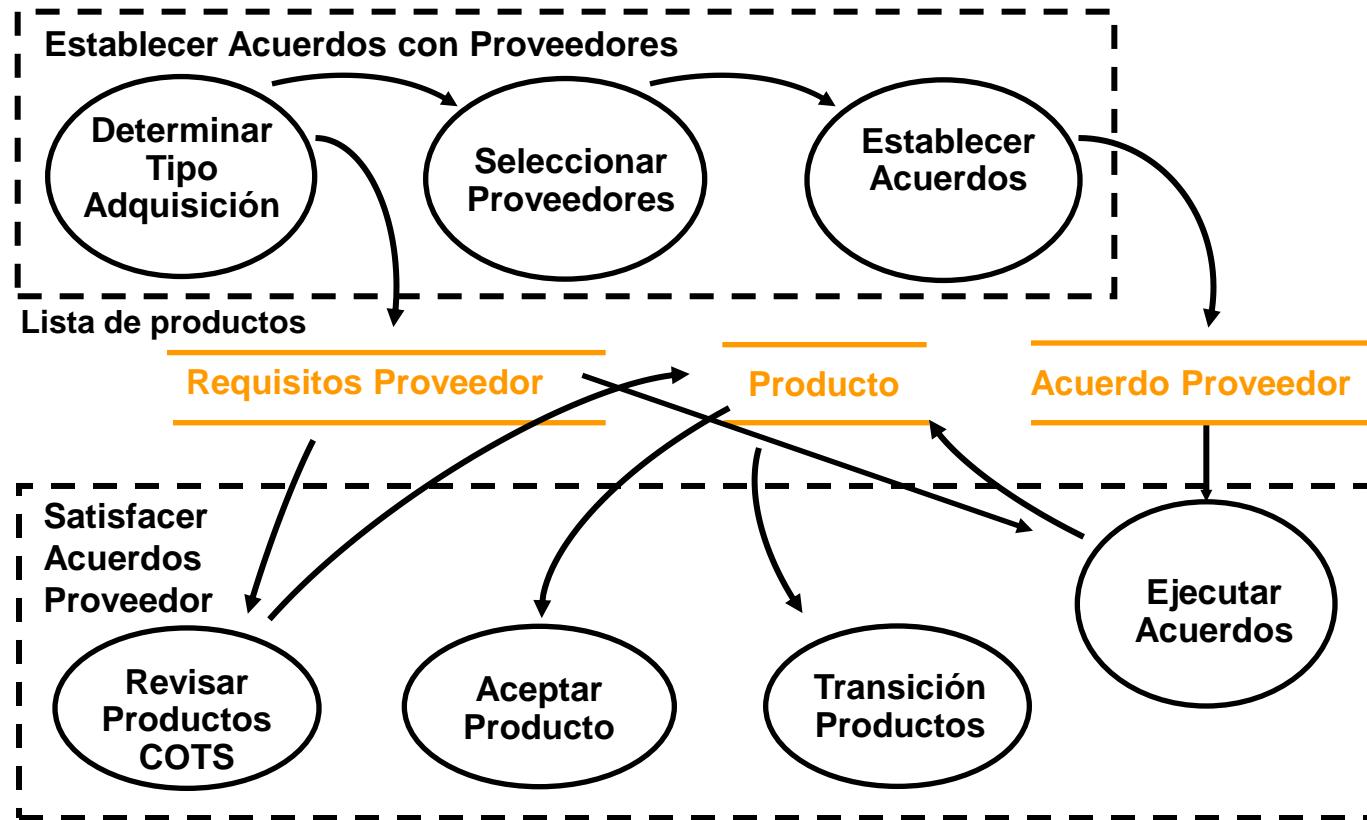
Propósito: Proporcionar información sobre el progreso del proyecto que permita tomar acciones correctivas cuando la ejecución del proyecto se desvía significativamente del plan.



Áreas de proceso

Gestión de proyecto: gestión y acuerdos con proveedores

Propósito: Gestionar la adquisición de productos a proveedores según un acuerdo formal existente.



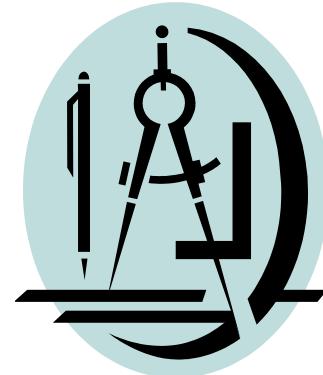
Áreas de proceso

Ingeniería

Las áreas de proceso de ingeniería cubren las prácticas de desarrollo y mantenimiento compartidas por diferentes disciplinas como ingeniería de software e ingeniería de sistemas.

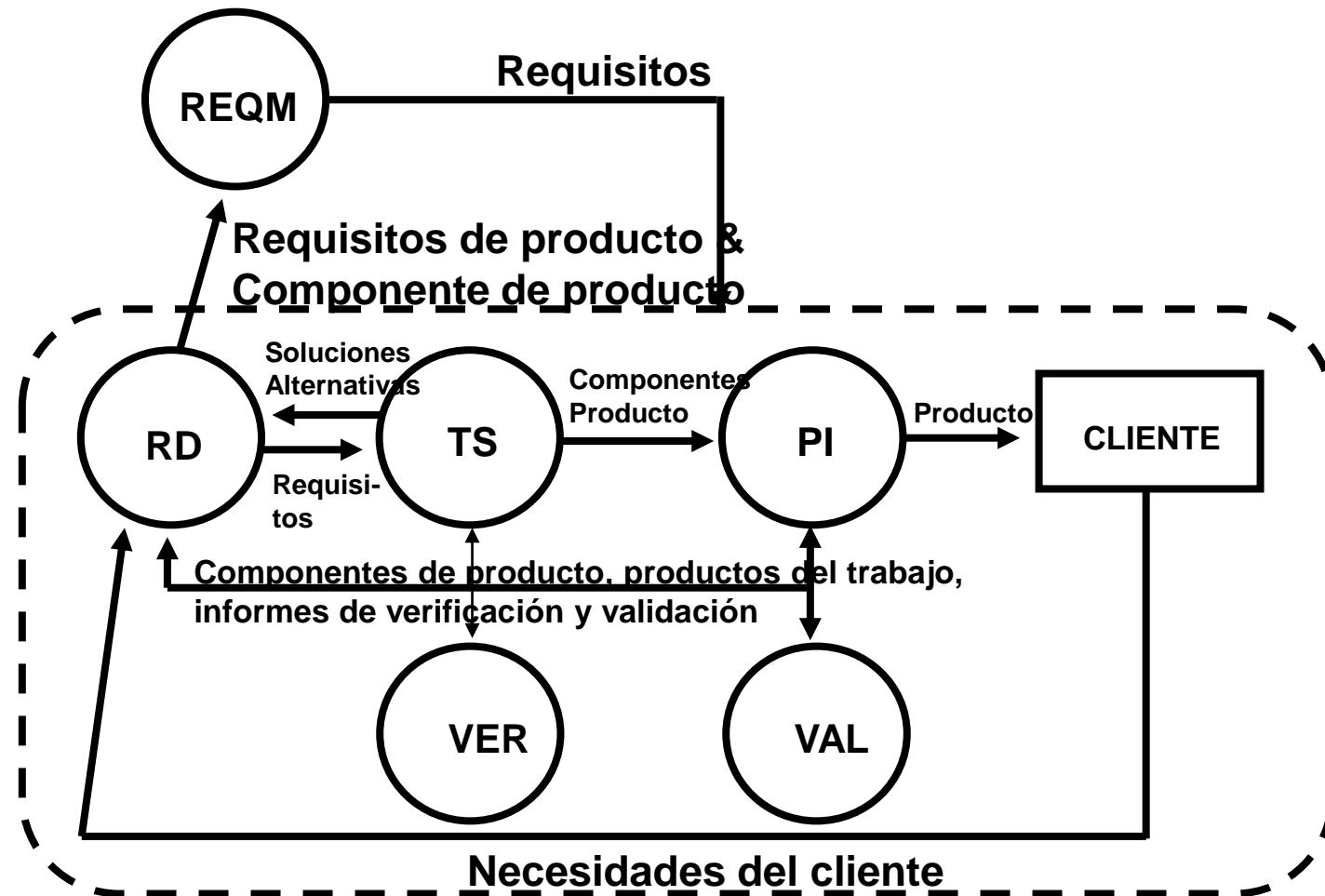
El modelo **CMMI SE/SW** incluye **seis áreas de proceso** en ingeniería:

- **Desarrollo de requisitos**
- **Gestión de requisitos**
- **Soluciones técnicas**
- **Integración de producto**
- **Verificación**
- **Validación**



Áreas de proceso

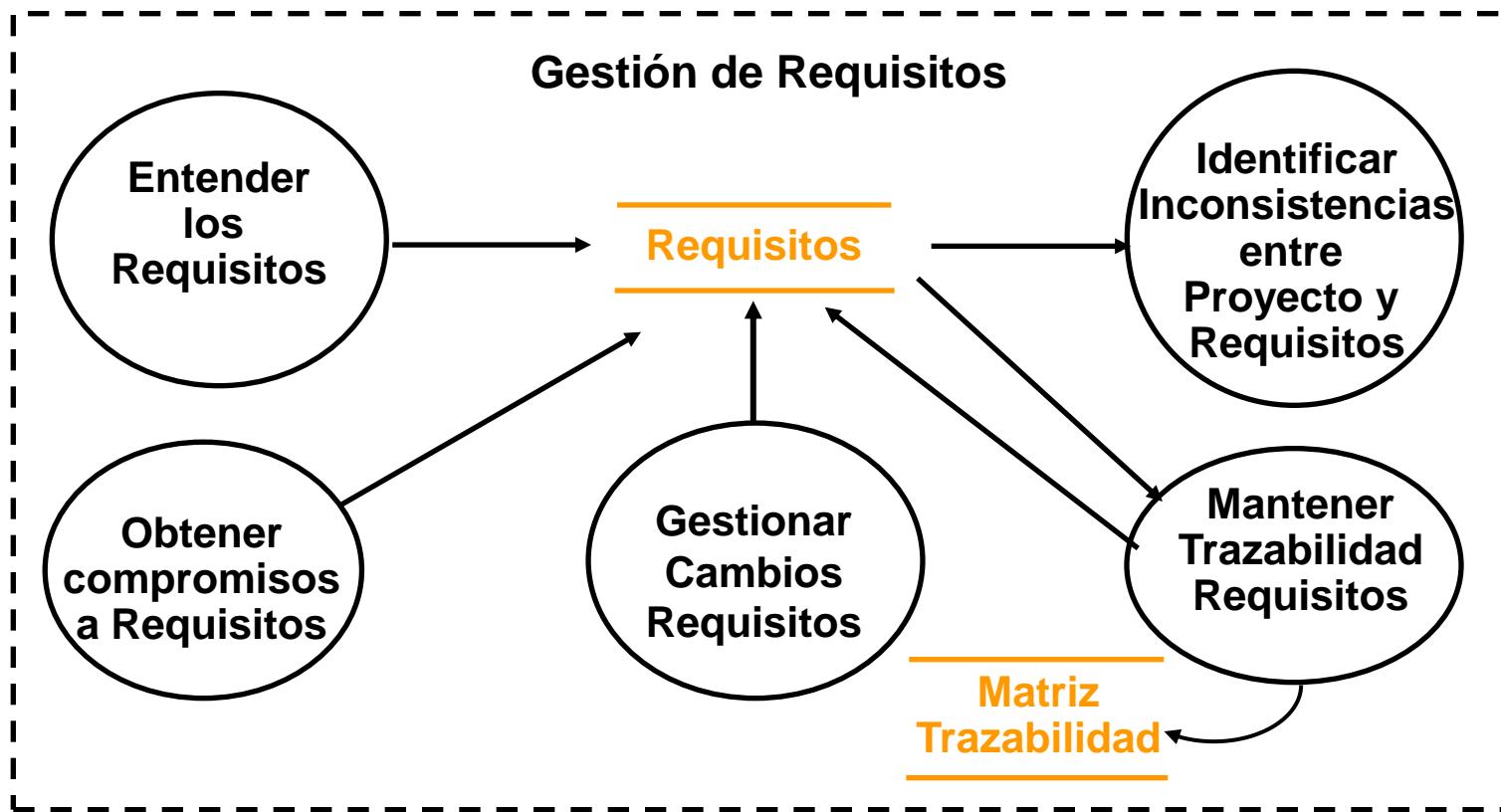
Ingeniería: áreas básicas



Áreas de proceso

Ingeniería: gestión de requisitos

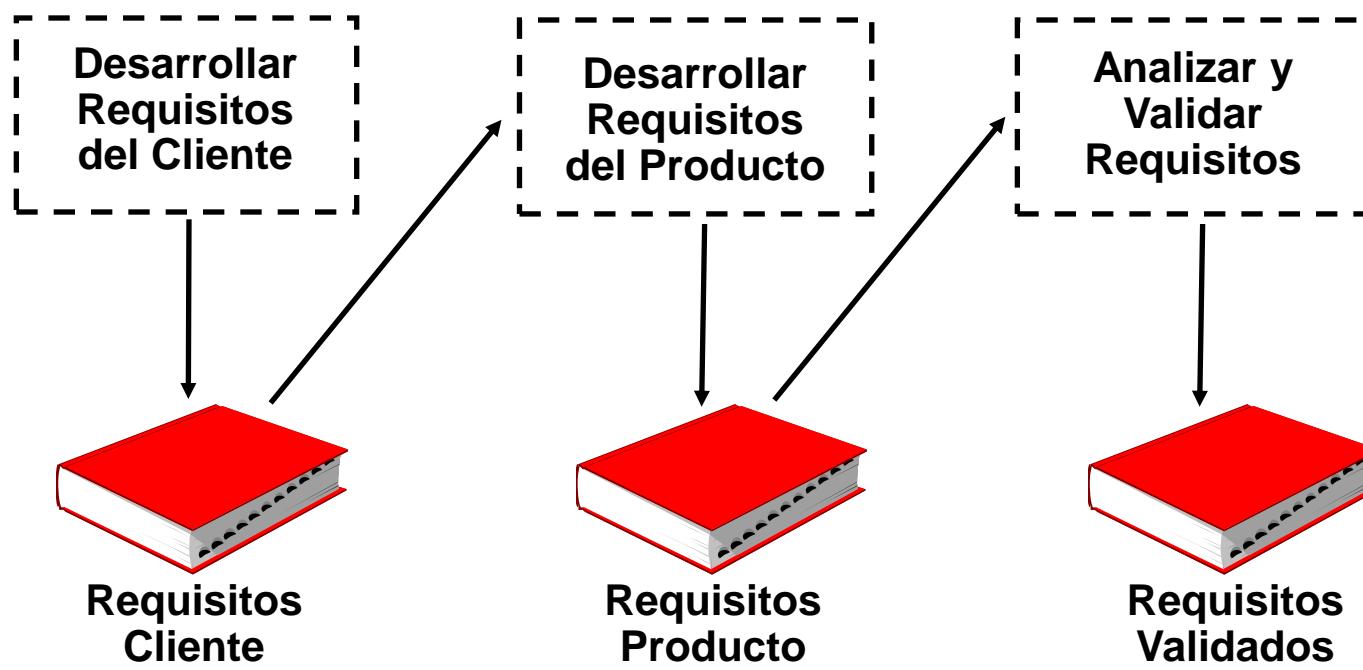
Propósito: Gestionar los requisitos del producto y de componentes del producto del proyecto e identificar inconsistencias entre los requisitos, los planes del proyecto y los resultados del trabajo.



Áreas de proceso

Ingeniería: desarrollo de requisitos

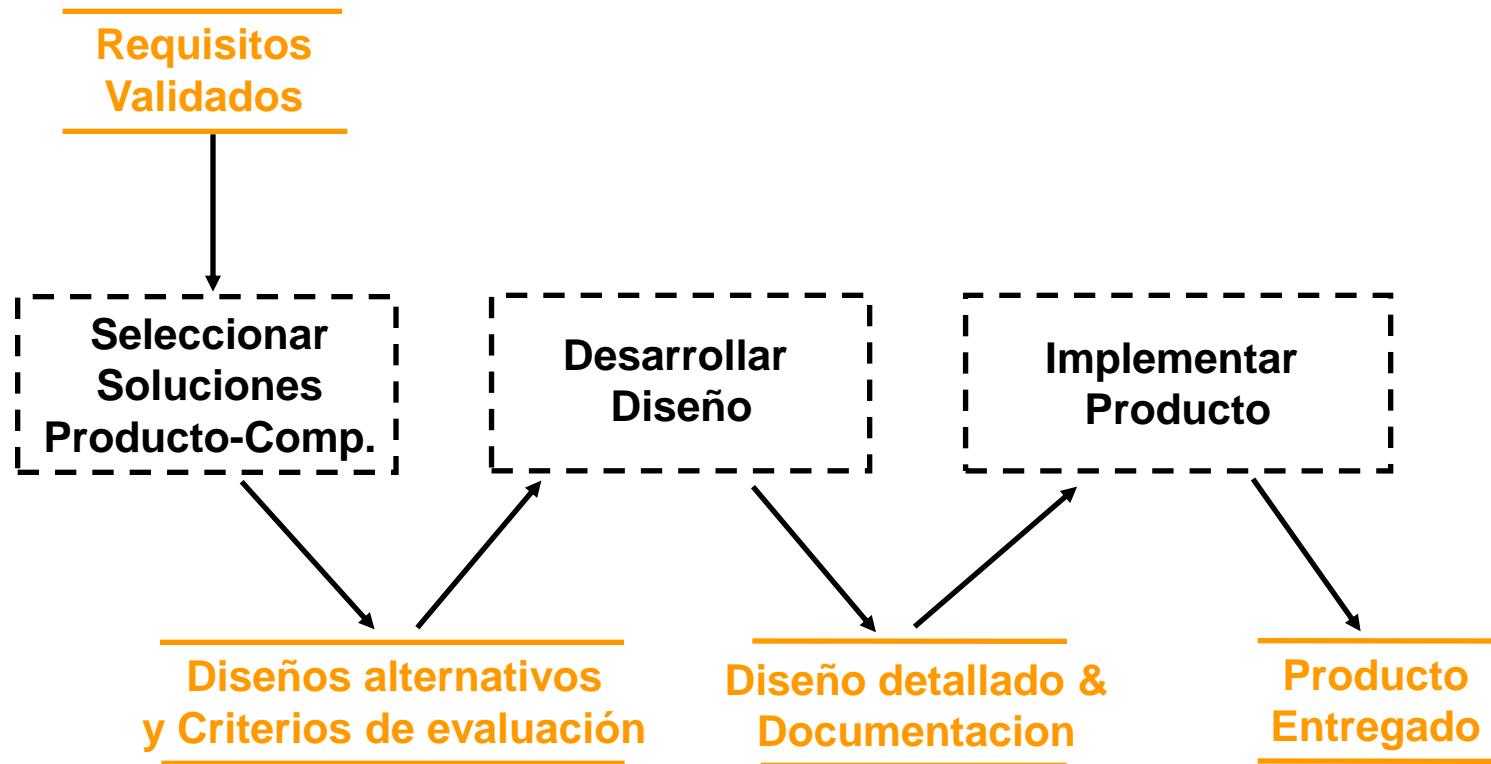
Propósito: Producir y analizar los requisitos del cliente, del producto y de los componentes de producto.



Áreas de proceso

Ingeniería: solución técnica

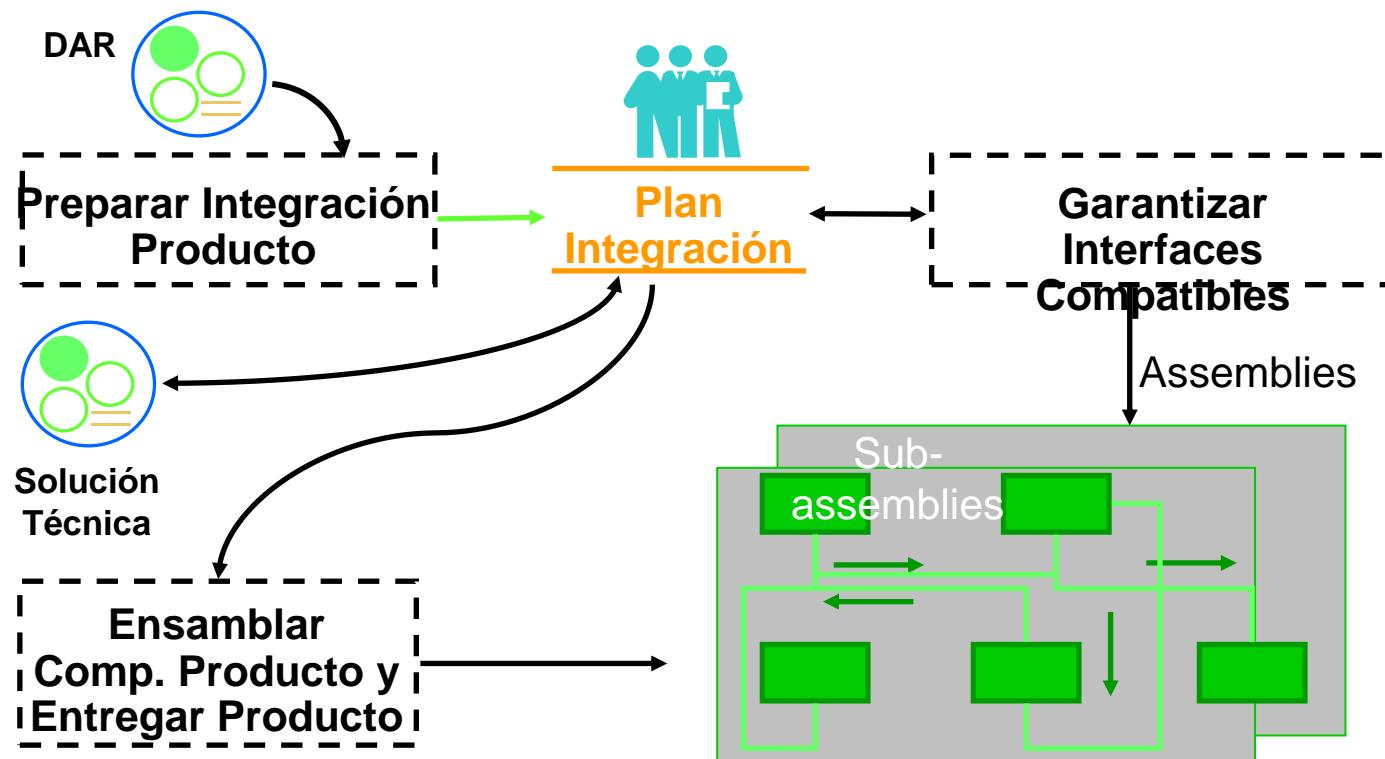
Propósito: Desarrollar, diseñar e implementar soluciones a los requisitos.



Áreas de proceso

Ingeniería: integración de producto

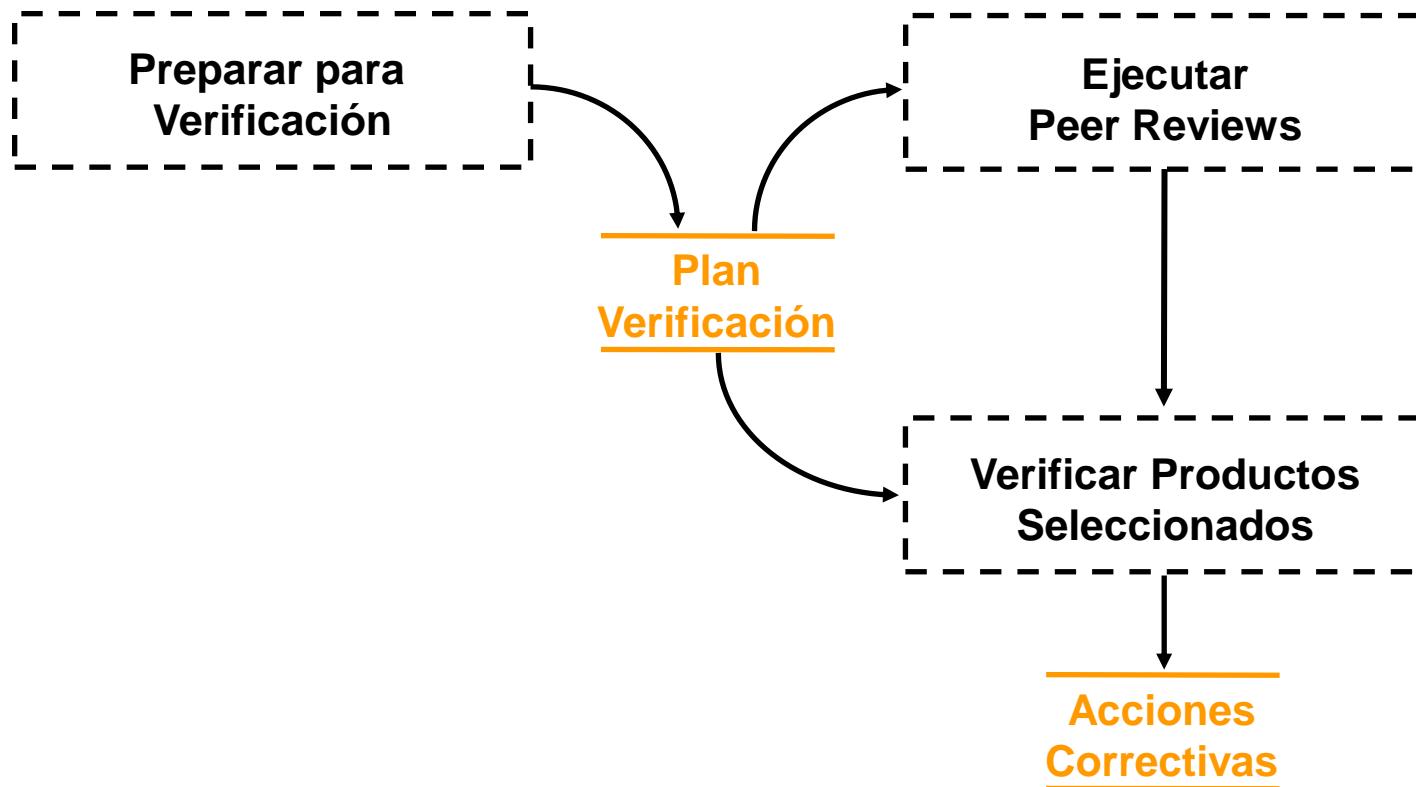
Propósito: Ensamblar el producto asegurando que funciona apropiadamente y entregar el producto.



Áreas de proceso

Ingeniería: verificación

Propósito: Asegurar que los resultados del trabajo seleccionados cumplen los requisitos especificados.



Áreas de proceso

Ingeniería: validación

Propósito: Demostrar que un producto o componente de producto satisface su uso previsto en el entorno previsto.

- Requisitos Cliente
- Requisitos Producto
- Productos
- Validación de Requisitos

Preparar para Validación

- Plan Validación Requisitos
- Plan Validación Producto
- Procesos y necesidades de Soporte

Validar Producto o Componentes Producto

- Conformidad
- Deficiencias

Áreas de proceso

Soporte

Las áreas de procesos de soporte cubren las prácticas que sirven de base para el desarrollo del producto y su mantenimiento.

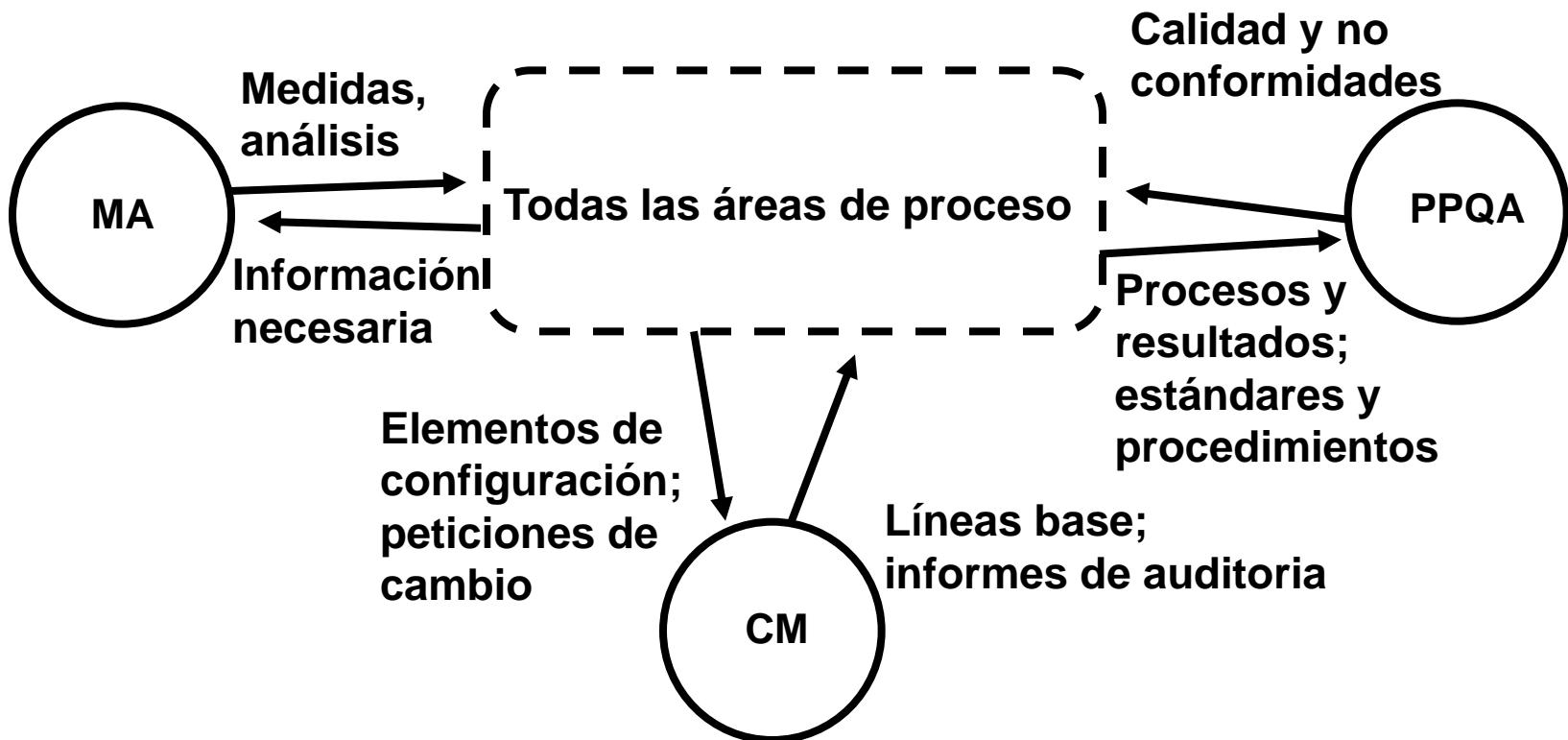
El modelo **CMMI SE/SW** incluye **cinco áreas de proceso** en soporte:

- **Gestión de la configuración**
- **Gestión de la calidad de procesos y productos**
- **Medición y análisis**
- **Análisis y resolución de decisiones**
- **Análisis y resolución de problemas**



Áreas de proceso

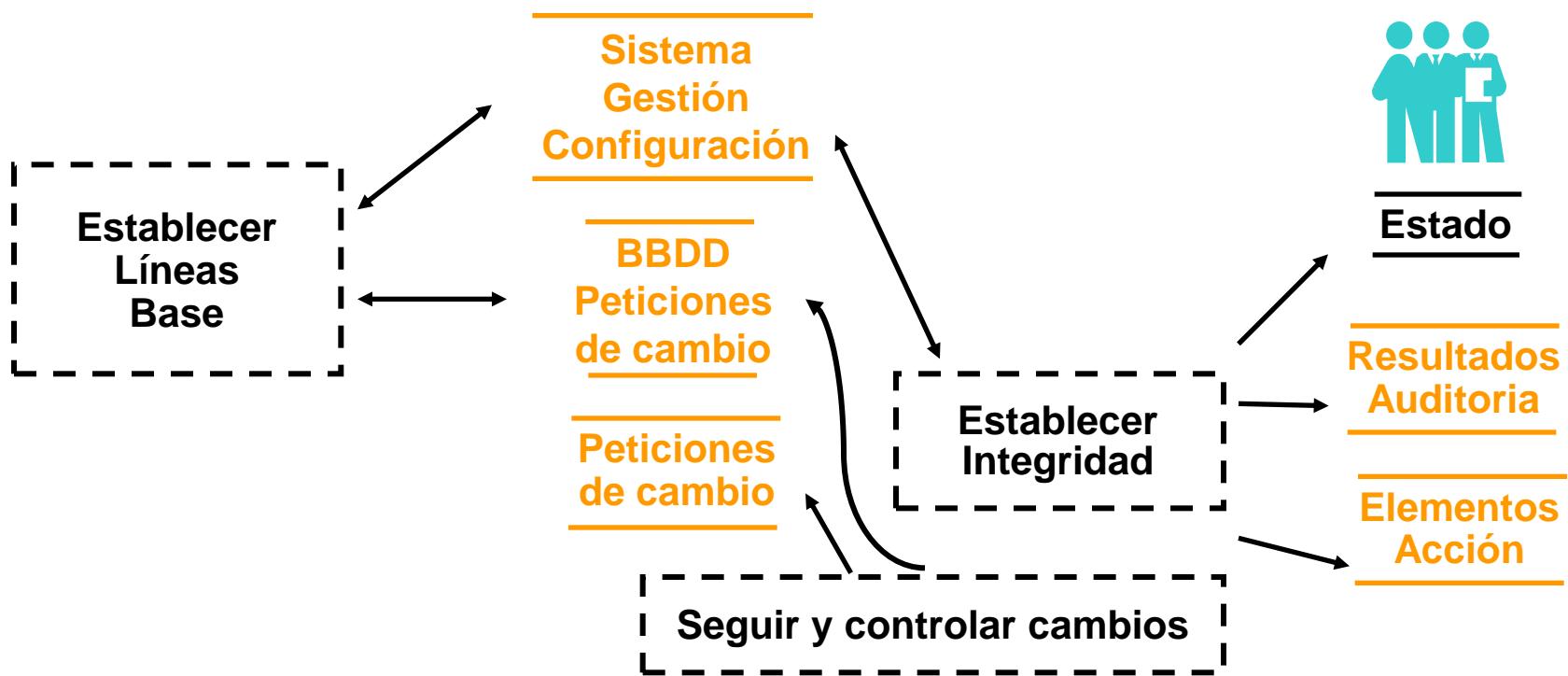
Soporte: áreas básicas



Áreas de proceso

Soporte: gestión de la configuración

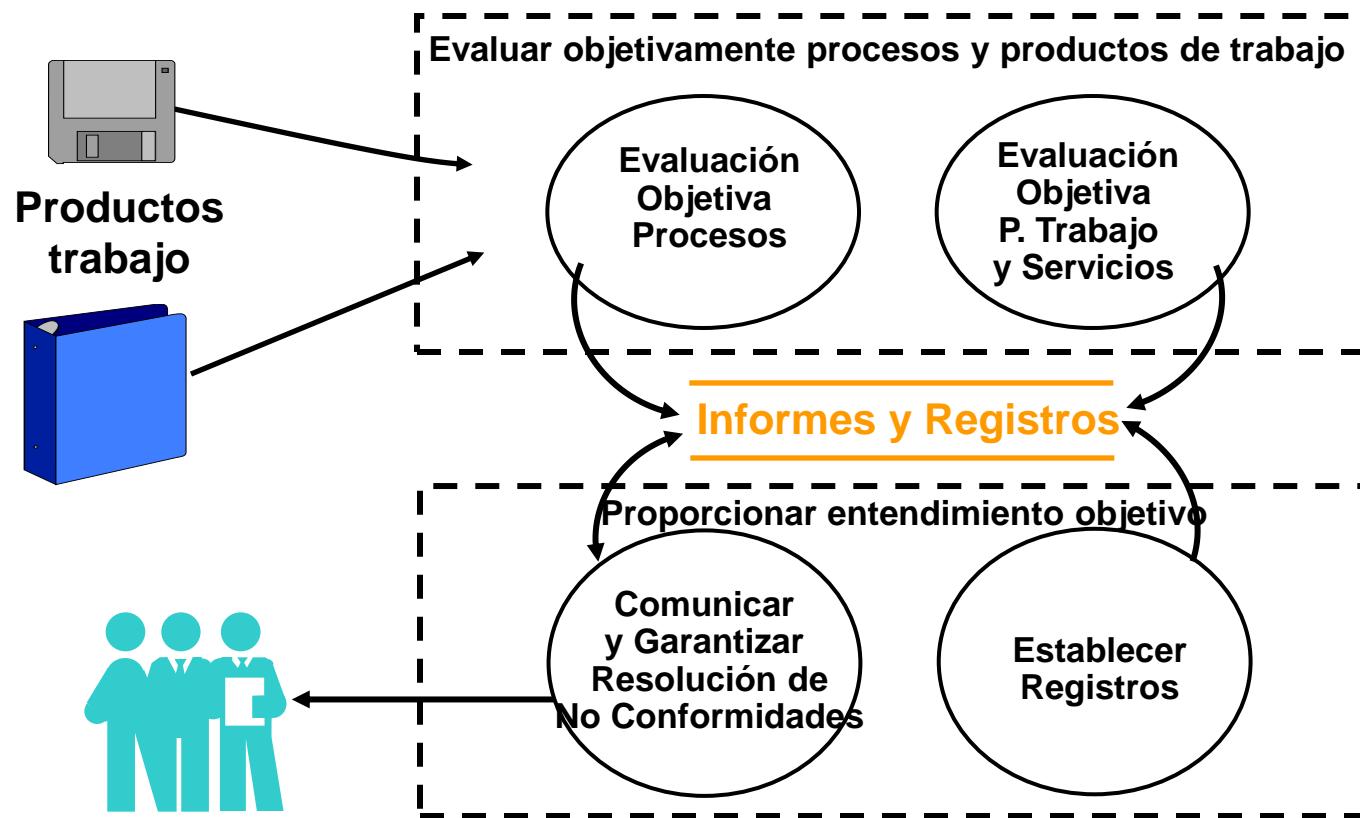
Propósito: Establecer y mantener la integridad de todos los productos de trabajo, utilizando identificación de la configuración, control de la configuración, informes de estado de configuración y auditorías de la configuración.



Áreas de proceso

Soporte: aseguramiento de la calidad de procesos y producto

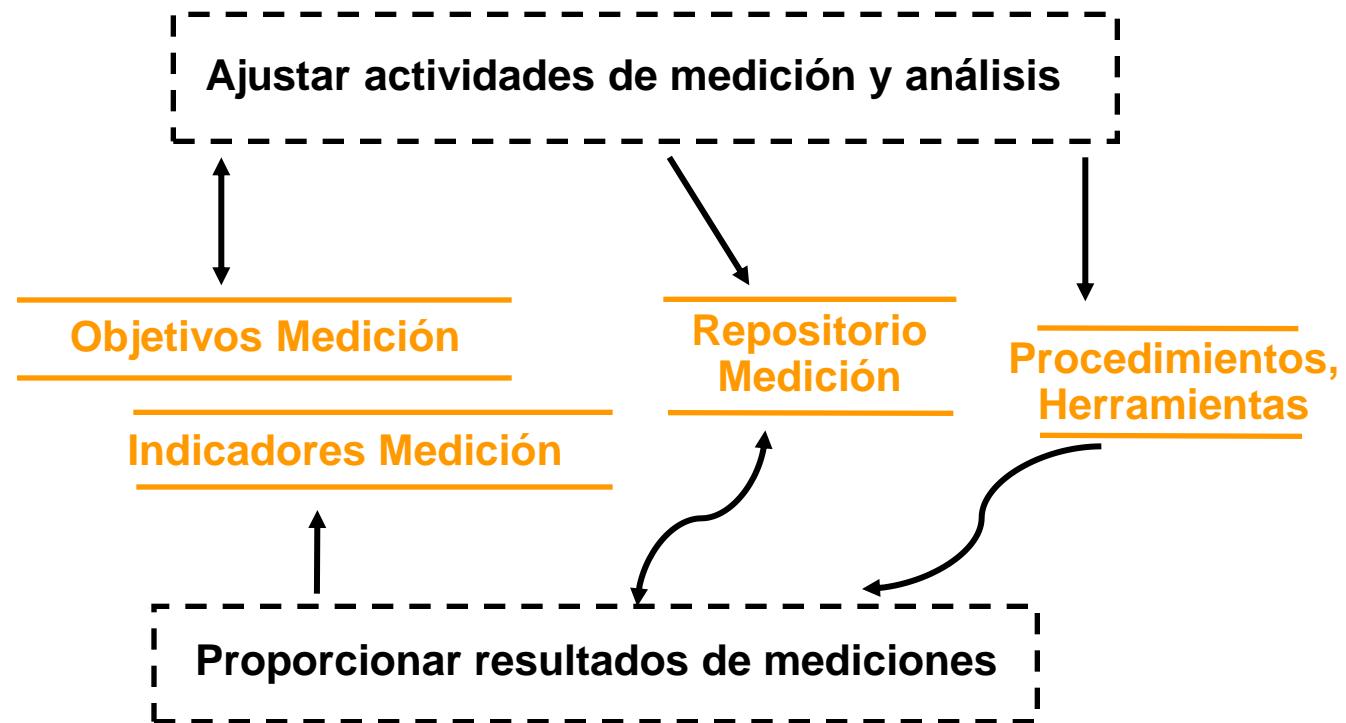
Propósito: Proporcionar un entendimiento objetivo de los procesos y los productos del trabajo asociado.



Áreas de proceso

Soporte: medición y análisis

Propósito: Desarrollar y mantener una capacidad para medir, utilizada para dar soporte a las necesidades de información de la gerencia.



Áreas de proceso

Gestión de procesos

Las áreas de procesos de soporte cubren las actividades de definición, planificación, recursos, desarrollo, implementación, monitorización, control, evaluación, medición y mejora de procesos.

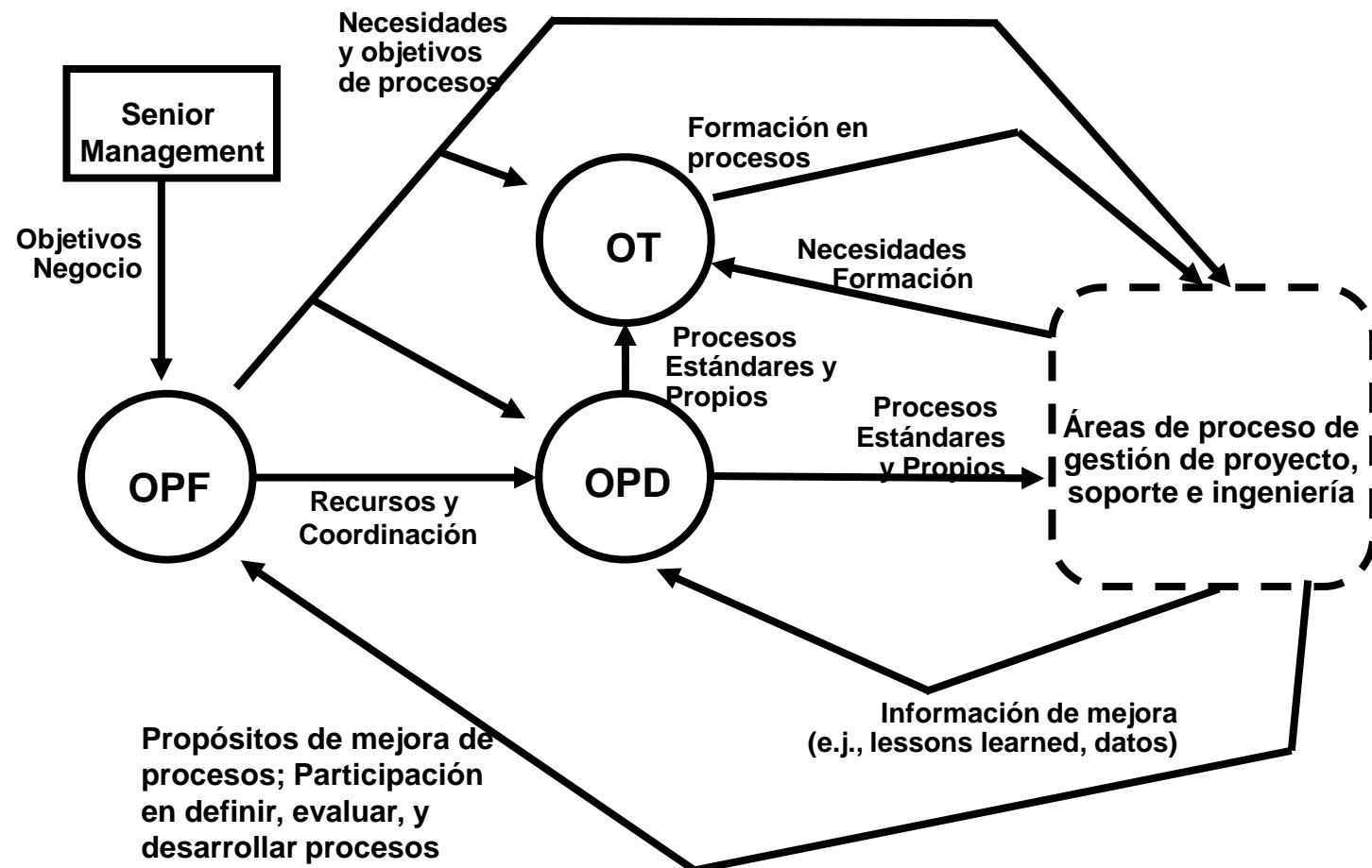
El modelo **CMMI SE/SW** incluye **cinco áreas de proceso** en gestión de procesos:

- **Definición de procesos**
- **Enfoque de procesos a la organización**
- **Formación**
- **Rendimiento de los procesos**
- **Innovación y desarrollo**



Áreas de proceso

Gestión de procesos: áreas básicas



Modelo de procesos

ISO / IEC 15504

1.0

AENOR

ER

Madurez
Software 2

ISO/IEC 15504

ISO/IEC 15504: Origen

Proyecto SPICE

En enero de 1993 la comisión ISO/IEC JTC1 aprobó un programa de trabajo para el desarrollo de un modelo que fuera la base de un futuro estándar internacional para la evaluación de los procesos del ciclo de vida del software.

Este trabajo recibió el nombre de proyecto SPICE (Software Process Improvement and Capability dEtermination), y en junio de 1995, con la publicación de su primer borrador, desde ISO fueron invitadas diferentes organizaciones para aplicarlo y valorar sus resultados.



Proyecto -> Instrucción Técnica -> Estándar

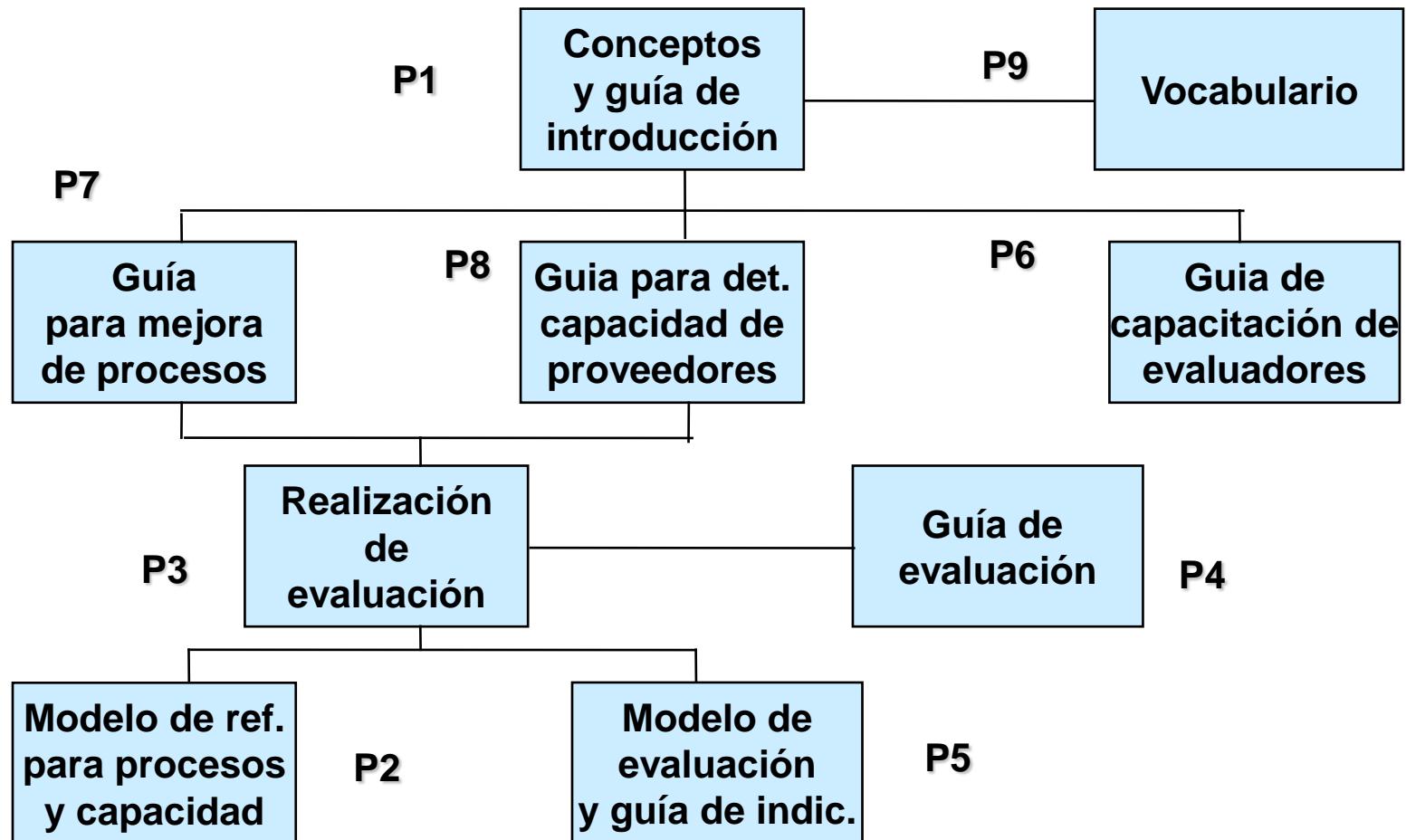
En 1998, pasada la fase de proyecto, y tras las primeras evaluaciones, el trabajo pasó a la fase de informe técnico con la denominación ISO/IEC TR 15504.

La instrucción técnica consta de 9 apartados, recogidos en volúmenes independientes, que se han ido publicando como redacción definitiva del estándar internacional ISO/IEC 15504 durante el periodo 2003-2005.

Ámbito de aplicación

- Cualquier organización de software que quiera establecer y mejorar su capacidad en adquisición, suministro, desarrollo, operación evolución y soporte de software.
- Independientemente de estructuras, filosofías de gestión, modelos de ciclo de vida de software, tecnologías o metodologías de desarrollo.

Estructura



Características

- Marco para métodos de evaluación, no es un método o modelo en sí.
- Comprende:
 - Evaluación de procesos
 - Mejora de procesos
 - Determinación de capacidad
- Alineado con ISO/IEC 12207 Information Technology Software Life Cycle Processes

Dimensiones del modelo

El modelo tiene una arquitectura basada en dos dimensiones:

■ Dimensión de proceso

Caracterizada por las declaraciones del propósito de un proceso, que son objetivos esenciales mensurables de un proceso.

■ Dimensión de capacidad de proceso

Caracterizada por una serie de atributos de proceso, aplicables a cualquier proceso, que representan características mensurables necesarias para gestionar un proceso y mejorar su capacidad.

Características

Dimensión de proceso

Agrupa los procesos en **tres grupos** correspondientes a los procesos del ciclo de vida que contienen **cinco categorías** de acuerdo al tipo de actividad.

Procesos primarios

- CUS: Cliente – Proveedor
- ENG: Ingeniería

Procesos de soporte

- SUP: Soporte

Procesos organizacionales

- MAN: Gestión
- ORG: Organización

Dimensión de proceso

CUS: Cliente - proveedor

La categoría CUS está formada por procesos que afectan directamente al cliente, soportan el desarrollo y la transición del software al cliente y permiten la correcta operación y uso del producto y/o servicio de software.

- **CUS.1 Proceso de adquisición**
 - CUS.1.1 Proceso de preparación de la adquisición
 - CUS.1.2 Proceso de selección de proveedor
 - CUS.1.3 Procesos de seguimiento de proveedor
 - CUS.1.4 Proceso de aceptación del cliente
- **CUS.2 Proceso de suministro**
- **CUS.3 Proceso de obtención de requisitos**
- **CUS.4 Proceso de operación**
 - CUS.4.1 Proceso de uso operacional
 - CUS.4.2 Proceso de soporte al cliente

Dimensión de proceso

ENG: Ingeniería

La categoría ENG está formada por procesos que directamente especifican, implementan o mantienen el producto de software, su relación con el sistema y documentación.

■ **ENG.1 Proceso de desarrollo**

- ENG.1.1 Proceso de análisis y diseño de requisitos de sistema.
- ENG.1.2 Proceso de análisis de requisitos de software.
- ENG.1.3 Procesos de diseño de software.
- ENG.1.4 Proceso de construcción de software.
- ENG.1.5 Proceso de integración de software.
- ENG.1.6 Proceso de prueba de software.
- ENG.1.7 Proceso de integración y prueba de sistema.

■ **ENG.2 Proceso de mantenimiento de software**

Dimensión de proceso

SUP: Soporte

La categoría SUP está formada por procesos que dan soporte al resto de procesos (incluidos los SUP), en distintos puntos del ciclo de vida del software.

- **SUP.1 Proceso de documentación**
- **SUP.2 Proceso de gestión de configuración**
- **SUP.3 Proceso de aseguramiento de calidad**
- **SUP.4 Proceso de verificación**
- **SUP.5 Proceso de validación**
- **SUP.6 Proceso de revisión conjunta**
- **SUP.7 Proceso de auditoría**

Dimensión de proceso

MAN: Gestión

La categoría MAN está formada por procesos utilizados en la gestión de cualquier tipo de proyecto o proceso en el ciclo de vida del software.

- **MAN.1 Proceso de gestión**
- **MAN.2 Proceso de gestión de proyecto**
- **MAN.3 Gestión de calidad**
- **MAN.4 Gestión de riesgos**

Dimensión de proceso

ORG: Organización

La categoría ORG está formada por procesos que establecen los objetivos de negocio de la organización.

- **ORG.1 Proceso de alineación organizacional.**
- **ORG.2 Proceso de mejora**
 - ORG.2.1 Proceso de definición de proceso.
 - ORG.2.2 Proceso de evaluación de proceso.
 - ORG.2.3 Proceso de mejora de proceso.
- **ORG.3 Proceso de gestión de RR.HH.**
- **ORG.4 Proceso de infraestructura**
- **ORG.5 Proceso de medición**
- **ORG.6 Proceso de reutilización**

Dimensión de proceso

Componentes de proceso

- **Identificador**

Identifica la categoría del proceso y el nº de secuencia en la categoría. Distingue entre procesos de primer y segundo nivel.

- **Nombre**

Frase descriptiva del contenido del proceso

- **Tipo**

Hay 5 tipos de proceso. 3 de primer nivel (básico, extendido y nuevo) y 2 de segundo nivel (componente, componente extendido)

- **Propósito**

Párrafo que establece el propósito del proceso indicando los objetivos globales de su ejecución.

- **Salidas**

Lista de resultados observables de la implementación exitosa del proceso

- **Notas**

Dimensión de capacidad

Capacidad de proceso: rango de resultados que espera obtenerse al seguir el proceso.

- ❑ Define una escala de medida para determinar la capacidad de cualquier proceso
 - ❑ Consta de seis niveles de capacidad
 - Nivel 0** Incompleto
 - Nivel 1** Realizado
 - Nivel 2** Gestionado
 - Nivel 3** Establecido
 - Nivel 4** Predecible
 - Nivel 5** En optimización
 - ❑ ...y un conjunto de atributos de procesos asociados al nivel de capacidad

Dimensión de capacidad

Niveles de capacidad y atributos

- **Nivel 0: Proceso Incompleto**
- **Nivel 1: Proceso Realizado**
- **Nivel 2: Proceso Gestionado**
 - PA 2.1 Gestión de realización
 - PA 2.2 Gestión productos
- **Nivel 3: Proceso Establecido**
 - PA 3.1 Definición de proceso
 - PA 3.2 Recursos de proceso
- **Nivel 4: Proceso Predecible**
 - PA 4.1 Medición
 - PA 4.2 Control de proceso
- **Nivel 5: Proceso en optimización**
 - PA 5.1 Cambio de proceso
 - PA 5.2 Mejora continua

Dimensión de capacidad

Medición de atributos

Los atributos de un proceso se evalúan con **N (Not)**, **P (Partially)**, **L (Largely)** y **F (Fully)**, siendo:

N	No alcanzado (0% a 15%). Escasa o ninguna evidencia de la consecución del atributo.
P	Parcialmente alcanzado (16% a 50%). Evidencia de un enfoque sistemático y de la consecución del atributo. Algunos aspectos de la consecución pueden ser impredecibles.
L	Ampliamente alcanzado (51% a 85%). Evidencia de un enfoque sistemático y de una consecución significativa del atributo. La realización del proceso puede variar en algunas áreas.
F	Totalmente alcanzado (86% a 100%). Evidencia de un enfoque completo y sistemático y de la consecución plena del atributo.

Ciclo de vida del software

1.0



Introducción

En este tema se tratan los siguientes conceptos:

- Ciclo de vida del software.
- Procesos del ciclo de vida.
- Modelos de ciclo de vida.

Ciclo de vida del software

El marco del ciclo de vida del software cubre desde la conceptualización de las ideas iniciales del producto hasta el fin de su uso (retirada).

ISO/IEC 12207 1995

Desde el punto de vista del estándar (v. Introducción a la Ingeniería del Software) un proceso es un conjunto de actividades y tareas relacionadas, que al ejecutarse de forma conjunta transforman una entrada en una salida.

Procesos primarios del ciclo de vida del software

12207 define los siguientes procesos primarios en el ciclo de vida del software:

ADQUISICIÓN

Proceso global que sigue el adquiriente para obtener el producto.

SUMINISTRO

Proceso global que sigue el suministrador para proporcionar el producto.

DESARROLLO

Proceso empleado por el suministrador para el diseño, construcción y pruebas del producto.

OPERACIÓN

Proceso seguido por el operador en el “día a día” para el uso del producto.

MANTENIMIENTO

Proceso empleado para mantener el producto, incluyendo tanto los cambios en el propio producto como en su entorno de operación.

Procesos de soporte del ciclo de vida del software

El estándar 12207 identifica los procesos de soporte que pueden ser utilizados desde un proceso primario, o incluso desde otro proceso de soporte.

Los procesos de soporte son:

DOCUMENTACIÓN

Actividades empleadas para registrar información específica empleada por otros procesos.

GESTIÓN DE LA CONFIGURACIÓN

Actividades empleadas para mantener un registro de los productos generados en la ejecución de los procesos.

ASEGURAMIENTO DE LA CALIDAD

Actividades empleadas para garantizar de forma objetiva que el producto y los procesos asociados son conformes a los requisitos documentados y a las planificaciones.

VERIFICACIÓN

Actividades empleadas para verificar el producto.

VALIDACIÓN

Actividades empleadas para validar el producto.

Procesos de soporte del ciclo de vida del software

REUNIONES DE REVISIÓN

Reuniones empleadas por las dos partes para evaluar el estado del producto y de las actividades.

AUDITORÍAS

Actividades para determinar que el proyecto cumple con los requisitos, planes y contratos.

RESOLUCIÓN DE PROBLEMAS

Actividades para analizar y resolver problemas relativas al proyecto, sea cual sea su fuente y naturaleza.

Procesos organizacionales

El estándar 12207 identifica los procesos que deben realizarse en el contexto de la organización que va a ejecutar el proyecto.

Normalmente estos procesos se aplican de forma común sobre múltiples proyectos. De hecho las organizaciones más maduras los identifican e institucionalizan.

GESTIÓN

Describe las actividades de gestión de la organización, incluyendo también la gestión de proyectos.

INFRAESTRUCTURA

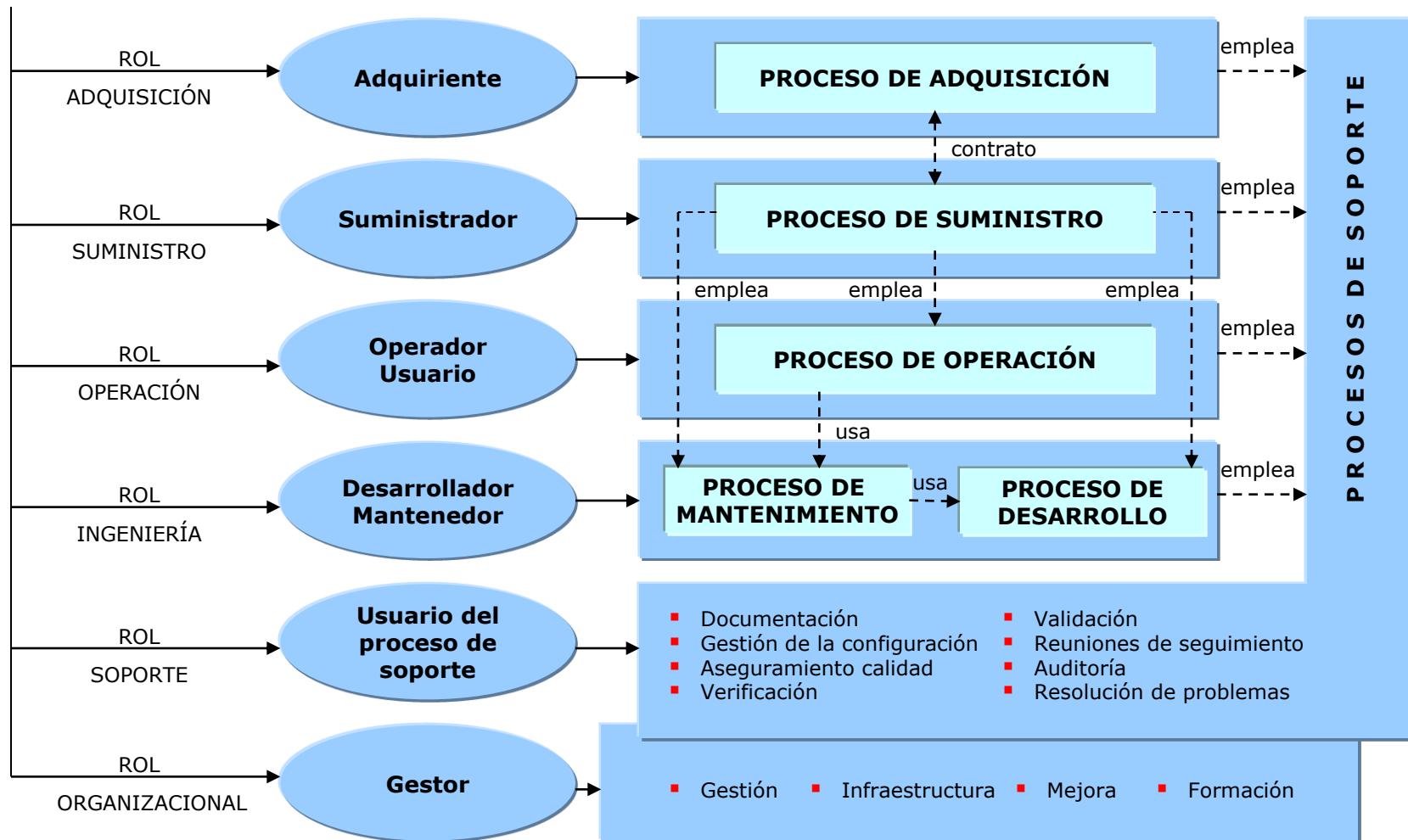
Actividades necesarias para que puedan realizarse otros procesos del ciclo de vida. Incluye entre otros el capital y el personal.

MEJORA

Actividades realizadas para mejorar la capacidad del resto de procesos.

FORMACIÓN

VISIÓN GENERAL DE LOS PROCESOS, RELACIONES Y ROLES



Ciclos de desarrollo y patrones de gestión de proyectos

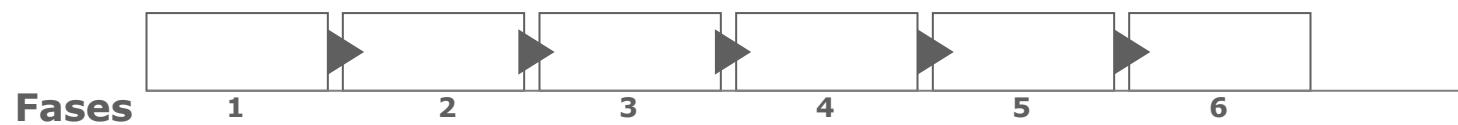
1.0



El trabajo

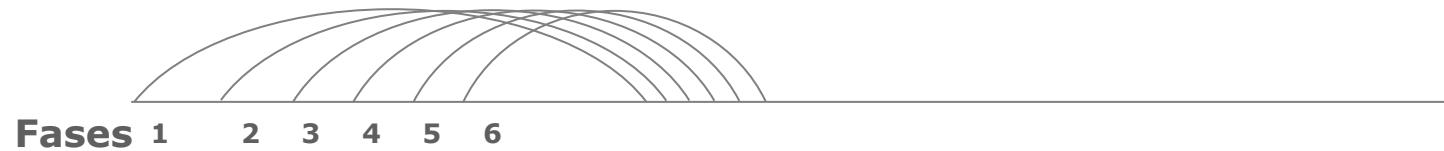
Se puede realizar de forma

SECUENCIAL (cascada)



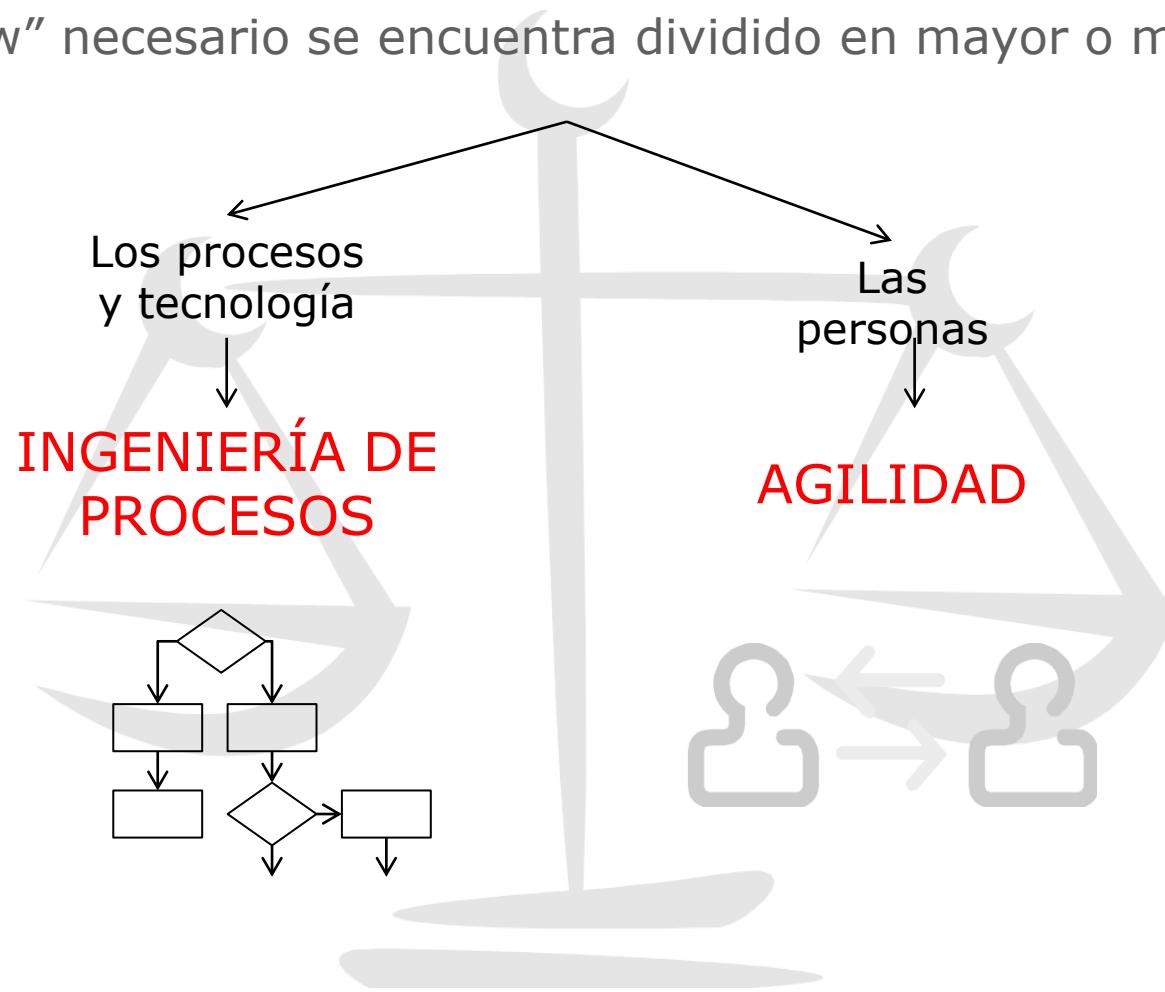
O

CONCURRENTE



El conocimiento

o “know how” necesario se encuentra dividido en mayor o menor % en:

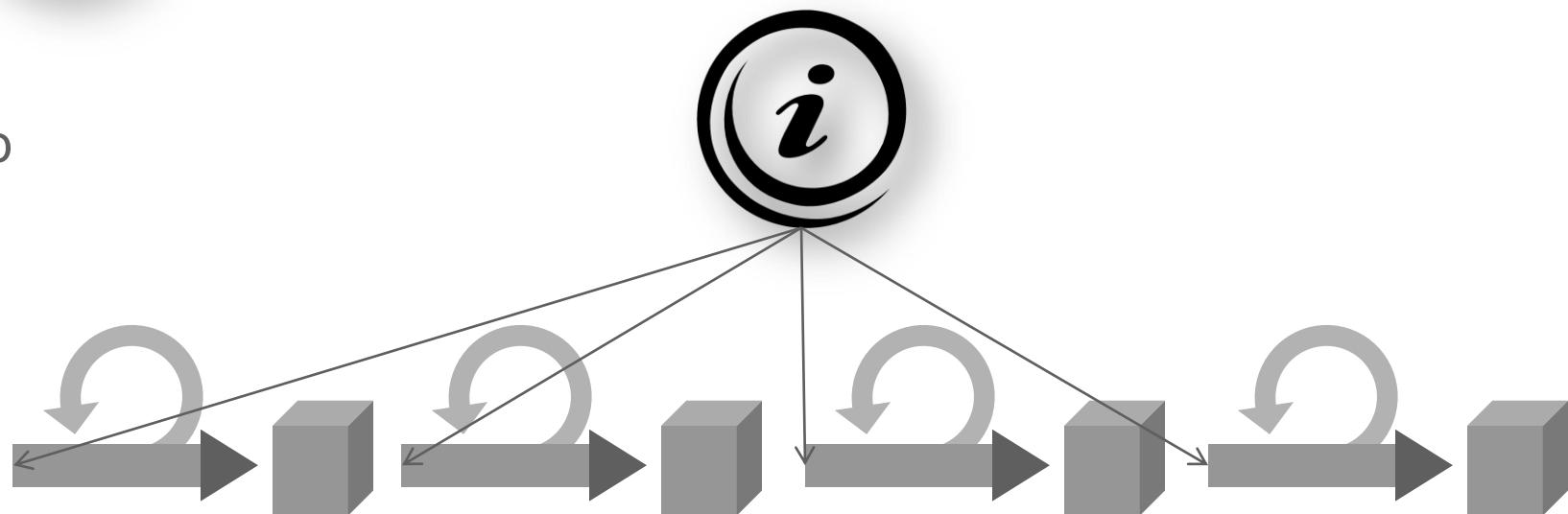


El desarrollo

Se realiza de forma

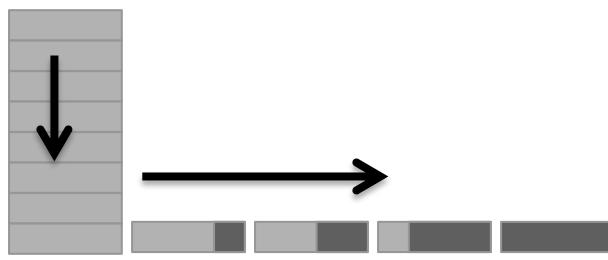


O

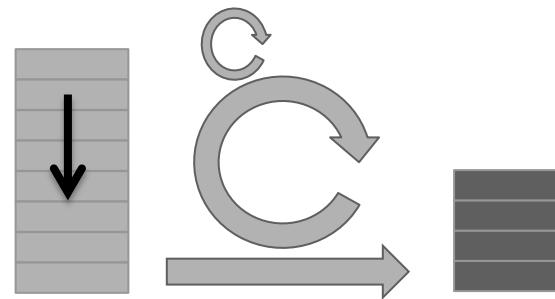


En un DESARROLLO INCREMENTAL

Las entregas se pueden producir con secuencia:

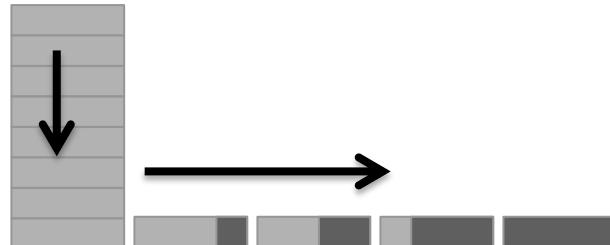


CONTINUA

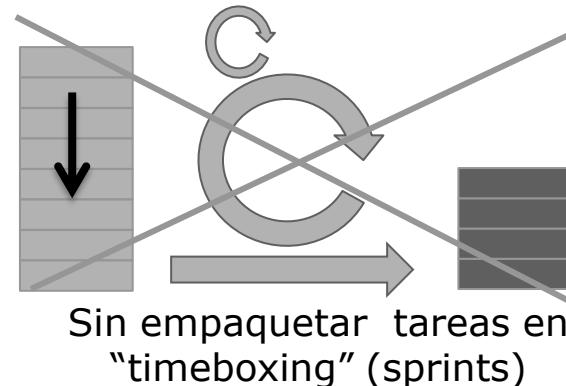


ITERATIVA

Para gestionar un DESARROLLO INCREMENTAL



Con un flujo continuo de trabajo



Sin empaquetar tareas en
"timeboxing" (sprints)

Son apropiadas las **técnicas de gestión visual kanban** para evitar los cuellos de botella y los tiempos muertos.

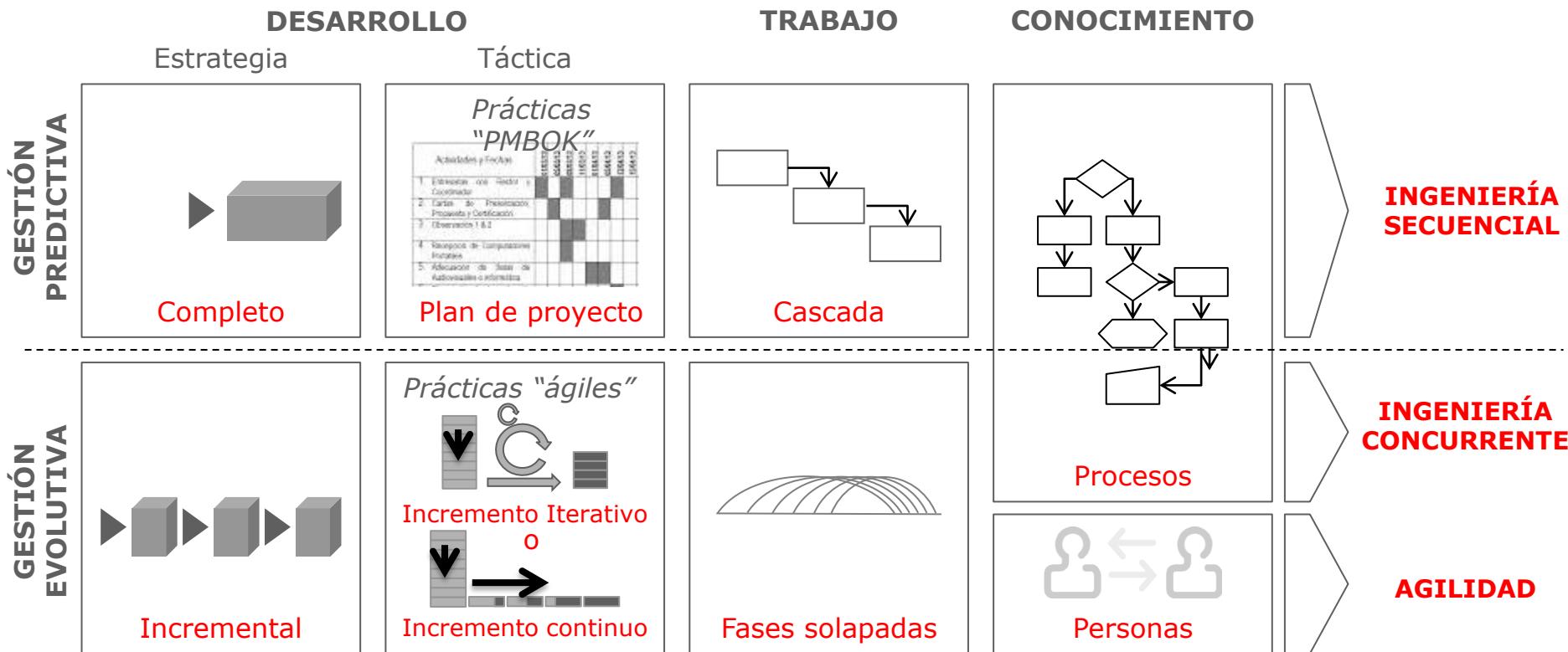
Ajustándolas con criterios de flexibilidad a las circunstancias de nuestro trabajo y equipo



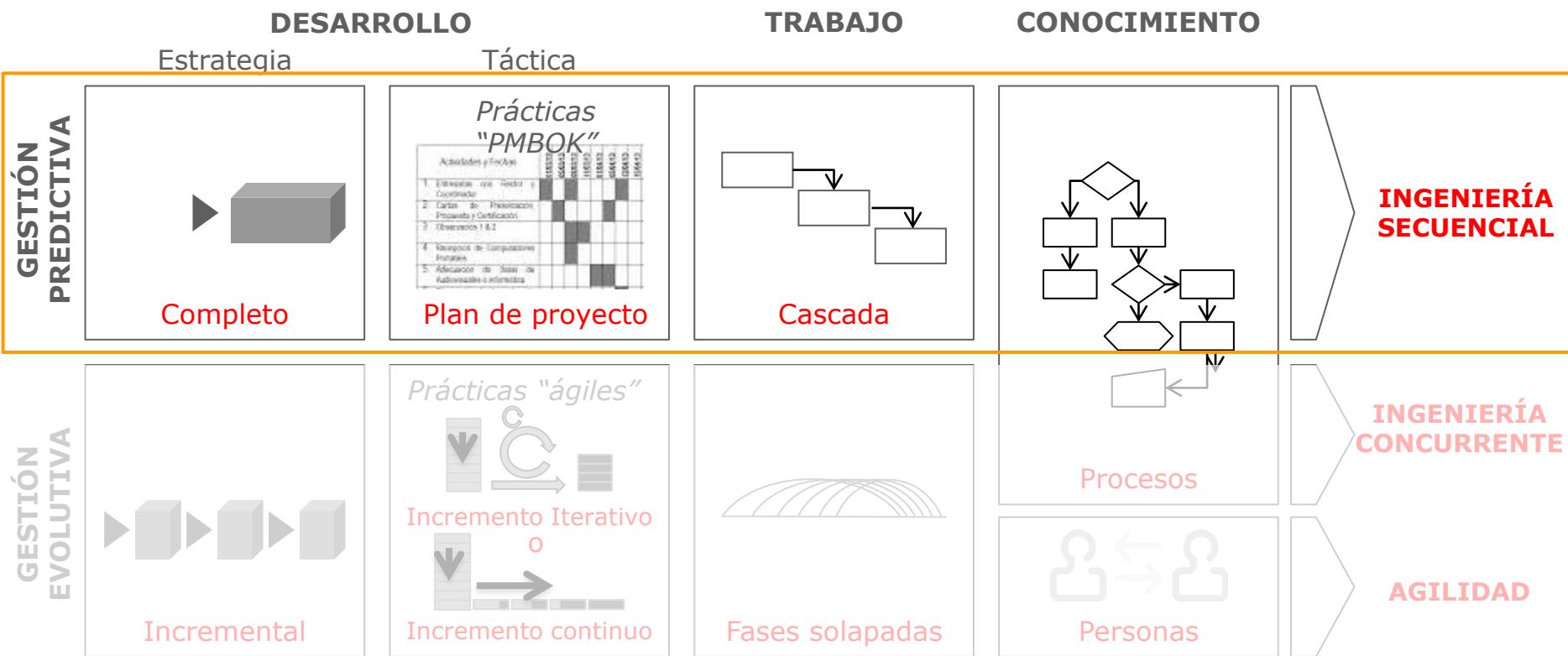
Trabajo {
Secuencial
Libre

Equipo {
Polivalente
Especialistas

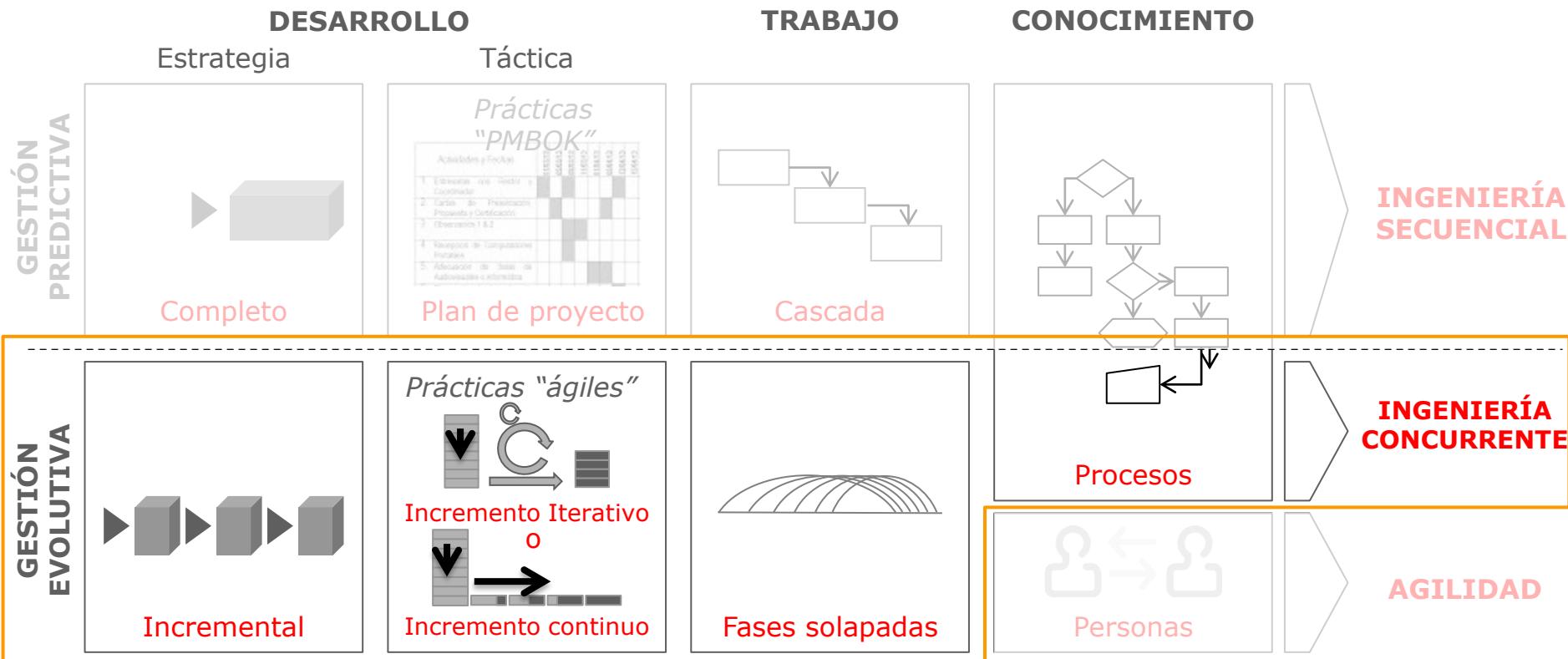
Gestión de proyectos: diagrama de conceptos



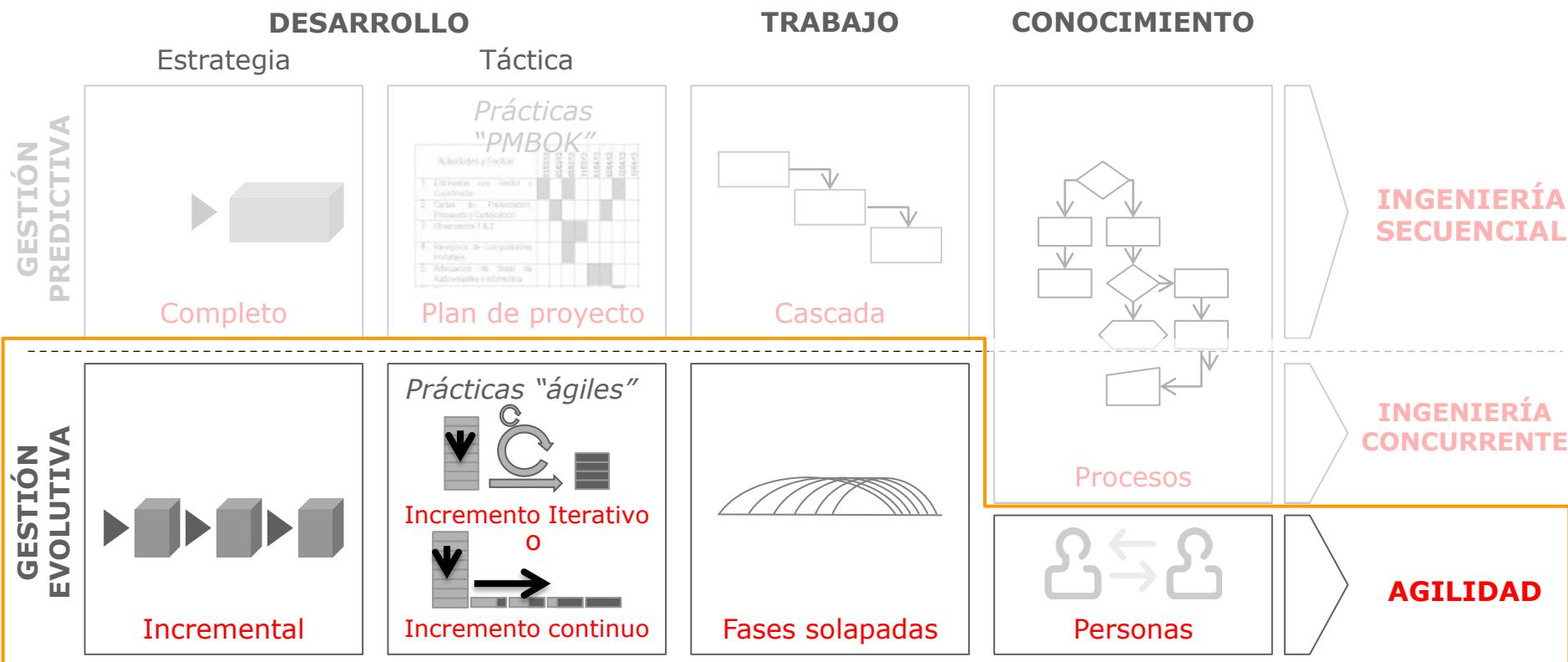
Gestión predictiva



Ingeniería concurrente



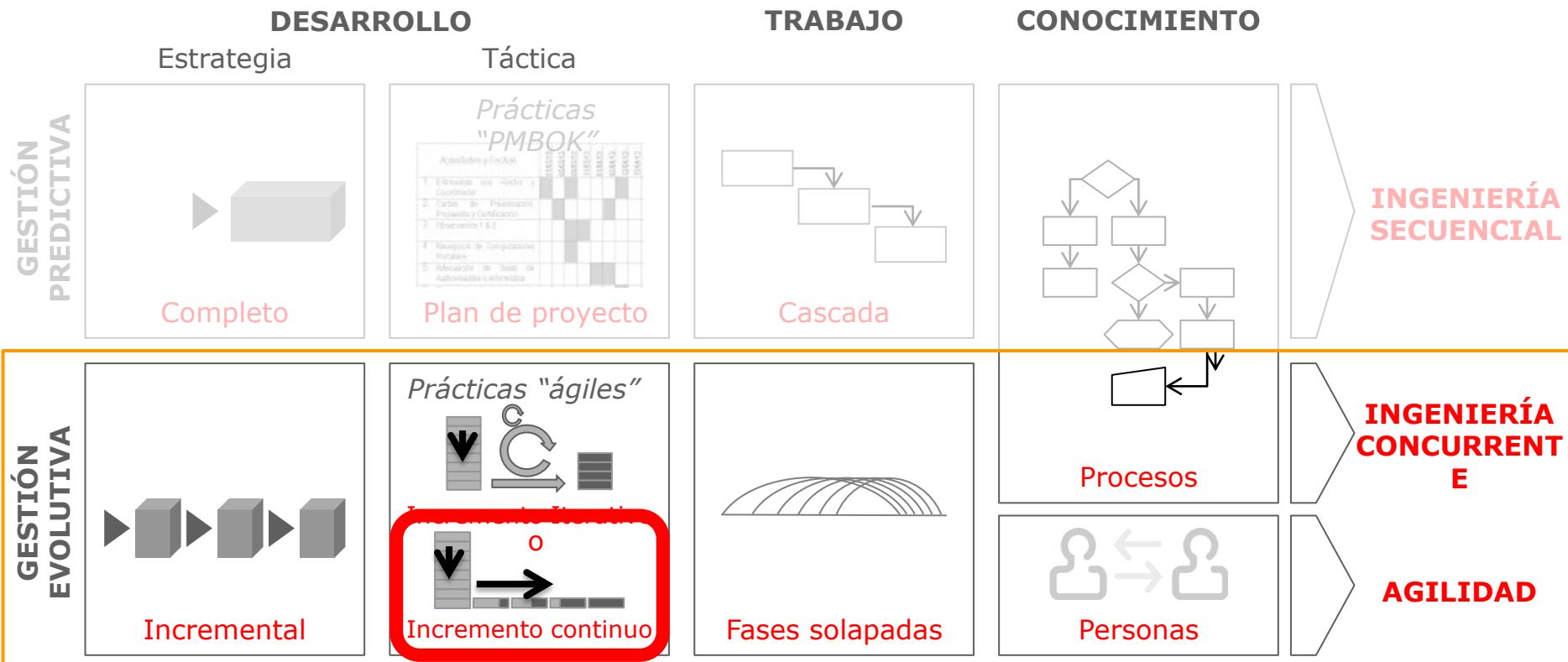
Agilidad



Prácticas Scrum



Prácticas kanban



Elección del modelo de gestión y ciclo de desarrollo

Al iniciar el proyecto, el responsable de la arquitectura de procesos debe realizar los siguientes pasos:

- Análisis de las circunstancias ambientales del proyecto.
- Diseño del modelo específico de ciclo de vida para el proyecto (sobre las bases de los diseños más apropiados, para el desarrollo y la evolución del sistema de software).
- Mapeo de las actividades sobre el modelo.
- Desarrollo del plan para la gestión del ciclo de vida del proyecto.

Debe considerar aspectos como:

- Posibilidad de descomposición del sistema en subsistemas de software, con agendas y entregas diferenciadas.
- Estabilidad esperada de los requisitos.
- Novedad del proceso o procesos gestionados por el sistema en el entorno del cliente.
- Criticidad de las agendas y presupuestos.
- Grado de complejidad del interfaz de operación, criticidad de la usabilidad.
- Grado de conocimiento y familiaridad con el entorno de desarrollo, componentes externos empleados, etc.

Los requisitos del software en la gestión de proyectos predictiva

1.0



Importancia de los requisitos

Para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, **decepcionará al usuario y desprestigiará al que lo ha desarrollado.**

Roger S. Pressman Ingeniería del Software Mc Graw Hill 1995

La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan ardua como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas software. **Ninguna otra parte del trabajo puede perjudicar tanto el resultado** final si se realiza de forma errónea. Ninguna otra parte es tan **difícil de rectificar posteriormente**.

Frederick P. Brooks, Jr., The Mythical Man-Month, Addison-Wesley, 1995.

Importancia de los requisitos

■ ¿Qué porcentaje de proyectos concluyen con éxito?

Un estudio realizado por Standish Group analizó el desarrollo de **8000 proyectos de software**, realizados por **350 empresas diferentes** y concluyó que **sólo el 16% de los proyectos de software se realizan con éxito**.

El estudio identificó como **principales causas de los problemas**:

- **Requisitos deficientes**
- La planificación de **agendas y estimaciones** de costes **no** se realizaron **en base a los requisitos**
- Deficiencias en la aplicación de procesos y desconocimiento del ciclo de vida del proyecto

Los criterios para determinar el éxito de un proyecto son:

- Sin desviaciones en las fechas previstas.
- Sin desviaciones en los costes estimados.
- Que el producto final cubra las expectativas y necesidades del cliente.
- Que funcione correctamente.

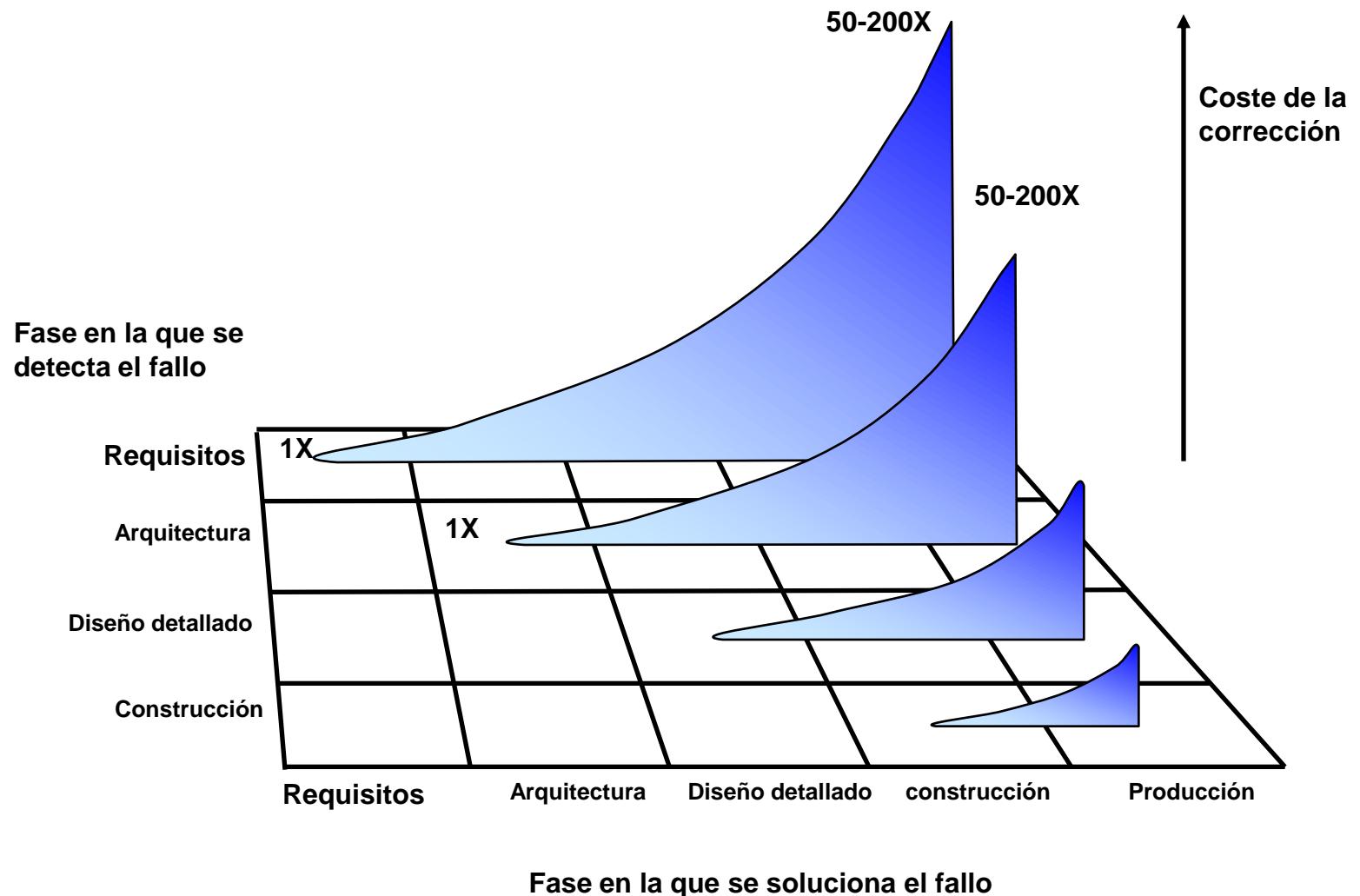
Importancia de los requisitos

■ ¿Por qué fracasan los proyectos?



- Requisitos incompletos: 13%
- Cambios en requisitos: 9%
- No implicación de usuarios: 12%
- Expectativas no realistas: 10%
- Producto no necesario: 8%
- TOTAL: 52%

Importancia de los requisitos



Importancia de los requisitos

- Sus defectos repercuten en todas las fases

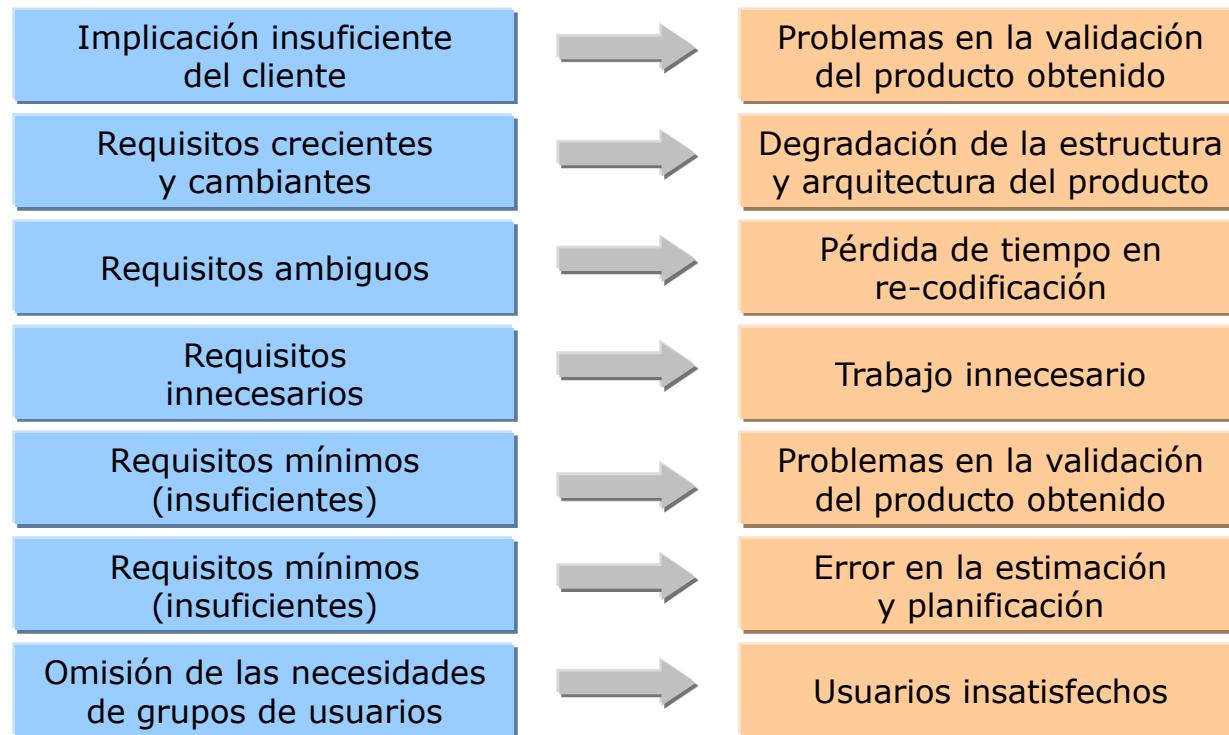


Los errores en los requisitos se comportan como una enfermedad contagiosa que siempre repercuten en todas las fases del proyecto.

- Estimación: No es posible estimar con rigor costes y recursos necesarios para desarrollar algo que no se conoce.
- Planificación: No se puede confiar en la planificación para el desarrollo de algo que no se sabe bien como es.
- Diseño: Los errores en requisitos, las modificaciones frecuentes, las deficiencias en restricciones o futuras evoluciones, producen arquitecturas que más tarde se confirmarán como erróneas y serán modificadas.
- Construcción: Las deficiencias en los requisitos obligan a programar en ciclos de prueba y error que derrochan horas y paciencia de programación sobre patrones de "recodificación continua" y "programación heroica".
- Validación y verificación: Terminado el desarrollo del sistema, si las especificaciones tienen errores de bulto, o peor aún, no están reflejadas en una especificación de requisitos, no será posible validar el producto con el cliente.

Importancia de los requisitos

■ Los defectos comunes en los requisitos y sus consecuencias.



Importancia de los requisitos

■ Los defectos comunes en los requisitos y sus consecuencias.

Implicación insuficiente del cliente

Algunos clientes no comprenden la importancia de trabajar con rigor en la obtención de los requisitos, para garantizar la calidad de los resultados.

También es frecuente que los desarrolladores prefieran comenzar a trabajar en la construcción del sistema, porque les resulta más atractivo que reunirse con el cliente.

Hay situaciones en las que resulta difícil encontrar representantes del cliente que conozcan a fondo el problema, o que por tratarse de un sistema o negocio nuevo, nadie en el entorno del cliente tenga claras todas las funcionalidades que se necesitan.

Requisitos crecientes y cambiantes

"Independientemente del punto del ciclo de vida en que nos encontremos, el sistema cambiará y la tendencia al cambio persistirá a lo largo de todo el ciclo de vida"

Software Configuration Management, Prentice-Hall, 1980.

Es normal que los requisitos evolucionen durante el desarrollo del sistema, pero los cambios deben partir de una descripción inicial correcta, y gestionarse convenientemente, midiendo su impacto en la planificación del proyecto, y consensuándolo con todos los participantes.

La evolución de los requisitos durante el desarrollo de los proyectos puede incrementar o modificar funcionalidades ya implementadas, desbordando los costes y agendas planificados.

Importancia de los requisitos

■ Los defectos comunes en los requisitos y sus consecuencias.

Requisitos crecientes y cambiantes

Partir de una especificación de requisitos incompleta incrementará el número de modificaciones que sufrirá el proyecto durante el desarrollo. Si los desarrolladores han diseñado un sistema que no corresponde con las expectativas del cliente, la introducción sistemática (generalmente con agendas apretadas, o sin modificar las agendas iniciales), generará parches de programación, con inserción de código adicional que puede trastocar principios básicos de diseño y degradar la arquitectura del sistema obteniendo finalmente un producto con serias deficiencias técnicas.

Requisitos ambiguos

La ambigüedad es un defecto habitual de las descripciones de requisitos.

Si lectores diferentes obtienen interpretaciones diferentes, o si un lector puede interpretar los requisitos de formas diferentes, éstos son ambiguos.

La ambigüedad crea expectativas diferentes entre las partes del proyecto, y hace que los desarrolladores programen funcionalidades que no se ajustan a lo que los usuarios necesitan.

La consecuencia inevitable de este problema es la re-programación

La reprogramación puede consumir más del 40% del coste total de un desarrollo y se ha estimado que hasta un 85% de las revisiones pueden deberse a errores en los requisitos^[1].

Para evitarla hay que confirmar que se comparte la visión obtenida con la que tienen las diferentes fuentes de requisitos, y que los distintos participantes obtienen la misma interpretación de la documentación de requisitos.

[1] Calculating the Return of Investment from More Effective Requirement Management”, Leffingwell, Dean. 1997.

Importancia de los requisitos

■ **Los defectos comunes en los requisitos y sus consecuencias.**

Requisitos innecesarios

Es frecuente la tendencia de algunos desarrolladores a incluir funcionalidades que no figuran en la especificación de requisitos, con la suposición de que los usuarios lo agradecerán. Muchas veces los usuarios no les encuentran utilidad y quedan finalmente programadas pero sin uso, suponiendo un coste de desarrollo innecesario.

Las sugerencias y posibilidades aportadas por el equipo de desarrollo pueden descubrir mejoras importantes para el cliente o los usuarios, pero no deben implementarse sin consultarlas y validarlas previamente.

Desde el punto de vista del equipo de desarrollo la mejor perspectiva es respetar la sencillez y funcionalidad, y no ir más allá de los requisitos, sin la aprobación del cliente.

También es frecuente que el cliente pida funcionalidades que a primera vista pueden parecer necesarias, pero que en realidad no añaden funcionalidad al producto, y que sin embargo suponen un esfuerzo importante de desarrollo. Identificar estas funcionalidades, y pensar dos veces si realmente se necesitan puede ahorrar trabajo innecesario

Importancia de los requisitos

■ **Los defectos comunes en los requisitos y sus consecuencias.**

Requisitos insuficientes (mínimos)

Muchas veces el cliente tiene tan sólo el concepto general del producto que desea, y no comprende por qué es tan importante detallar los requisitos.

La tentación en estos casos es partir de una descripción mínima, o incluso de una explicación verbal, e ir preguntando y revisando a los programadores conforme el desarrollo avanza.

Esta forma de trabajo puede resultar apropiada sólo para la construcción de sistemas experimentales o prototipos, pero en general suele terminar con la frustración de los desarrolladores y el desconcierto y desesperación del cliente.

Este planteamiento también genera la situación muy frecuente de contar a los desarrolladores la idea general de un nuevo producto, para pedirles una estimación del tiempo necesario para desarrollarlo.

Normalmente la visión general, sin descender a los detalles que implica, genera previsiones optimistas que terminarán desbordadas al descubrir durante el desarrollo las implicaciones que pasan inadvertidas en la concepción inicial.

Las estimaciones prematuras, basadas en información limitada pueden fácilmente desbordarse en más del doble. Siempre que sea preciso ofrecer valoraciones previas es conveniente ofrecer varias posibilidades (mejor caso, caso probable, peor caso), o incluir un porcentaje posible de error probable.

Importancia de los requisitos

■ **Los defectos comunes en los requisitos y sus consecuencias.**

Omisión de las necesidades de algunos grupos de usuarios

Entre los usuarios de un sistema es frecuente que se incluyan grupos de personas con necesidades diferentes, que empleen funcionalidades distintas, e incluso que presenten diversos perfiles de experiencia y conocimientos.

Al identificar las posibles fuentes de requisitos hay que localizar todos los posibles usuarios y obtener información de sus características, necesidades y expectativas

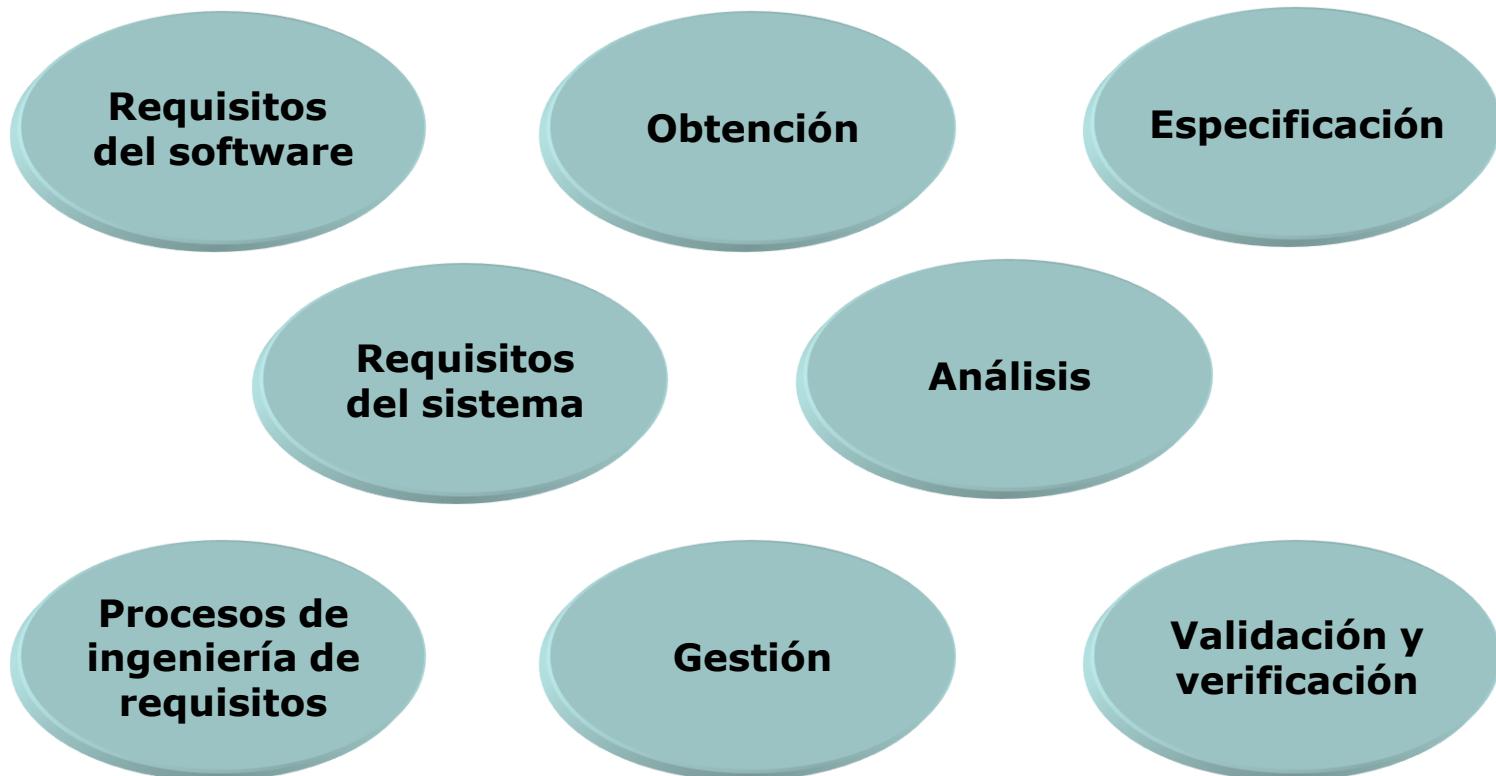
Importancia de los requisitos

■ **Beneficios de los buenos requisitos.**

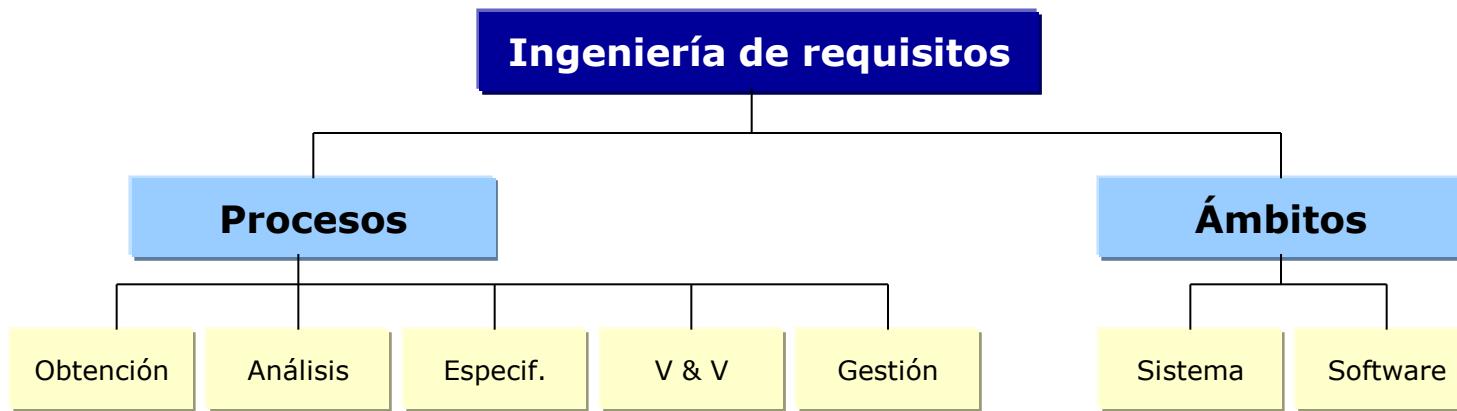
- Acuerdo entre desarrolladores, clientes y usuarios sobre el trabajo que debe realizarse.
Unos requisitos bien elaborados y validados con el cliente evitan descubrir al terminar el proyecto que el sistema no era lo que se pedía.
- Acuerdo entre desarrolladores, clientes y usuarios sobre los criterios que se emplearán para su validación.
Resulta muy difícil demostrar al cliente que el producto desarrollado hace lo que el pidió si su petición no está documentada y validada por él.
- Base objetiva para la estimación de recursos (coste, personal en número y competencias, equipos y tiempo)
Si los requisitos no comprenden necesidades reales, las estimaciones no dejan de ser meras apuestas. Las estimaciones en el fondo son cálculos de probabilidad que siempre implican un margen de error; por esta razón disponer de la mayor información posible reduce el error.
- Concreción de los atributos de calidad (ergonomía, mantenibilidad, etc.)
Más allá de funcionalidades precisas, los requisitos recogen atributos de calidad necesarios que en ocasiones son desconocidos por los desarrolladores, produciendo finalmente sistemas sobredimensionados o con serias deficiencias de rendimiento.
- Eficiencia en el consumo de recursos: reducción de la re-codificación, reducción de omisiones y malentendidos.
Tener un conocimiento preciso de lo que hay que hacer evita la prueba y error, repetición de partes ya desarrolladas, etc.

Conceptos

■ Conceptos clave.



Conceptos



La ingeniería del software y la ingeniería de requisitos son ingenierías muy recientes. En la actualidad acaba de cerrarse la versión 1.0 de SWEBOK, que constituye el esfuerzo más serio y consensuado hasta la fecha para definir las áreas de conocimiento que la integran. En este estado de cosas no es extraño encontrar que, diferentes autores clasifican o presentan los conceptos clave de forma diferente, si bien los conceptos básicos siempre son los mismos: Obtención, análisis, especificación, validación y verificación y gestión.

Así por ejemplo, Karl Wiegers centra su interés clasificatorio en la diferencia entre el desarrollo de los requisitos y su posterior gestión.

SWEBOK plantea una representación esquemática plana (no distingue entre gestión y desarrollo) y centra su interés sólo en los requisitos del software.

IEEE carga el peso de la clasificación en la diferencia entre requisitos del sistema y del software.

Nuestro punto de vista contempla las 5 áreas clave, que se trabajan tanto en el ámbito de los requisitos del sistema como en los requisitos del software.

Los requisitos del software en la gestión de proyectos predictiva

Conceptos



Obtención (*elicitation*)

El primer paso consiste en conocer y comprender las necesidades y problemas del cliente.

En la obtención se identifican todas las fuentes de requisitos implicadas en el sistema y, en función de las características del entorno y del proyecto se emplean las técnicas más apropiadas para la identificación de las necesidades que deben satisfacerse.

Análisis

Una vez obtenida la información necesaria del entorno, es necesario sintetizarla, darle prioridades, analizar posibles contradicciones o conflictos, descomponer el sistema y distribuir las necesidades de cada parte, delimitar los límites del sistema y definir su interacción con el entorno.

Conceptos

Especificación

Cuando ya se conoce el entorno del cliente y sus necesidades, es necesario plasmarlas en forma de requisitos en los documentos que sirven de base de entendimiento y acuerdo entre cliente y desarrollador, y que establecerán tanto la guía desarrollo como los criterios de validación del producto final.

Documentar los requisitos es la condición más importante para gestionarlos correctamente.

Verificación y validación

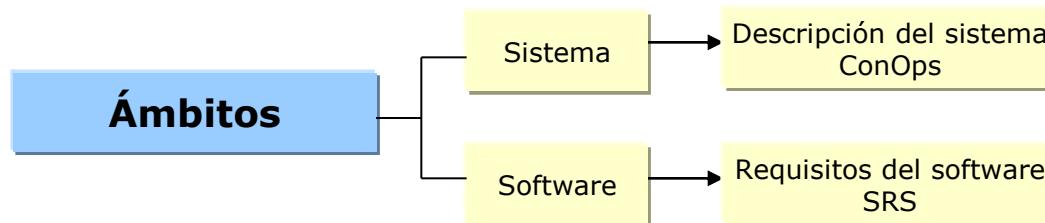
Los requisitos deben ser formal y técnicamente correctos (verificación), y satisfacer las necesidades del sistema, sin omitir ninguna ni incluir funcionalidades innecesarias (validación).

El significado de estos dos términos genera confusiones habitualmente. El criterio básico que los diferencia es que verificación se refiere a la calidad formal, en este caso de los documentos de requisitos (no son ambiguos, no son incompletos, son posibles, verificables, etc.) y validación comprende la adecuación en el entorno de producción, en el caso de la documentación de requisitos, la conformidad por parte del cliente de que reflejan lo que él quiere.

Gestión

Los requisitos cambiarán durante el desarrollo del sistema, y es necesario poder trazar en cada cambio todas las partes afectadas, así como poder medir el impacto que cada modificación implica en la planificación del proyecto.

Conceptos



■ Descripción del sistema

Documento, también denominado ConOps y normalizado en el estándar IEEE Std. 1362-1998.

Definición:

Documento dirigido a los usuarios, que describe las características de un sistema propuesto, desde el punto de vista del usuario. La Descripción del Sistema es el medio de comunicación que recoge la visión general, cualitativa y cuantitativa de las características del sistema; compartido por la parte cliente y desarrolladora.

Conceptos

■ Propósito de la descripción del sistema

El desarrollo de la Descripción del Sistema proporciona una actividad de análisis y un documento que tiene la función de enlace entre las necesidades del usuario, y las especificaciones técnicas del desarrollo.

La Descripción del sistema proporciona:

- La descripción de las necesidades operacionales del usuario sin entrar en detalles técnicos.
- La documentación de las características del sistema y las necesidades operacionales del usuario, de forma que puedan ser verificadas sin requerir conocimientos técnicos.
- El documento que recoge los deseos del usuario, sin requerir una cuantificación medible. Por ejemplo, el usuario puede indicar que desea que los tiempos de respuesta de las consultas sean rápidos, y las razones de su deseo, sin necesidad de cuantificar esos términos. Más adelante, el desarrollo y análisis de los requisitos del sistema, el analista concretará y cuantificará esos deseos.
- El documento en el que, comprador y suministrador, reflejan las posibles estrategias de solución, y las restricciones que deben respetarse.

Descripción del sistema

■ Propósito del estándar IEEE 1362

Ofrece un formato y contenidos para la confección de las descripciones de sistema en los desarrollos y modificaciones de sistemas intensivos de software.

El estándar no especifica técnicas exactas, sino que proporciona las líneas generales que deben respetarse. No es por tanto un modelo final, sino una guía de referencia sobre la que cada organización debe desarrollar sus propias prácticas y procedimientos para preparar y actualizar su documentación con las descripciones de los sistemas.

Las partes esenciales de un ConOps son:

Punto 3: descripción del sistema existente.

Punto 4: justificación del desarrollo o de la modificación,

Punto 5: Descripción del sistema propuesto.

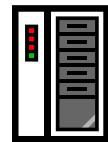
Los proyectos de tamaño pequeño requieren descripciones de sistema menos formales, pero no por su reducido tamaño debe ignorarse.

Si el proyecto de software forma parte de un proyecto mayor, la descripción del sistema de software puede ser un documento separado, o ir incluido en la descripción del sistema completo.

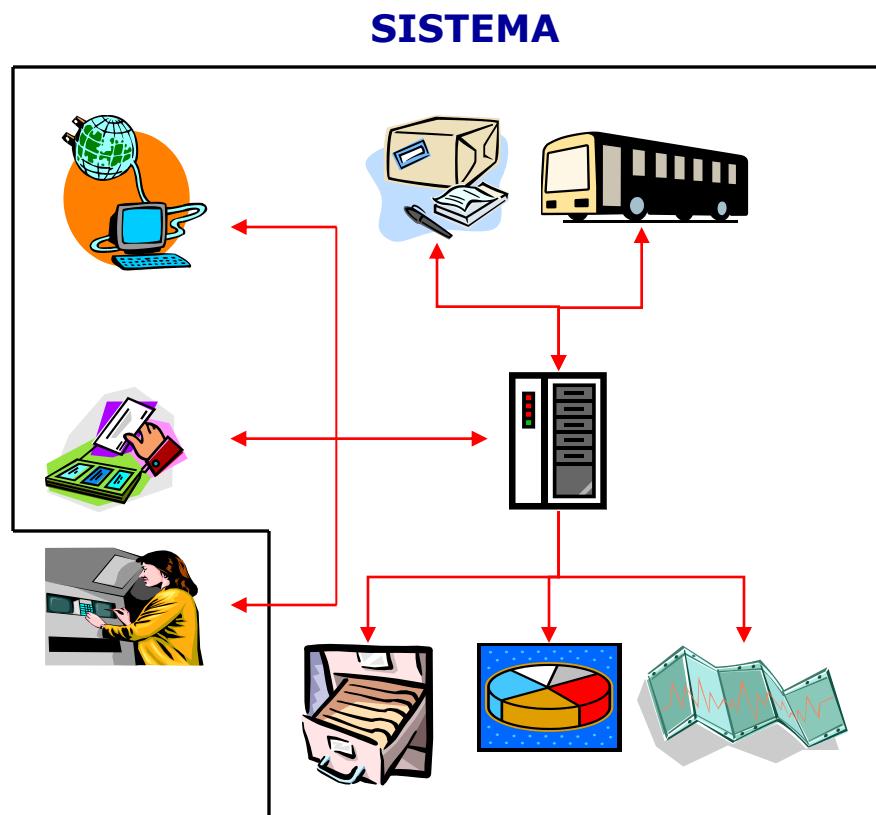
El estándar puede aplicarse a todos los tipos de sistemas de software: sólo software, intensivos de software o software/ hardware/personas. Aunque los conceptos del estándar también podrían aplicarse a sistemas de hardware, esta no es su finalidad.

El estándar identifica los elementos que al menos debe incluir una Descripción del sistema. El usuario puede incorporar otros elementos, agregando cláusulas y sub-cláusulas.

Descripción del sistema



Descripción del sistema



EVOLUCIÓN PREVISTA



Especificación de requisitos del software (SRS)

Un documento SRS es la especificación de las funciones que realiza un determinado producto de software, programa o conjunto de programas en un determinado entorno.

El documento de especificación de requisitos puede desarrollarlo personal representativo de la parte suministradora, o de la parte cliente; si bien es aconsejable la intervención de ambas partes.

Los aspectos básicos que una descripción de requisitos debe cubrir son:

- **Funcionalidad.** Descripción de lo que el software debe hacer.
- **Interfaces externos.** Cómo debe interactuar el software con las personas, el sistema de hardware, o con otros sistemas (software y hardware).
- **Rendimiento.** Indicación de la velocidad, disponibilidad, tiempos de respuesta, tiempos de recuperación, tiempos de determinadas funciones.
- **Atributos.** Consideraciones de portabilidad, corrección, mantenibilidad, seguridad, etc.
- **Restricciones de diseño en la implementación.** Indicación de las restricciones que puedan afectar por la necesidad de sometimiento a estándares, lenguajes, políticas de integridad de bases de datos, límites de recursos disponibles para el desarrollo, sistema operativo, etc.

Las especificaciones de requisitos de software deben evitar incluir requisitos de diseño o de proyecto.

Especificación de requisitos del software (SRS)

- ~~No deben incluir restricciones de diseño gratuitas~~

Deben especificar el QUÉ, no el CÓMO

Una descripción de requisitos del sistema limita las alternativas de diseño posibles, pero esto no significa que deba decidir cuál debe ser la solución de diseño del sistema.

La especificación de requisitos de software determina qué funcionalidades deben realizarse, qué datos deben generarse en cada resultado, en qué lugar y quién los debe producir. La SRS debe centrarse en los servicios que se realizarán, pero, en general, no debe especificar elementos de diseño como los siguientes:

- División del software en módulos.
- Distribución de funciones en los módulos.
- Descripción del flujo de información entre los módulos.
- Elección de las estructuras de datos.

Deben centrarse únicamente en el punto de vista externo del sistema, y no en el funcionamiento interno

Especificación de requisitos del software (SRS)

■ **Restricciones de diseño necesarias**

En algunos casos especiales, los requisitos pueden restringir el diseño de forma severa. Por ejemplo, algunos requisitos de seguridad pueden implicar consideraciones de diseño como:

- Mantener ciertas funciones en módulos separados.
- Permitir o limitar la comunicación entre determinadas áreas del programa.
- Comprobar la integridad de los datos en variables críticas.

Algunos ejemplos de restricciones de diseño válidas los constituyen los requisitos físicos, los de rendimiento y el cumplimiento de estándares en el desarrollo y procesos de garantía de calidad.

■ **Exclusión de parámetros y datos de planificación del proyecto**

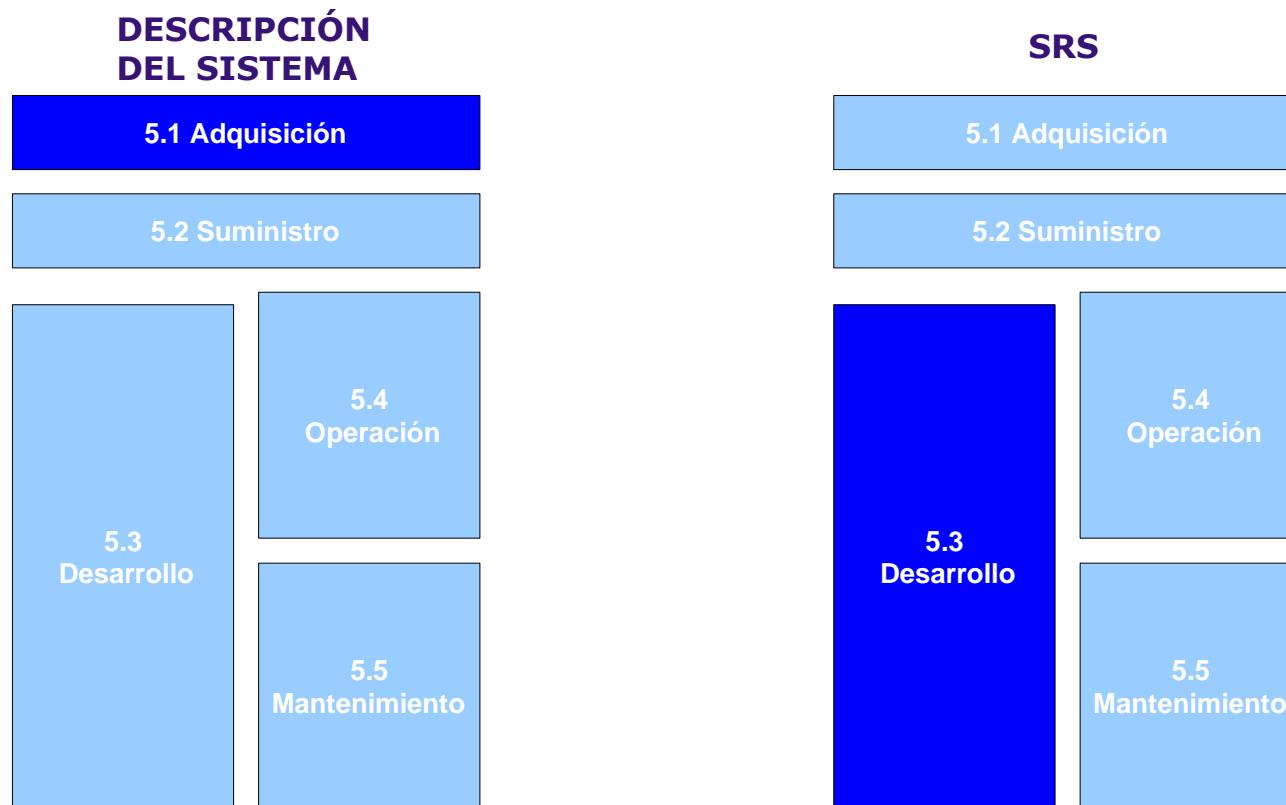
La especificación de requisitos de software se centra en el producto, no en el proceso de producción del producto.

Los requisitos de proyecto representan los términos contractuales entre el cliente y el suministrador, y no deben incluirse en la SRS. Normalmente incluyen información relativa a los procesos de adquisición o de suministro:

- Coste.
- Agenda de entregas.
- Procedimientos de seguimiento.
- Métodos de desarrollo del software.
- Control de calidad.
- Criterios de validación y verificación.
- Procedimientos de aceptación

Diferencias: descripción del sistema – SRS

- Pertenecen a procesos primarios diferentes

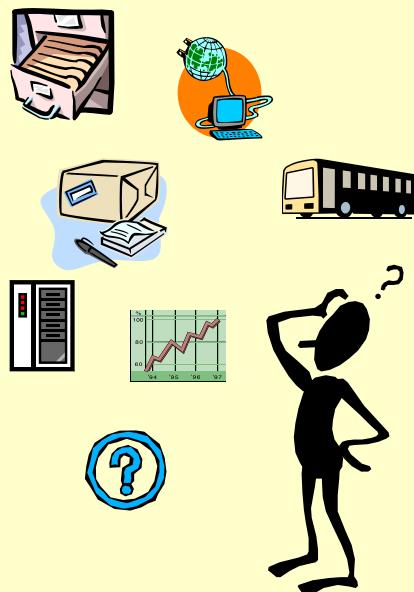


Procesos primarios del ciclo de vida del software (ISO 12207)

Diferencias: descripción del sistema – SRS

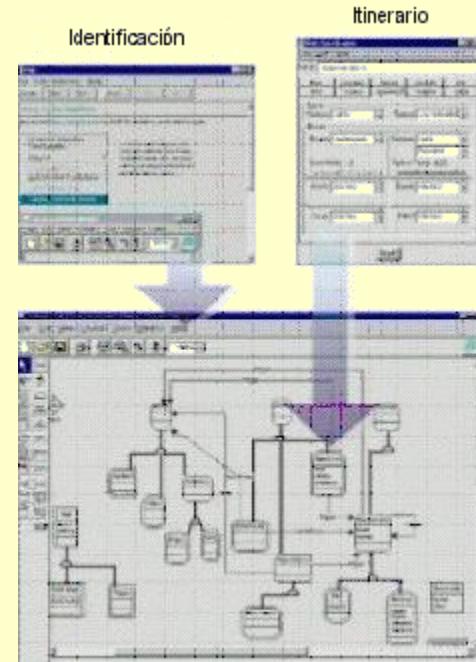
- Pertenecen a entornos diferentes

ENTORNO DEL PROBLEMA



DESCRIPCIÓN DEL SISTEMA

ENTORNO DE LA SOLUCIÓN



SRS

Conceptos clave

Independientemente de las técnicas o procesos que se apliquen para realizar las diferentes tareas relacionadas con el área de requisitos, son cinco los objetivos que hay que cubrir:

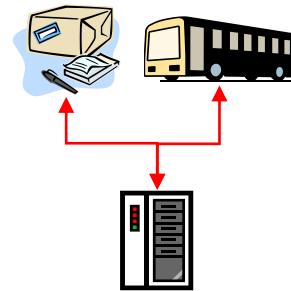
- **ANALIZAR EL PROBLEMA**
- **COMPRENDER LAS NECESIDADES DE LOS USUARIOS**
- **DEFINIR EL SISTEMA**
- **DESARROLLAR LOS REQUISITOS DEL SOFTWARE**
- **GESTIONAR LOS REQUISITOS**



Analizar el problema



Comprender las necesidades de los usuarios



Definir el sistema



Desarrollar los requisitos del software



Gestionar los requisitos

Conceptos clave

■ **Analizar el problema**

Consiste en comprender los problemas reales de los usuarios, y proponer soluciones que cubran sus necesidades.

El objetivo del análisis es conseguir la mayor comprensión posible antes de que empiece el desarrollo.

El analista de requisitos es en realidad un “solucionador de problemas”.

Durante el análisis debe comprender las necesidades de los usuarios, y proponer soluciones. En esta tarea es necesario **explorar y comprender el entorno del cliente**.

Al realizar el análisis hay que

Evitar la tendencia frecuente a los prejuicios creyendo que ya conocemos las necesidades del cliente, y que su principal problema en realidad es que no entiende cuál es su problema.

Tener en cuenta que siempre hay varias maneras de abordar un problema, y que en ocasiones, cambiar la perspectiva del usuario puede generar la solución más eficiente y rentable, aunque no siempre es posible.

Comenzar el análisis sin ideas preconcebidas y teniendo presente el objetivo: conseguir la mayor comprensión posible del problema.

El análisis del problema comprende

- 1.- Identificación del problema.**
- 2.- Identificación de las partes implicadas.**
- 3.- Delimitación de la solución.**
- 4.- Identificación de las restricciones.**



Conceptos clave

■ Comprender las necesidades de los usuarios

La obtención de los requisitos es sin duda la parte más difícil del desarrollo de un sistema, y en la actualidad es la principal causa de problemas.

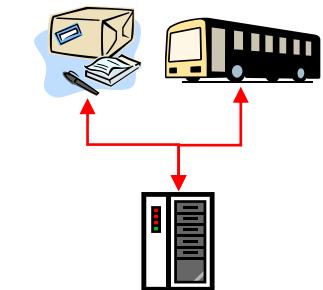
En el apartado "Obtención de requisitos" desarrolla de forma exclusiva este punto.



■ Definir el sistema

La descripción del sistema marca el punto intermedio entre el análisis del problema, y la descripción detallada de los requisitos del software. Es el documento que ofrece una visión general, y ofrece la idea global del sistema en su conjunto. Marca una pausa antes de seguir avanzando hacia los detalles, **para evitar que los árboles nos impidan ver el bosque**.

El resultado de esta fase es el documento de "Definición del sistema", frecuentemente llamado también "ConOps" (Concept of Operations).



Conceptos clave

■ Definir el sistema

La descripción del sistema es el resultado del análisis conceptual, y debe contener toda la información necesaria para describir las necesidades de los usuarios, expectativas, entorno operativo, procesos y características del sistema que se ha ideado para darles solución.

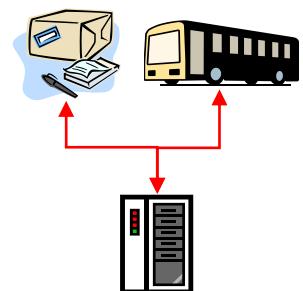
Los elementos esenciales de la descripción del sistema son:

- Descripción del sistema o de la situación actual.
- Descripción de las necesidades que han motivado el desarrollo de un sistema nuevo, o de la necesidad de modificar el actual.
- Modos de operación propuestos para el nuevo sistema.
- Tipos de usuarios y características.
- Funcionalidades propuestas.
- Restricciones que debe respetar el sistema.

Por "Definir el sistema" no consideramos sólo la redacción del "Con Ops" por el ingeniero de requisitos. **También comprende la verificación y validación del documento.**

Por verificación se entiende la supervisión del documento para garantizar que resulta formalmente correcto.

Validación implica la conformidad de las partes afectadas por el sistema (usuarios, clientes, etc.).



Conceptos clave

■ Desarrollar los requisitos del software

Tras analizar los problemas y necesidades de los usuarios, conocer las limitaciones que tener en cuenta, y haber sintetizado en la descripción del sistema la visión global de la solución que se pretende construir, es el momento de profundizar en los detalles.

El nivel de precisión que se debe alcanzar en la descripción de los requisitos del software (SRS), depende de varios factores, incluyendo el contexto de la aplicación, los conocimientos del equipo de desarrollo, así como su experiencia en desarrollos similares.

Los requisitos del software también deben verificarse y validarse, para garantizar, por un lado, que son formalmente correctos, y por otro que dan respuesta a las necesidades de todas las partes implicadas.



Conceptos clave

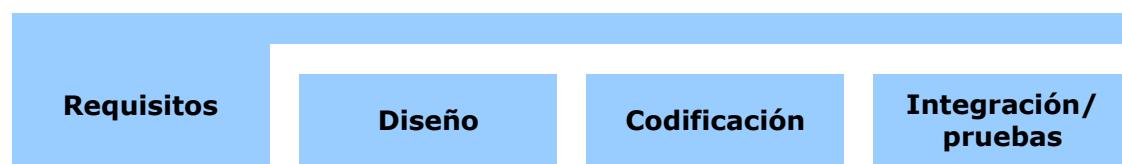
■ Gestionar los requisitos

Una vez que se ha comprendido el problema del usuario, se ha definido y descrito el sistema que se desea construir para solucionarlo, y detallado los requisitos del software, comienza la fase de diseño y desarrollo.

Se puede considerar que la fase de requisitos ya ha terminado al generar los documentos de descripción del sistema y descripción de requisitos del software. Pero lo cierto es que los ciclos de desarrollo secuenciales, o de cascada pura son muy raros, y, aun el caso de que inicialmente se haya planteado este ciclo, desde la gestión del proyecto se debe considerar la posibilidad de incorporar modificaciones en los requisitos durante el periodo de desarrollo.

Cuanto más complejo sea el sistema, y más larga la agenda de desarrollo, habrá mayor probabilidad de modificaciones sobre los requisitos; y si no se gestionan convenientemente deteriorarán, en mayor o menor medida, la planificación y la calidad del proyecto.

Si bien es cierto que no es posible plantear escenarios de desarrollo ideales en los que, tras una definición inicial de los requisitos, éstos se van a mantener inamovibles durante todo el desarrollo del producto; tampoco es posible incorporar modificaciones sobre los requisitos que han servido de base para la planificación del proyecto, y el diseño de la solución, sin que la incorporación obligue a medir las consecuencias que van a tener sobre el trabajo ya realizado, el pendiente de realizar, las posibles reconsideraciones de diseño, y en consecuencia sobre los costes y agendas del proyecto.



La gestión de requisitos da continuidad a esta área durante todo el proyecto



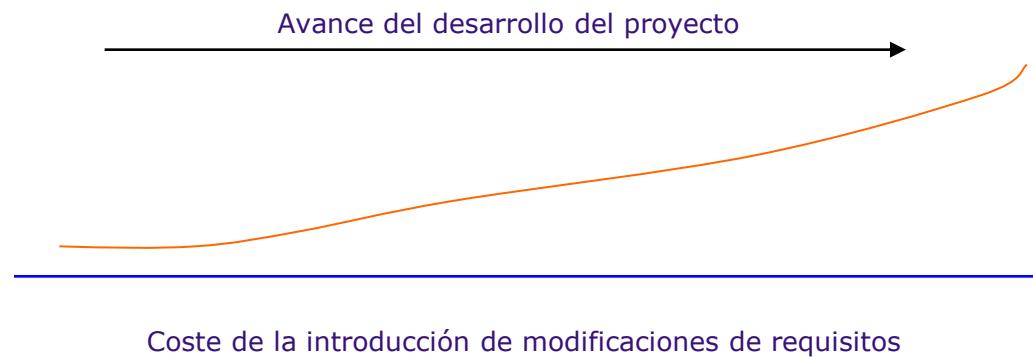
Conceptos clave

■ Gestionar los requisitos

El hecho de tener que gestionar los requisitos durante todo el ciclo de vida del sistema no quiere decir que cualquier momento del desarrollo sea un buen momento para seguir descubriendo cuáles son las necesidades de los clientes. La incorporación de nuevos requisitos, o la modificación de los iniciales resulta mucho más costosa conforme van avanzando las fases del proyecto. Por esta razón, los ciclos secuenciales o de cascada con escasas iteraciones de retroceso son los más eficientes en el consumo de recursos.

Las razones que normalmente no permiten llegar a un conocimiento detallado del problema en la fase inicial de los requisitos suelen ser:

- Sistemas complejos.
- Sistemas para dar soporte a procesos de negocio poco maduros.
- Desarrollos evolutivos impuestos por la necesidad de implantaciones parciales tempranas para los usuarios.



Conceptos clave

■ Gestionar los requisitos

Al analizar el problema del cliente, y desarrollar los documentos de requisitos no hay que escamotear esfuerzos para profundizar en las funcionalidades del sistema, y caer en la tentación de dejar pendientes de concreción, o insuficientemente analizadas partes del problema para más adelante.

La gestión de los requisitos implica que cada modificación de requisitos:

- Debe provenir de una fuente autorizada.
- Debe alcanzar el consenso de las partes implicadas.
- Obliga a un análisis del impacto.
- Implica una revisión de la planificación del proyecto.
- Debe informarse al cliente de los efectos sobre la planificación y recursos necesarios, para obtener su aprobación.
- Debe incorporarse formalmente a la documentación de requisitos



Obtención de los requisitos

■ Síndromes en la obtención de los requisitos

Cuatro son los principales desafíos para el analista de requisitos:

- **Sí... pero no exactamente así.**
- **¡Vaya!, pues esto no debería ser así.**
- **¿Ya está todo?**
- **Usuarios contra desarrolladores**

Obtención de los requisitos

■ **iVaya!, pues esto no debería ser así.**

Este es un problema inherente al desarrollo de software.

Los usuarios no ver el sistema hasta que lo empiezan a usar, y es normal que sea entonces cuando descubran que algunas partes no se adecuan exactamente a sus expectativas.

El software no es físico ni tangible. Al cliente de una vivienda se le puede mostrar una maqueta o un plano. Un proyecto de mobiliario se puede dibujar, pero nuestro producto no es físico, es difícil de representar, de conceptualizar de forma concreta y objetiva.

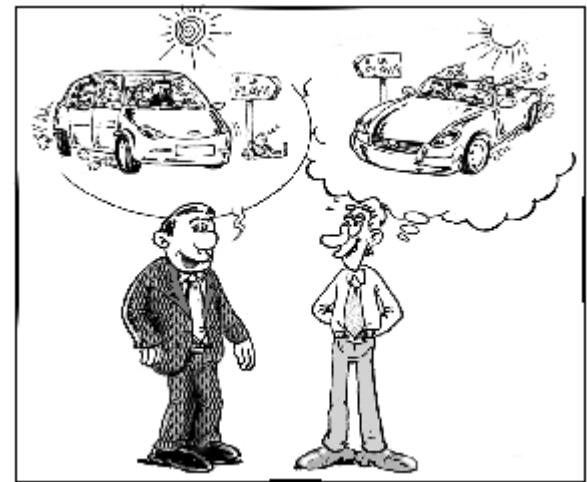
Si el analista de requisitos no comprende bien lo que el cliente necesita, éste se dará cuenta de la disparidad de criterios cuando ya sea tarde, cuando el sistema esté en sus manos; de forma que habremos producido algo que no cumple sus expectativas.

Por esta razón, inherente a la intangibilidad del software, la obtención de requisitos es la fase más importante de un desarrollo.

El ingeniero de requisitos debe tener en cuenta que este síndrome es un riesgo consustancial con su trabajo, y que su misión es anticiparse para que al final del desarrollo produzca el menor efecto posible.

Los medios para reducir su efecto son:

- Evitar quedarse con las primeras descripciones genéricas.
- No dar nada por supuesto.
- Evitar las ambigüedades.
- Conocer el entorno y las necesidades del cliente.
- Dedicar esfuerzo y tiempo para la obtención de requisitos, adecuado al tamaño y complejidad del sistema.



Obtención de los requisitos

■ Sí... pero no exactamente así.

Este síndrome es similar al anterior, porque tiene el mismo resultado: el descubrimiento tardío por parte del cliente de que determinadas partes del sistema no solucionan adecuadamente su problema, pero a diferencia del anterior, su origen no está en omisiones o ambigüedades en el proceso de obtención, sino en la inmadurez de los procesos a los que el nuevo sistema debe dar soporte, o en el desconocimiento o actitud por parte de los interlocutores del cliente.

Aunque tenga el mismo efecto que el síndrome anterior, identificar que tienen causas diferentes interesa en la medida en que requieren soluciones, o formas de trabajo distintas.

El ingeniero de requisitos debe identificar mayores probabilidades de riesgo si en el contexto adquieren relevancia las siguientes situaciones:

- El sistema no sustituye o modifica a otro existente, sino que se desarrolla para dar soporte a procesos de negocio novedosos para la organización que lo solicita.
- Los interlocutores nombrados por el cliente no son conocedores expertos de los procesos cubiertos por el sistema.
- Faltan representantes de partes implicadas por procesos importantes del nuevo sistema.
- Escasa implicación del cliente, que por falta de recursos, tiempo o incluso por pereza intelectual no se sienta con el ingeniero de requisitos a desmenuzar las particularidades de sus procesos, dando por válidos los requisitos finalmente obtenidos, sin prácticamente mirarlos.



Obtención de los requisitos

■ Sí... pero no exactamente así.

Estas situaciones aumentan las probabilidades de terminar un sistema perfectamente validable sobre una descripción de requisitos correcta y completa, sin ambigüedades, pero en el que al final el cliente descubrirá que en, menor o mayor medida, no le soluciona el problema como hubiera sido deseable.

Por supuesto en esta situación, como desarrolladores podremos "argumentar" que tenemos la razón de nuestra parte, puesto que habremos construido lo que el cliente nos pidió, y el problema estriba en que él no sabía bien lo que quería, o se ha dado cuenta de lo que en realidad necesita, cuando ha empezado a trabajar con el nuevo sistema que hemos desarrollado.

De cualquier forma no es una situación ni cómoda ni deseable. Nuestro cliente como experto en su negocio tiene su ego,

y difícilmente reconocerá que no sabía o no quiso explicarnos lo que debíamos construir. Si afortunadamente disponemos de un documento de requisitos formalmente correcto, validado con su firma, tendremos un salvoconducto para hacer efectiva nuestra factura, o defendernos de acciones legales, pero en ningún caso habremos cubierto nuestro objetivo: desarrollar soluciones para los clientes, y habremos creado un sistema que no sirve y un cliente cabreado y descontento.

Este síndrome también es inherente al desarrollo de sistemas de software, y con él resulta fácil deducir las funciones y competencias que debe cubrir el ingeniero de requisitos, así como de ser persona con "ojo clínico" y registro amplio de recursos.

Si se enfrenta a procesos poco maduros deberá involucrarse en mayor medida en el entorno organizacional del cliente y aportar en su trabajo parte más propia de consultoría que de analista de requisitos.



Obtención de los requisitos

■ Sí... pero no exactamente así.

Deberá también aportar asesoría profesional al cliente informándole del riesgo alto que encierra el proyecto de producir versiones que se demostrarán inadecuadas para la realidad de sus procesos, y de la conveniencia de profundizar el máximo posible en el conocimiento de los procesos antes de elaborar los requisitos, así como de emplear técnicas de prototipado en la obtención de requisitos, y ciclos de desarrollo en cascada. Resultan más aconsejables desarrollos incrementales o evolutivos, con ciclos en espiral y seguimiento por parte del cliente.

Si el analista se encuentra con problemas de comunicación o de actitud por parte del cliente deberá conducir la situación y adaptar su registro de actuación de forma que sin perder la assertividad, logre establecer una implicación adecuada del cliente y un flujo de comunicación productivo.



Obtención de los requisitos

■ ¿Ya está todo?

Cuándo se puede dar por terminado un trabajo?

Cuando ya no queda más por hacer.

¿Cómo sabe el ingeniero de requisitos que ha descubierto todos los requisitos necesarios?.

Esta incertidumbre es también inherente al trabajo del ingeniero de requisitos, porque nunca tendrá la certeza de haber descubierto todas las necesidades y restricciones, y sobre todo porque siempre puede dar por descontado que algo se queda sin descubrir.



La única forma de afrontar esta circunstancia es dedicar tiempo suficiente a la obtención y análisis, e identificar a todos los participantes o partes implicadas en el proyecto.

Aunque nunca podrá afirmar haber localizado todos los requisitos, el objetivo en este caso es alcanzar el convencimiento de haber descubierto lo suficiente, y que las posibles omisiones pertenecerán a cuestiones menores, que pueden surgir durante la gestión de los requisitos, o a lo largo del mantenimiento del futuro sistema.

Obtención de los requisitos

■ Usuarios contra desarrolladores

No es posible saber qué necesita el cliente, si no disponemos de comunicación fluida con los interlocutores de su organización; y por desgracia es demasiado frecuente que los desarrolladores y los usuarios, se relacionen sobre la base de la desconfianza mutua, y empleen idiomas distintos.

Tanto la actitud de los desarrolladores como la de los usuarios no suele ser favorable para trabajar unos con otros. Los primeros prefieren concentrar su trabajo en el entorno técnico, y olvidarse de "hablar con los clientes".

Los usuarios, por su parte, esperan su nuevo programa, con la misma actitud que podrían esperar un coche tras haberlo encargado en el concesionario.

Los analistas y los usuarios pertenecen a dos comunidades que desconfían mutuamente.

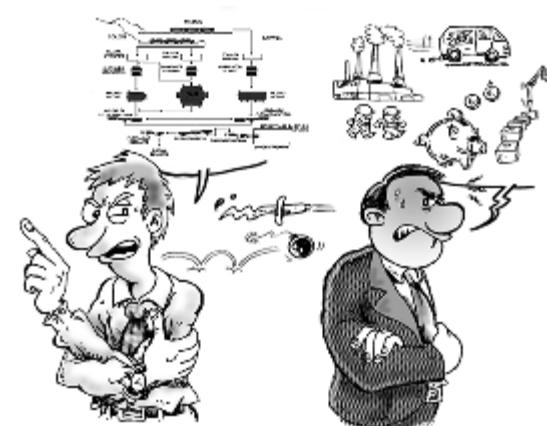
Los usuarios ven a los desarrolladores como personas incapaces de conseguir sistemas que funcionen correctamente sin la necesidad de estar constantemente "parcheándolos".

Los desarrolladores se ven solos y desamparados como únicos responsables de todo cuando ocurra o tenga relación con el sistema.

Por supuesto, nosotros no esperamos que los usuarios cambien, pero tenemos que conocer estos problemas, y el ingeniero de requisitos debe estar preparado para encontrarse con estas dificultades y minimizar sus consecuencias.

Se supone que durante la obtención de los requisitos, tanto los usuarios como los desarrolladores comparten el mismo objetivo: definir cómo ha de ser el nuevo sistema, pero lo cierto es que cada uno tiene objetivos diferentes.

Por nuestra parte estamos interesados en desarrollar una buena descripción de requisitos, completa y correcta. Queremos especificar un sistema técnicamente viable, que integre la funcionalidad necesaria de forma eficiente sobre un diseño limpio y robusto.



Obtención de los requisitos

■ **Usuarios contra desarrolladores**

Por su parte los usuarios (cuando se implican) centran su interés en definir el sistema con que esperan trabajar, sin querer saber nada de agendas, viabilidad, prioridades, etc.

Para abordar con las mayores garantías de éxito este problema, por nuestra parte:

Debemos sumergirnos en la organización del cliente; estudiar, analizar y comprender los procesos y problemas a los que tiene que dar cobertura el nuevo sistema.

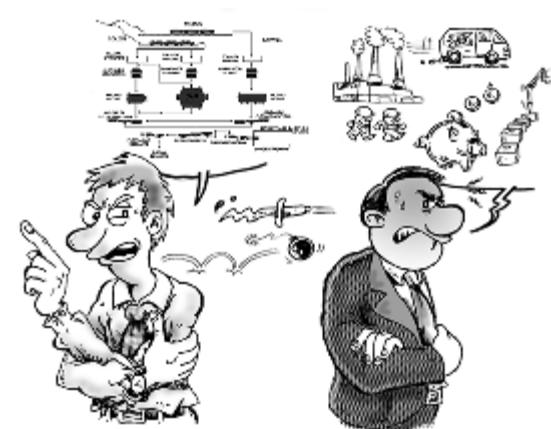
En las comunicaciones de requisitos, así como en la descripción del sistema, tenemos que emplear un lenguaje natural, sin tecnicismos; y adoptar la terminología habitual del entorno del cliente.

Mantener un enfoque y unidad de criterio común por todas las personas de nuestra organización, de cara al cliente.

Por parte del cliente:

Debe facilitar interlocutores conocedores de los procesos y problemas que debemos conocer, con tiempo y motivación suficiente para trabajar con nosotros.

Los interlocutores deben ser concretos y específicos en sus descripciones, revisar y validar los documentos de requisitos generados.



Obtención de los requisitos

■ Problemas frecuentes en la obtención de requisitos

Los problemas más frecuentes pertenecen a 3 categorías:

- **Delimitación confusa del ámbito del sistema.**
- **Comprensión**
- **Inestabilidad**

Problema: delimitación confusa del ámbito del sistema

Antes de entrar en la obtención de requisitos con detalle es necesario conocer cuáles son los objetivos y los límites del sistema.

Si no controlamos los límites y objetivos esperados del sistema, el sistema nos controlará a nosotros

Los contextos que es necesario conocer para centrar apropiadamente el sistema en su entorno son:

- **Organización**
- **Entorno**
- **Proyecto**

Obtención de los requisitos

Problema: delimitación confusa del ámbito del sistema

Para evitarlo deben analizarse y conocerse los tres ámbitos señalados

ORGANIZACIÓN

Para llevar a cabo la obtención de requisitos es preciso conocer y comprender la organización en la que trabajará el sistema, y los objetivos que se pretenden conseguir.

ENTORNO

Los factores del entorno del sistema influyen de forma determinante en el proceso de obtención de requisitos. Los más importantes son:

- Restricciones: de hardware, sobre el software o sobre los procesos de desarrollo.
- Madurez de los procesos del entorno de operación.
- Grado de certidumbre de los interfaces con otros sistemas.

PROYECTO

El contexto en el que se desarrolla el proyecto también afecta a los procesos de obtención de requisitos, que deberá adecuar los métodos y técnicas de obtención a las características del proyecto:

- Características específicas de cada grupo de agentes implicados en el proyecto (usuarios, cliente, desarrolladores, normativas, etc.)
- Restricciones impuestas por las partes implicadas en la obtención de los requisitos (agenda, coste, parámetros de calidad deseados, etc.)

Obtención de los requisitos

Problema: comprensión

El 56% de los errores deslizados en los sistemas desarrollados se deben a deficiencias en la comunicación “usuario – analista” durante la obtención de los requisitos, y este tipo de errores son los más caros de corregir porque llegan a consumir hasta el 82% del tiempo de desarrollo^[1].

Los problemas de comprensión producen requisitos incompletos, con ambigüedades, inconsistentes; y en definitiva incorrectos, porque no definen las necesidades reales de los usuarios.

Estos problemas se pueden agrupar en tres categorías:

- Dar por supuesto lo desconocido.
- Lenguaje.
- Información desestructurada.

Problema: inestabilidad

Los requisitos son inestables y cambian durante el desarrollo y tras la entrada en servicio del sistema.

La solución para evitar problemas radica en el proceso de gestión de requisitos.

[1] Goodrich, Victoria, and Olfman, Lorne. An experimental Evaluation of Task and Methodology Variables for Requirements Definition Phase Success. In Bruce D. Shriner (editor), Proceedings of the twenty-third Annual Hawaii International Conference on System Sciences, p. 201-209. IEEE Computer Society, January 1990

Obtención de los requisitos

■ Técnicas de obtención de requisitos

TÉCNICAS

ENTREVISTAS

Reuniones JAD, cuestionarios
reuniones de grupo
entrevista, lluvia de ideas

ESCENARIOS

Casos de uso, tarjetas CRC
diagramas de flujo, escenarios

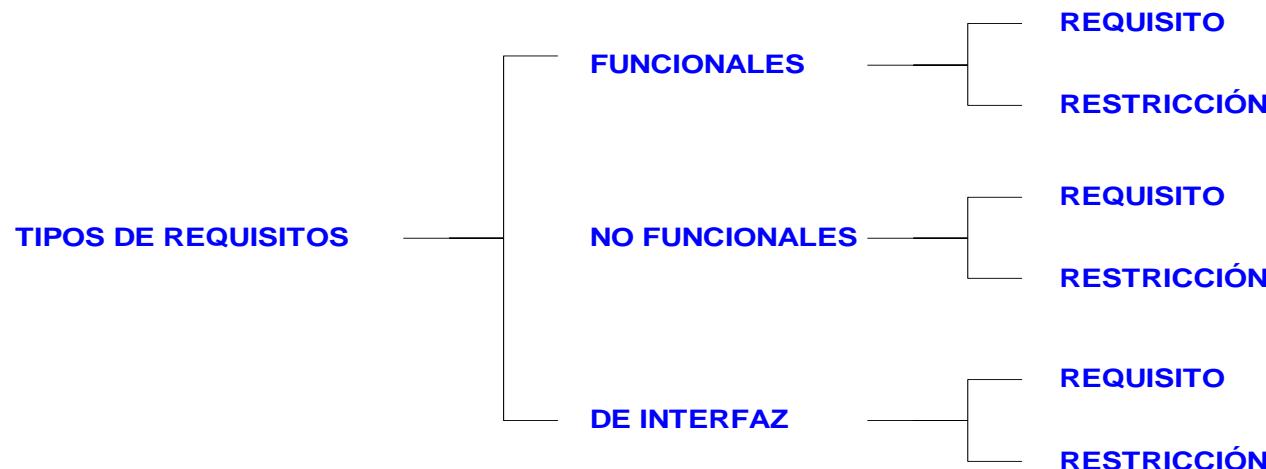
PROTOTIPOS

Prototipos rápidos
prototipos evolutivos

OBSERVACIÓN

Introspección
análisis de protocolo
documentación, otros sistemas

Clasificación de los requisitos



■ Requisitos funcionales

Definen el comportamiento del sistema.

Describen las tareas que el sistema debe realizar.

Al definir un requisito funcional es importante mantener el equilibrio entre la excesiva generalidad, insuficiencia de detalle o ambigüedad, y el exceso de detalle con precisiones o descripciones innecesarias o redundantes.

Clasificación de los requisitos

■ Requisitos no funcionales

Definen aspectos, que sin ser funcionalidades, (tareas que el sistema debe realizar) resultan deseables desde el punto de vista del usuario. Generalmente comprenden atributos de calidad:

- Tiempos de respuesta.
- Características de usabilidad.
- Facilidad de mantenimiento.
- etc.

■ Requisitos de interfaz

Definen las interacciones del sistema con su entorno:

- Usuarios
- Otros sistemas

Clasificación de los requisitos

■ **Restricciones**

Los requisitos, en su definición purista definen el QUÉ, y no el CÓMO; pero en el conjunto de necesidades que debe cubrir un sistema, no sólo hay que tener en cuenta QUÉ cosas hay que hacer, sino también en ocasiones CÓMO deben hacerse.

La clasificación entre requisitos puros (QUÉ) y restricciones (CÓMO) la debe considerar el analista para que el equipo de trabajo sepa hasta qué punto determinados aspectos limitan sus opciones de trabajo, y poder mantener así la trazabilidad con su origen (necesidad apuntada por el usuario, normativa legal, limitación técnica, etc.)

Con carácter general las restricciones imponen limitaciones:

- En la libertad de los analistas al realizar el diseño del sistema.
- En los procesos o formas de trabajar que se emplearán en el desarrollo del sistema.

El analista del sistema elige entre todas las opciones tecnológicamente posibles aquellas que según su criterio profesional y las circunstancias del sistema, aportan mejor solución para la implementación de los requisitos funcionales y no funcionales.

La indicación por parte del cliente de instrucciones como:

Debe emplearse base de datos Oracle.

Los procesos de desarrollo deben ser conformes a Métrica 3.

El sistema final debe ejecutarse sobre la plataforma libre Linux.

Debe desarrollarse empleando Java.

El interfaz de comunicación con un programa externo de contabilidad debe hacerse de la siguiente forma...

Clasificación de los requisitos

■ Problemas de clasificación y nivel de rigor necesario

Para nosotros la base teórica de clasificación es un marco de referencia con la definición de los criterios de clasificación.

En la relación de requisitos de un sistema, no resulta interesante entrar en análisis puristas para determinar si cada requisito lo es de interfaz, funcional, etc.

La diferencia entre:

"El sistema comprueba la autenticación y autorización del usuario y le da acceso a una pantalla con el menú general o en caso de error le redirige a la pantalla de usuario y contraseña otra vez"

Y:

"RS. 3 El sistema sólo permite el acceso al menú principal a usuarios autorizados.

RT.3.1 El sistema identifica al usuario solicitando a través de la pantalla de operación su nombre y contraseña de acceso."

En el segundo caso, el equipo de trabajo sabe que debe descartar opciones de identificación a través de tarjetas, o dispositivos biométricos, o cualquier otra opción posible.

Se trata por tanto de conocer y comprender el concepto de restricción, para aplicarlas sólo cuando son necesarias, dejando así el mayor margen posible de libertad para el diseño de la solución de software.

Calidad de la documentación

■ Características de las buenas descripciones de requisitos

Requisitos	Especificación
Posibles	Completa
Necesarios	Correcta
Priorizados	Consistente
Concretos	Modificable
Verificables	Trazable

Propiedades de los buenos requisitos

Posibles

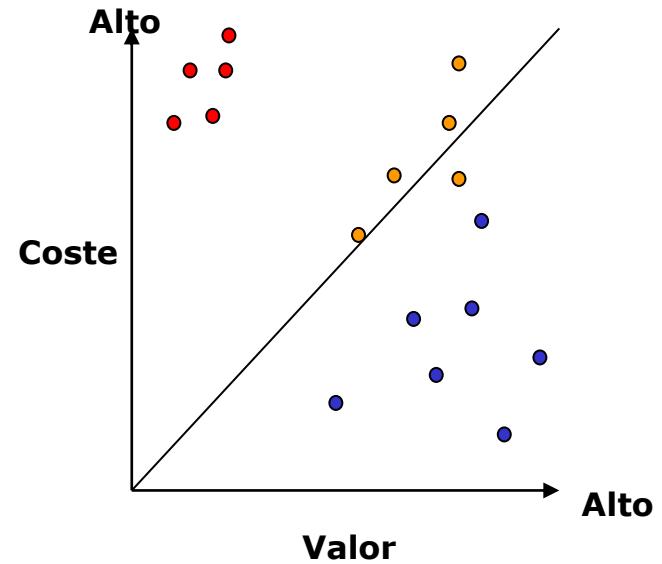
Cada requisito debe **poder implementarse** dentro de las capacidades y limitaciones conocidas del sistema y su entorno. El director técnico deberá comprobar la viabilidad de los requisitos antes de comprobar el documento.

Necesarios

Un requisito es necesario si es algo:

- que el cliente realmente necesita
- requerido para la conformidad con un requisito
- requerido para la conformidad con un interfaz, externo o estándar.

Para evitar requisitos innecesarios, el cliente debe valorar cada funcionalidad y como afectará al sistema si esta o no.



Propiedades de los buenos requisitos

Requisitos priorizados

Los requisitos de una SRS deben incluir una indicación de la importancia del requisito en el conjunto del sistema.

Normalmente todos los requisitos de un producto de software no son igual de importantes. Algunos resultan esenciales, y otros son deseables.

Cada requisito debe identificar estas diferencias de forma clara, de esta forma ayuda a:

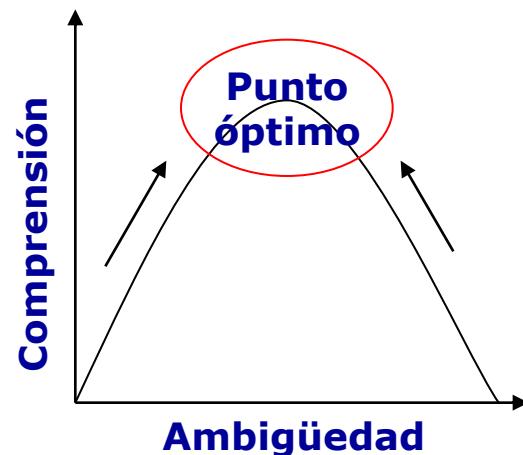
- Los clientes tengan una consideración más adecuada de cada requisito, y a menudo clarifica asunciones que pudieran estar ocultas.
- Que los desarrolladores tomen decisiones de diseño correctas y dediquen niveles de esfuerzo apropiado a las diferentes partes del producto.
- Que el gestor del proyecto pueda establecer prioridades de ejecución, y disponga de información adicional en caso de problemas de agenda.

Propiedades de los buenos requisitos

Concretos

Un requisito es concreto si tiene una única interpretación. Como mínimo esto requiere que cada característica del producto final se describa empleando un término único. En los casos en los que el término puede tener diferentes significados según el contexto, éste debe incluirse en el glosario de la SRS con el significado con el que se emplea.

Punto óptimo: Mayor grado de comprensión con la menor ambigüedad



Modos eficaces de evitar la ambigüedad:

- Inspecciones formales de los documentos de requisitos.
- Escritura de casos de prueba
- Elaboración de casos de uso.
- Elaboración de diagramas.

Clasificación de los requisitos

Verificable

Un requisito es verificable si, y sólo si a través de un proceso concreto y finito es posible comprobar si el software lo cumple. En general los requisitos ambiguos no son verificables.

Los requisitos no verificables incluyen sentencias como “que trabaje eficientemente”, “interfaz de usuario amigable”, “debe responder rápidamente”. Estos requisitos no son verificables porque no es posible definir los términos “eficiente”, “amigable”, “rápido”. La sentencia “el programa no debe entrar nunca en un bucle infinito” tampoco es verificable porque un nivel de pruebas absoluto es teóricamente imposible.

Un ejemplo de requisito verificable es:

“El tiempo de respuesta para la compra de un billete sencillo no debe superar los 2 segundos el 90% de las veces, y una transacción de compra de un billete sencillo nunca debe tardar más de 5 segundos.”

Esta sentencia es verificable porque emplea términos concretos y magnitudes medibles y comprobables.

Si no es posible establecer un método para comprobar si el software cumple con un determinado requisito, el requisito debe eliminarse o revisarse

Propiedades de la documentación

Completa

Una SRS es completa si, y sólo si incluye los elementos siguientes:

- Todos los requisitos significativos, relativos a funcionalidad, rendimientos, restricciones de diseño, atributos e interfaces externos.
- Definición de las respuestas del software a todas las posibles entradas de datos en toda clase de situaciones. Es importante especificar las respuesta tanto para datos de entrada válidos, como inválidos.
- Referencias a todas las imágenes, tablas y diagramas y definición de todos los términos propios y unidades de medida no normalizadas.

No puede considerarse completa una SRS si en la descripción de algunos requisitos se incluye la frase "A determinar" o la expresión inglesa "TBD" (to be determined).

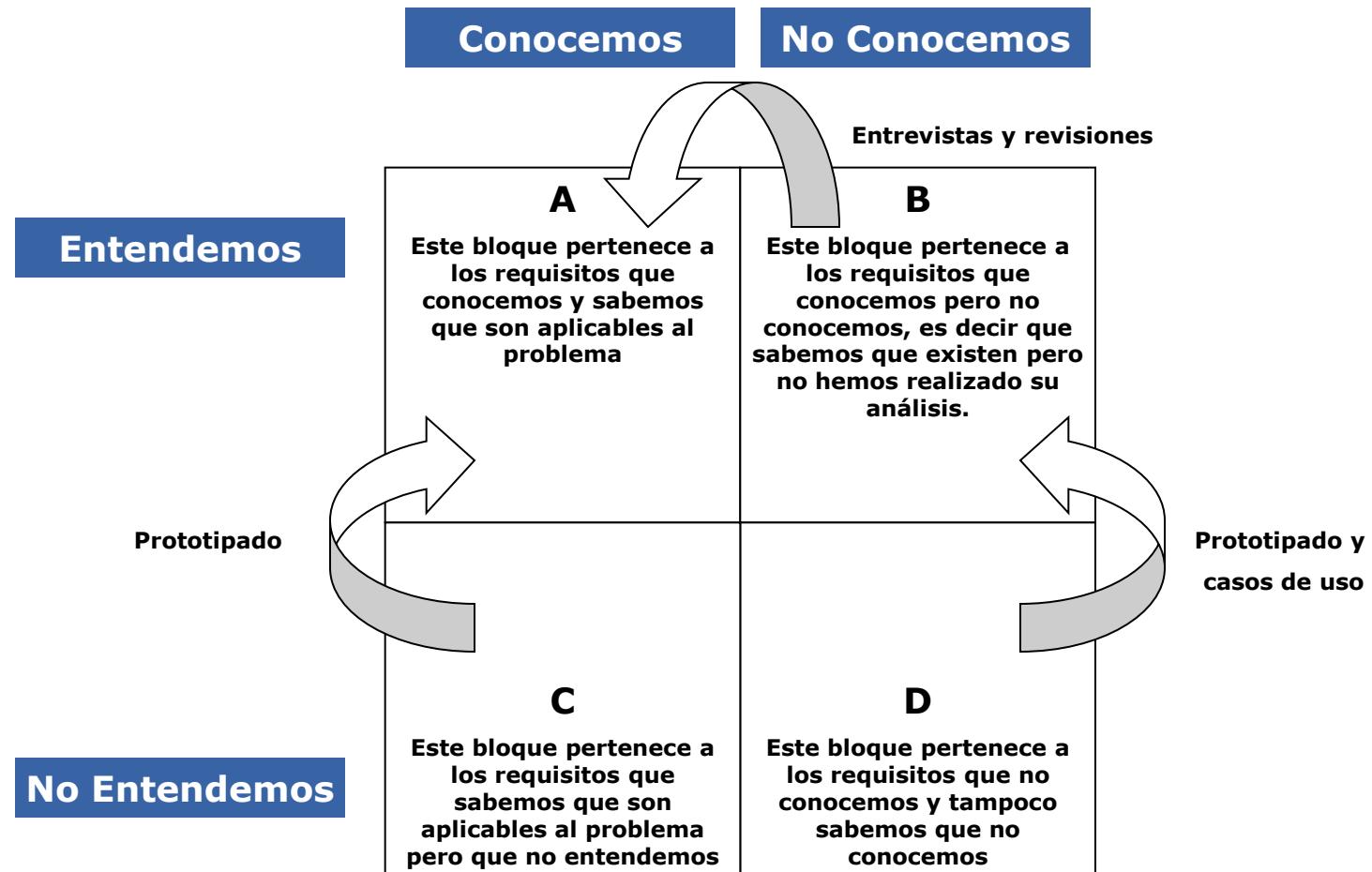
Si excepcionalmente se indica que un requisito se concretará más adelante es necesario indicar también:

Descripción de las causas por las que no se ha concretado el requisito.

Descripción de qué debe realizarse para poder eliminar el "TBD", quién es la persona responsable de llevarlo a cabo, y cuándo debe eliminarse

Clasificación de los requisitos

Completos



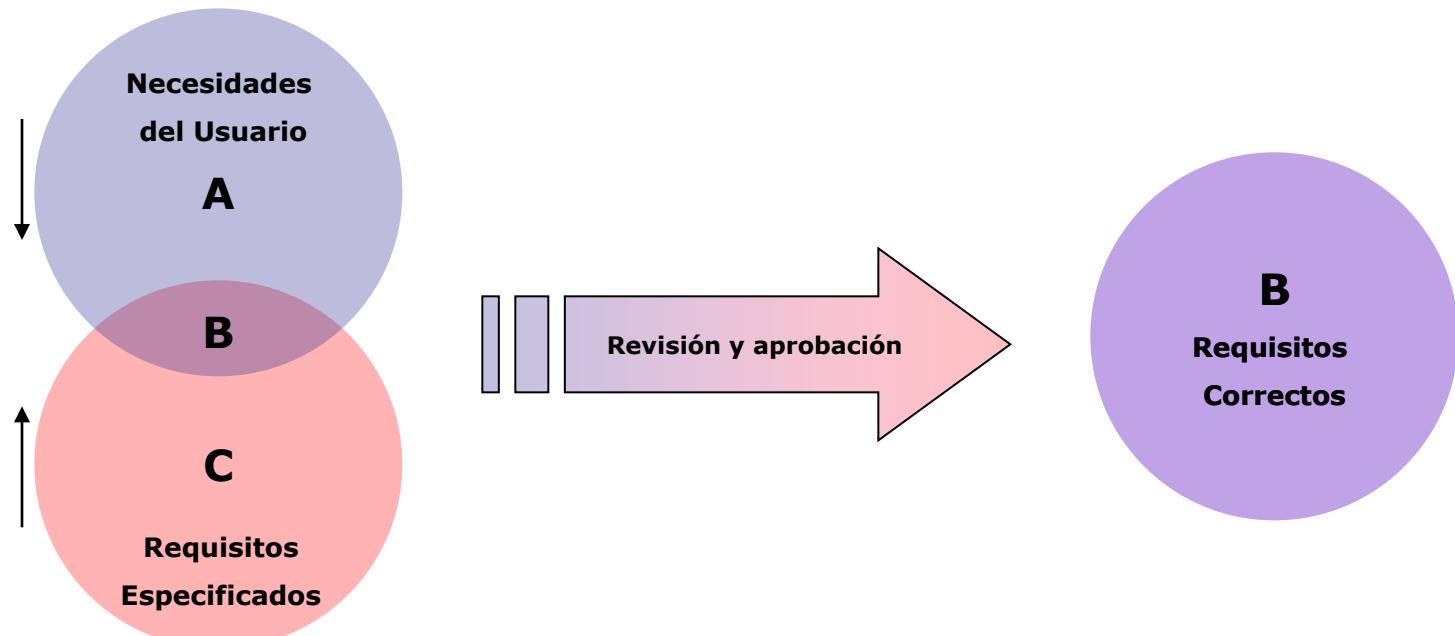
Clasificación de los requisitos

Correcta

Una especificación de requisitos de software es correcta si, y solo si todos y cada uno de los requisitos indicados son los que debe cubrir el software del sistema.

No hay ninguna herramienta que pueda garantizar la corrección. Una SRS debe compararse con las especificaciones de rango superior del proyecto (Descripción del sistema, documentación referenciada, etc.) para comprobar que cumple sus indicaciones.

También es recomendable que la parte cliente determine si la especificación de requisitos de software refleja sus necesidades actuales

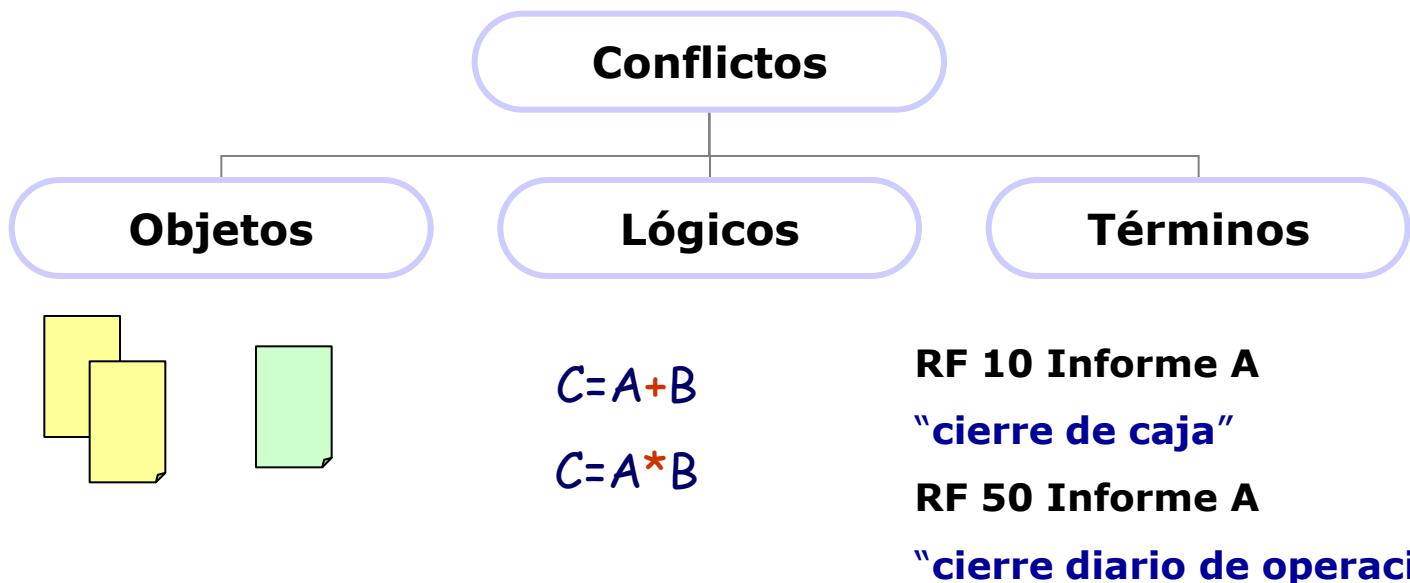


Clasificación de los requisitos

Consistente

El atributo de consistencia se refiere a consistencia interna no a conformidad o congruencia con documentos superiores (ej. descripción del sistema). La ausencia de esta congruencia supondría un problema de corrección y no de consistencia.

Una documentación es internamente consistente si, y solo si, no se establecen conflictos entre requisitos individuales o grupos de requisitos. Los tres tipos de conflictos posibles son:



Clasificación de los requisitos

Modificable

Un documento de requisitos es modificable si, y sólo si su estilo y estructura permiten que puedan llevarse a cabo modificaciones en los requisitos manteniendo la estructura y el estilo, de forma fácil, completa y consistente. La modificabilidad generalmente requiere en la documentación:

Que tenga una organización coherente y fácil, con una tabla de contenidos y un índice..

Que no sea redundante. (p. ej. que el mismo requisito no aparezca en dos lugares del documento)

Exprese cada requisito por separado, mejor que mezclados con otros requisitos.

La redundancia, por sí misma no es un error, pero puede acarrearlos. En ocasiones la redundancia puede hacer un SRS más legible, pero puede generar errores al actualizar el documento, y generar inconsistencias si sólo se actualiza una de las apariciones, olvidando la otra.

Clasificación de los requisitos

Trazable

Un SRS es trazable si establece de forma clara el origen de cada requisito, y facilita su referencia en las futuras etapas del desarrollo, o en las actualizaciones de la documentación. Se recomiendan los dos tipos siguientes de trazabilidad:

Trazabilidad remota (hacia fases previas del desarrollo). Para ello se debe referenciar la fuente del requisito.

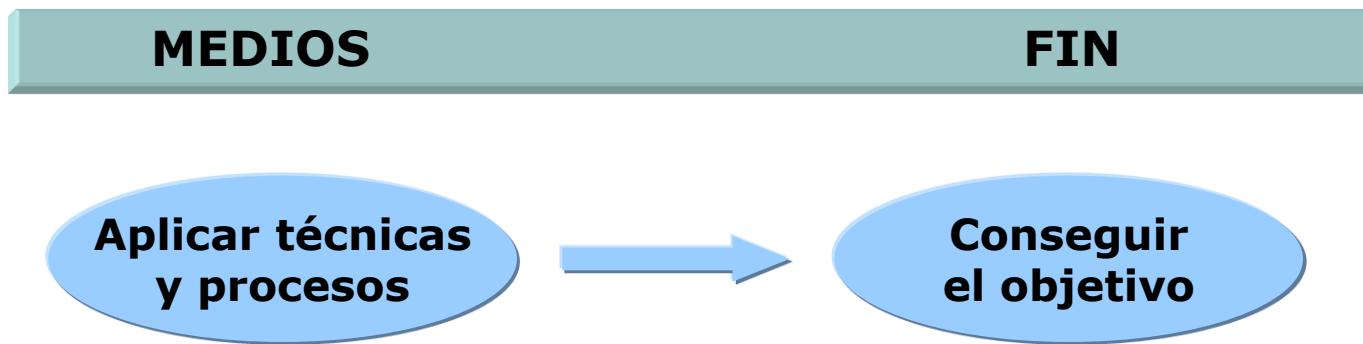
Trazabilidad futura (hacia fases posteriores del desarrollo). Para ello cada requisito debe tener un nombre o referencia única.

La trazabilidad remota es importante cuando el producto de software entra en la fase de operación y mantenimiento. Al modificar el diseño y el código es esencial poder determinar todos los requisitos que quedan afectados por una modificación

Conclusiones



Conclusiones



Diseño de software

1.0



Diseño

■ Definición

El proceso de definición de la arquitectura, componente, interfaces y otras características de un sistema o de un componente.

El resultado de este proceso.

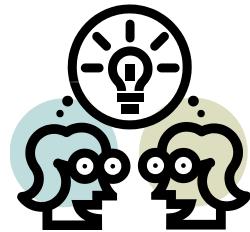
IEEE std. 610.12-1990 Glossary of software engineering terminology

El diseño del software comprende la descripción de la arquitectura del sistema con el nivel de detalle suficiente para guiar su construcción.

- Descomposición del sistema
- Organización entre los componentes del sistema
- Interfaces entre los componentes

Conclusiones

Diseño es la actividad del ciclo de vida en la que se analizan los requisitos del software para desarrollar una descripción de la estructura interna y la organización del sistema que servirá de base para su construcción.



Requisitos



Diseño



Construcción

Conclusiones

■ El diseño como creación de modelos

Una vez conocidas las necesidades de los usuarios es preciso diseñar una solución.

Empleando el símil con la construcción de edificios, tras conocer cuales son las necesidades que se desean cubrir con un edificio (hotel, colegio, vivienda familiar, edificio de apartamentos...), es el momento de diseñar la solución.

Las posibilidades son muchas, y exceptuando proyectos de tamaño mínimo, la complejidad de concebir todas las facetas e interacciones del sistema desborda la capacidad de abstracción mental para concebirlo en una única visión. Al mismo tiempo es necesario que todas las personas implicadas en el proyecto conozcan y compartan los "planos" de la solución.

Así pues, las razones del diseño son:

- Concepción u análisis de las posibles soluciones.
- Apoyo metodológico para abordar la complejidad de la solución.
- Registro documentado como medio de comunicación entre los participantes.

Un modelo es una representación simplificada de la realidad.

De igual forma que al concebir un edificio se divide la complejidad del sistema para hacerlo digerible, y se generan diversos modelos de los diferentes aspectos: planos de estructura, planos del subsistema de fontanería, del de electricidad, etc. los sistemas de software son también realidades complejas que es preciso conocer (modelizar) para llevar a cabo el diseño de su solución.

Actividades del diseño del software

El diseño del software comprende dos actividades intermedias entre la fase de requisitos y la de construcción:

■ Diseño de la arquitectura del software

Descripción de la arquitectura general, identificación de sus componentes y su organización y relaciones en el sistema.



■ Diseño detallado del software

Definición y estructura de los componentes y datos.

Definición de los interfaces

Elaboración de las estimaciones de tiempo y tamaño.



Considerando que la descripción del sistema (ConOps) dibuja una primera aproximación del sistema en su conjunto, algunos autores diferencian entre:

- Diseño del sistema (la visión del documento de descripción del sistema).
- Diseño de la arquitectura
- Diseño del detallado del software

Razones del diseño del software

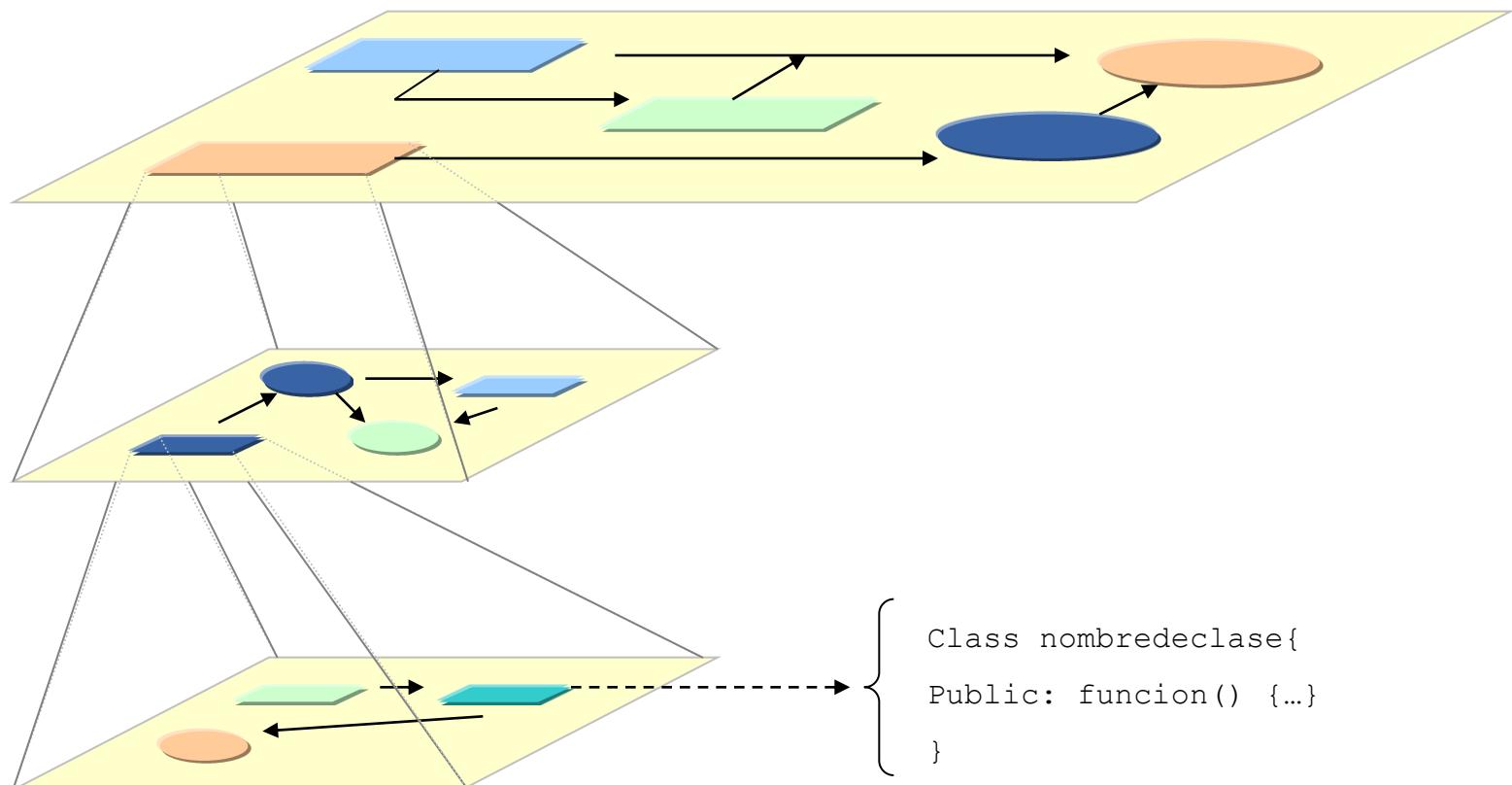
■ ¿Por qué?

El resumen de las razones expuestas que hacen necesarias las tareas de diseño antes de comenzar la construcción de un sistema son:

- Permite la descomposición del problema en partes y vistas de menor tamaño, más manejables para el trabajo intelectual del diseño de la solución.
- Permite el desarrollo de modelos que se pueden analizar para determinar si cumplen los distintos requisitos.
- Permite examinar soluciones alternativas.
- Los modelos se pueden utilizar para planificar el desarrollo de las actividades, y son el punto de partida para empezar las actividades de codificación y pruebas.

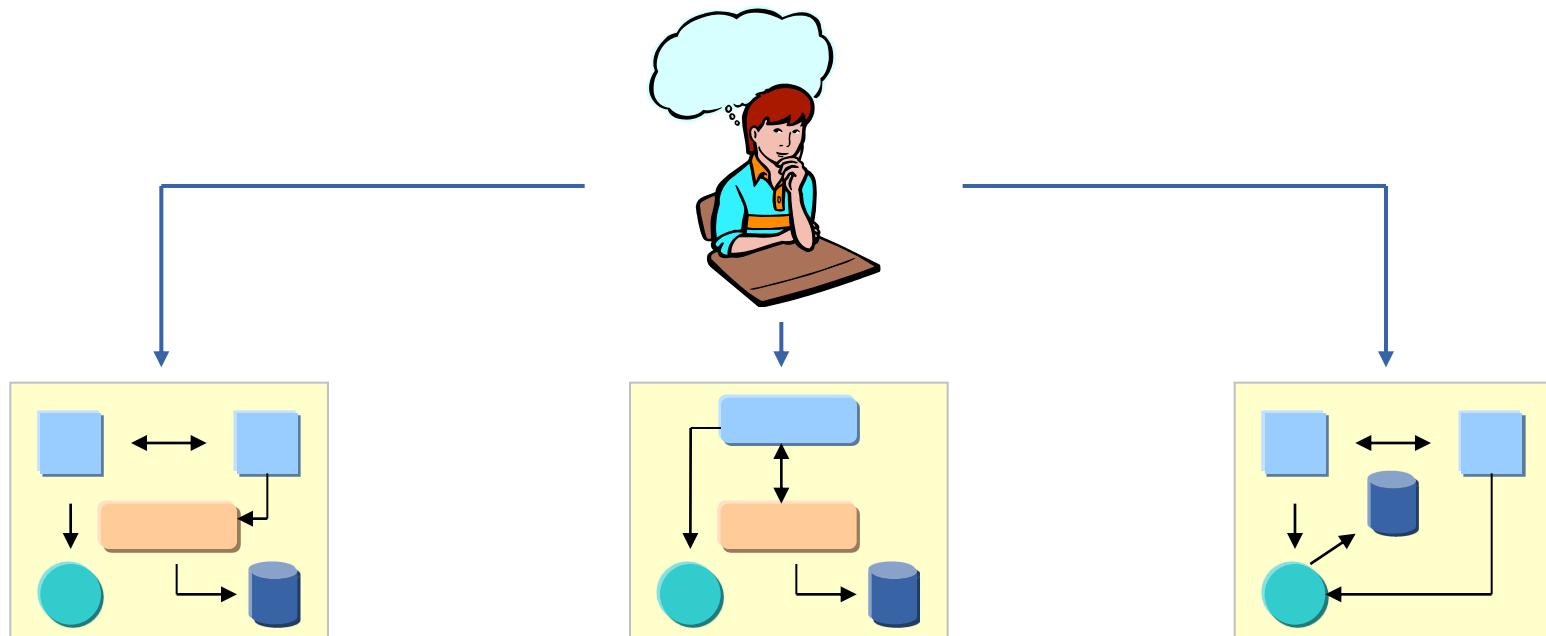
Conclusiones

■ Descomposición de la complejidad



Conclusiones

- Análisis de soluciones posibles a través de su modelado.



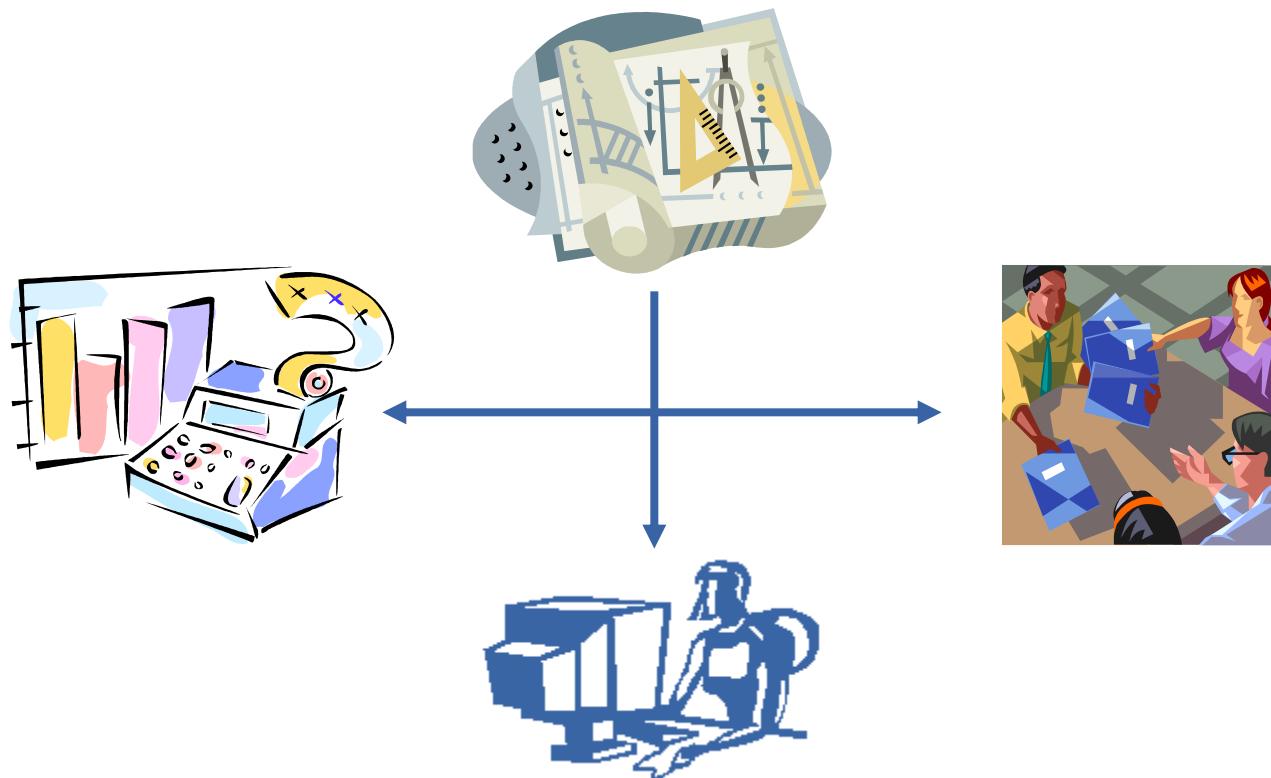
Requisitos



- Disponibilidad
- Coste desarrollo
- Coste mantenimiento
- Robustez
- Tiempos de respuesta
- Hardware necesario
- Etc.

Conclusiones

- Elemento de comunicación, Base de planificación y del desarrollo



Fin del proceso de diseño

■ Se considera que el proceso de diseño se ha completado cuando

- Todas las preguntas “Como” tienen respuesta
- La descripción del diseño de la arquitectura está completada
- La revisión del diseño se ha completado y cada equipo/persona implicado está de acuerdo con el diseño.
- Los borradores de manuales para mantenimiento y administración están realizados
- Se ha realizado la trazabilidad del diseño
- Se ha revisado el diseño de la arquitectura
- Se ha verificado el diseño de la arquitectura
- Se ha escrito la planificación de la integración del software.
- Se ha establecido la línea base del producto



Vistas del diseño de la arquitectura

Un sistema de software es una entidad ortogonal que puede contemplarse o analizarse desde diferentes “vistas”:

Puede enfocarse la atención en:

- Distribución física del software entre los diferentes elementos del sistema.
- Descomposición en las diferentes funcionalidades que realiza.
- Estructuras de la información que gestiona.
- Etc.

De esta forma el diseño puede generar modelos para cada una de las diferentes vistas empleadas en su análisis (modelo físico, modelo de datos, modelo se procesos, etc.).

Notación empleada

Si bien el concepto y la finalidad del diseño o modelado de un sistema de software es siempre el mismo, las notaciones pueden variar en función de las características de cada proyecto o de los conocimientos o preferencias de las personas u organización que lo realice.

A través del lenguaje de modelado empleado (UML, IDEF, Diagramas de flujo, etc.) se consiguen realizar dos tipo de descripciones:

■ Descripciones estructurales

Las notaciones para descripciones estructurales suelen ser gráficas y representan los diferentes componentes y sus relaciones.

- Lenguajes de descripción de arquitecturas (ADL): AADL, AESOP, CODE, MetaH, Gestalt, Modechart, UML, Unicon, Modechart, etc.
- Diagramas de clases y objetos
- Diagramas de componentes
- Diagramas entidad-relación
- Lenguajes de descripción de interfaz
- Etc.

■ Descripciones de comportamiento

- Diagramas de actividad
- Diagramas de colaboración
- Diagramas de flujo de datos
- Diagramas de flujo
- Pseudo-código y lenguajes de diseño (PDL)
- Diagramas de secuencia
- Etc.

Estrategias y métodos de diseño

Las principales estrategias que suelen emplearse para el diseño del software son:

- Orientadas a funciones (estructurada)
- Orientada a objetos (diseño orientado a objetos)
- Diseño centrado en las estructuras de datos (menos empleado)

■ **Diseño estructurado**

Esta es la aproximación clásica y se centra en la identificación y descomposición de las principales funciones del sistema hacia niveles más detallados.

■ **Diseño orientado a objetos**

Es la aproximación más popular actualmente, sobre la que se han desarrollado numerosos métodos partiendo de su concepción inicial en la década de los 80

A través del diseño orientado a objetos (OOD), se desarrollan las especificaciones de sistemas como modelos de objetos (sistemas compuestos por conjuntos de objetos que interactúan entre ellos). que, expuesta de forma muy básica, identifica a los nombres como objetos, a los verbos como los comportamientos que pueden ofrecer y a los adjetivos como sus métodos.

Para cada estrategia hay numerosos métodos (notaciones, lenguajes de modelado, técnicas).

Paradigma “OO” Orientación a objetos

OO no es una estrategia de diseño. El paradigma de orientación a objetos es más amplio y abarca un enfoque general para conceptualizar, diseñar y programar los sistemas de software.

■ Estrategias

Las estrategias OO cubren tanto los requisitos como el análisis, diseño y programación.

- Análisis Orientado a Objetos (OOA)
- Diseño Orientado a Objetos (OOD)
- Programación Orientada a Objetos (OOP)

■ Métodos

Las metodologías más importantes de análisis y diseño de sistemas, orientado a objetos, han terminado confluyendo en lo que es el UML (www.uml.org), bajo el respaldo del Object Management Group (www.omg.org).

Algunas de las principales metodologías, pioneras que han terminado confluyendo en el UML son:

- Object-Oriented Design (OOD), Booch.
- Object Modeling Technique (OMT), Rumbaugh.
- Object Oriented Analysis (OOA), Coad/Yourdon.
- Hierarchical Object Oriented Design (HOOD), ESA.
- Object Oriented Structured Design (OOSD), Wasserman.
- Object Oriented Systems Analysis (OOSA), Shaler y Mellor.
- Responsibility Driven Design (RDD), Wirfs-Brock, entre otros.

Paradigma “OO” Orientación a objetos

■ Enfoque OO

Este paradigma centra su foco en el concepto Objeto.

Objeto es aquello que tiene estado (propiedades más valores), comportamiento (acciones y reacciones a mensajes) e identidad (propiedad que lo distingue de los demás objetos).

La estructura y comportamiento de objetos similares están definidos en su clase común; los términos instancia y objeto son intercambiables. Una clase es un conjunto de objetos que comparten una estructura y comportamiento común.

La diferencia entre un objeto y una clase es que un objeto es una entidad concreta que existe en tiempo y espacio, mientras que una clase representa una abstracción, la "esencia" de un objeto, tal como son. De aquí que un objeto no es una clase, sin embargo, una clase puede ser un objeto.

■ Beneficios del enfoque OO

Los beneficios señalados por Booch en 1986 son:

Potencia, el uso del modelo OO ayuda a explotar el poder expresivo de los lenguajes de programación basados u orientados a objetos, como Smalltalk, Object Pascal, C++, CLOS, Ada, Java, C#

Reutilización, el uso del modelo OO favorece la reutilización, no solo del software, sino de diseños completos.

Mantenibilidad, produce sistemas que están construidos en formas intermedias estables y por ello son más resistentes al cambio en especificaciones y tecnología.

Paradigma “OO” Orientación a objetos

■ Principios del modelo OO

Fundamentales: **Abstracción, encapsulación, modularidad y jerarquía**. Booch afirma que un modelo en el que no esté presente alguno de estos principios NO es un modelo OO.

Complementarios: **Tipificación, concurrencia y persistencia**

- **Abstracción**. Simplificación en la descripción o especificación de un sistema consistente en enfatizar algunos detalles o propiedades del sistema, con detrimento o supresión de otros.
- **Encapsulación**. Ocultación de los detalles de un objeto que no contribuyen a sus características esenciales.
- **Modularidad**. Propiedad de un sistema que ha sido descompuesto en un conjunto de módulos coherentes e independientes.
- **Jerarquía o herencia**. Orden de las abstracciones organizado por niveles.
- **Tipificación**. Definición precisa de un objeto de forma tal que objetos de diferentes tipos no puedan ser intercambiados o, a lo sumo, pueden intercambiarse de manera muy restringida.
- **Concurrencia** . Propiedad que distingue un objeto que está activo de uno que no lo está.
- **Persistencia**. Propiedad de un objeto por la cual su existencia trasciende al tiempo (es decir, el objeto continua existiendo después de que su creador ha dejado de existir) y/o al espacio (es decir, la localización del objeto se mueve del espacio de dirección en que fue creado).

UML

UML es un lenguaje de modelado que permite especificar, visualizar y documentar modelos de sistemas de software.

Desde sus inicios fue concebido para ayudar a las tareas de análisis de los sistemas de software, y este es sin duda el ámbito de mayor utilización, si bien es cierto que en la actualidad también se emplea en el modelado y diseño de otros tipos de sistemas (modelos de negocio, producciones cinematográficas, etc.)

■ Tipos de diagramas UML

UML proporciona diagramas para representar modelos de las visiones estáticas y dinámicas del sistema, así como de su modularización.

REPRESENTACIONES		
Estructura estática	Comportamiento dinámico	Modularización
<ul style="list-style-type: none">▪ Diagrama de clases▪ Diagrama de objetos▪ Diagrama de componentes▪ Diagrama de despliegue	<ul style="list-style-type: none">▪ Diagrama de casos de uso▪ Diagrama de secuencia▪ Diagrama de colaboración▪ Diagrama de actividad▪ Diagrama de colaboración▪ Diagrama de estados	<ul style="list-style-type: none">▪ Paquetes▪ Subsistemas▪ Modelos

Descripción del diseño de software (SDD)

El resultado del proceso de diseño es la documentación denominada “Descripción del Diseño del Software”.

Un estándar empleado para desarrollar esta documentación de forma normalizada es el IEEE Std. 1016-1998.

■ IEEE Std. 1016-1998

Describe prácticas recomendadas para describir los diseños de software. Especifica la información que debe contener, y recomienda cómo organizarla.

Puede emplearse en software comercial, científico o militar sin limitaciones por el tamaño, complejidad o nivel de criticidad.

El estándar no establece ni limita determinadas metodologías de diseño, gestión de la configuración o aseguramiento de la calidad.

Descripción del diseño de software (SDD)

■ Ejemplo de una posible organización de la información en una SDD

- 1.- Introducción
 - 1.1 Propósito
 - 1.2 Alcance
 - 1.3 Definiciones y acrónimos
- 2.- Referencias
- 3.- Descomposición de la información
 - 3.1 Descomposición modular
 - 3.1.1. Descripción del módulo 1
 - 3.1.2. Descripción del módulo 2
 - 3.2 Descomposición de los procesos
 - 3.2.1. Descomposición del proceso 1
 - 3.2.2. Descomposición del proceso 2
 - 3.3 Descomposición de los datos
 - 3.3.1. Descripción de la entidad 1
 - 3.3.2. Descripción de la entidad 2
- 4.- Descripción de las dependencias
 - 4.1 Dependencias intermodulares
 - 4.2 Dependencias inter-procesos
 - 4.3 Dependencias de los datos
- 5.- Descripción de interfaces
 - 5.1 Interfaces entre módulos
 - 5.1.1 Interfaz del módulo 1
 - 5.1.2 Interfaz del módulo 2
 - 5.2 Interfaces entre procesos
 - 5.2.1 Interfaz del proceso 1
 - 5.2.2 Interfaz del proceso 2
- 6.- Diseño detallado
 - 6.1 Diseño detallado de los módulos
 - 6.1.1 Detalle del módulo 1
 - 6.1.2 Detalle del módulo 2
 - 6.2 Diseño detallado de los datos
 - 6.1.1 Detalle de la entidad 1
 - 6.1.2 Detalle de la entidad 2

Prácticas recomendadas

■ Trazabilidad del diseño

- Comprobación de que el diseño incluye todos los requisitos
- Comprobación de que el diseño no incluye funciones adicionales no especificadas en el SRS.
- Los resultados de la trazabilidad del diseño deben estar documentados para la reunión de revisión del diseño



■ Reunión de revisión del diseño de la arquitectura

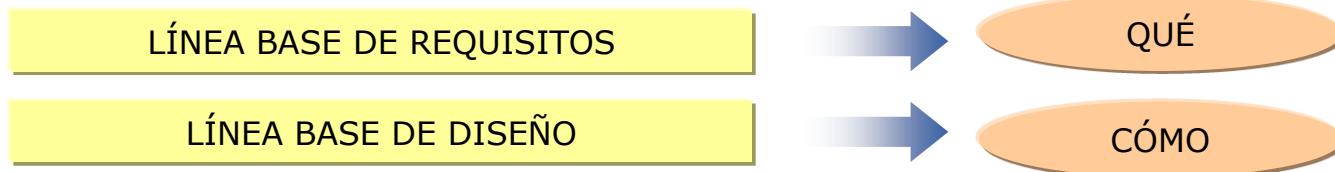
Revisión del diseño de la arquitectura

- Un equipo apropiado (Usuarios, cliente, ingeniero de soft) revisan el diseño.
- Una vez aprobado este diseño se puede comenzar a realizar el diseño detallado.

■ Verificación del diseño de la arquitectura

El diseño se verifica contra el SRS

- El proceso de verificación analiza si el diseño es incompleto, incorrecto, inefficiente, difícil de mantener, presenta un interfaz de usuario difícil de utilizar o aprender, o la documentación es de baja calidad.
- Se realiza un informe para documentar los posibles problemas encontrados y tomar nota de posibles incompatibilidades entre documentos.



Base para las tareas de planificación

La planificación comienza con la misma decisión de desarrollar un sistema de software, y es un esfuerzo continuo que termina cuando el proyecto ha concluido.

La planificación consiste en la especificación de:

- Metas y objetivos para el proyecto
- Estrategias, "política", y procedimientos

Explicándolo de otra forma es la decisión de:

- qué hacer
- cómo hacerlo
- cuando hacerlo
- quien va a hacerlo.

A lo largo del ciclo de vida, desde la concepción inicial del proyecto, la planificación se va revisando y depurando, y una vez obtenido el diseño se dispone de una base sólida.

El diseño es la representación formal de "qué hacer" y "cómo hacerlo", sobre la que se puede asignar "cuando" y "quién".

Planificación = Tareas de ingeniería de software y de gestión

La planificación del proyecto está dividido entre dos componentes relacionados:

- Planificación realizada por el gestor
- Planificación realizada por el ingeniero de sistemas



Ingeniero de Software decide:

- Qué tareas hay que realizar
- Orden y Dependencias de tareas
- Tamaño (Esfuerzo en horas)
- Solución técnica para la resolución del problema
- Qué herramientas de análisis y diseño hay que utilizar
- Riesgos técnicos
- El modelo de procesos (Técnicas)
- Actualizar la planificación cuando los requisitos o el entorno cambian.

Gestor de Proyecto decide:

- Las habilidades necesarias para realizar las tareas
- La agenda para terminar el proyecto
- El coste de esfuerzo
- Metodología para evaluar el estatus del proyecto
- Qué herramientas de planificación hay que utilizar
- Gestión de riesgos
- El modelo de procesos (Gestión)
- Actualizar la planificación cuando condiciones de gestión y entorno cambian.

Consideraciones

- El diseño es la estrategia de solución.
- Las tareas de codificación, integración y mantenimiento del sistema son la táctica.
- La estrategia debe ser adecuada a las necesidades de los usuarios (requisitos y atributos de calidad esperados).
- No surge de procesos, herramientas o lenguajes de modelado.
- Surge del talento de su creador.
- Los procesos, las herramientas y los lenguajes de modelado pueden resultar útiles como ayuda para descomponer la complejidad, y para comunicar el diseño a los participantes del proyecto.
- El talento de algunos profesionales les puede permitir manejar niveles de complejidad elevados sin necesitar apoyo de procesos, herramientas o lenguajes de modelado.
- A través del código es posible ver el diseño y la arquitectura del sistema.
- La documentación del código resulta útil para comunicar su diseño a través del espacio en sistemas en los que intervienen muchos desarrolladores, y del tiempo para facilitar su mantenimiento.
- Al emplear documentación para la comunicación del diseño es necesario trabajar con procesos suficientes para garantizar su integridad y actualidad a través de los cambios.
- El diseño no cumple su finalidad hasta que no queda plasmado en el código.
- El resultado del diseño puede fallar tanto errores en su estrategia como por distorsiones introducidas en la codificación, integración y mantenimiento.

Documentación de usuario

1.0



Conceptos generales

Formatos de distribución

- **Interno**

Documentación de usuario que se encuentra integrada y es accesible a través del software.

- **Externo**

Documentación de usuario que cuyo acceso no está integrado en la operativa del software.
El formato externo no quiere decir que emplee una distribución no informática, sino que se encuentra apartada de la operación del software.

De hecho la documentación externa puede distribuirse en CD, a través de descargas desde la web, etc.

Importancia de la calidad de la documentación

A pesar de su importancia, las organizaciones productoras de software suelen descuidar la calidad de la documentación de usuario.

En muchos casos la documentación se prepara en el último minuto, y orientando su desarrollo más como trámite que como herramienta de información para el usuario.

- Ayuda al cliente a obtener todo el valor de su inversión.

La operación de sistemas complejos sin un conocimiento detallado de los mismos puede dejar sin uso un porcentaje importante de los mismos.

- Una documentación completa y útil incrementa la facilidad de uso del sistema.

LA PRODUCCIÓN DE DOCUMENTACIÓN DE USUARIO INADECUADA ES UN PROBLEMA COMÚN EN LA INDUSTRIA DEL SOFTWARE

Conceptos generales

Tipos de documentos y contenidos posibles

La documentación de usuario de un sistema de software puede estar comprendida en uno o varios documentos físicos.

Un documento puede abordar uno o varios de los siguientes ámbitos:

- Instalación / desinstalación.
- Uso del sistema.
- Administración.

Un sistema de software puede disponer de manuales diferentes para cada uno de los subsistemas que lo componen.

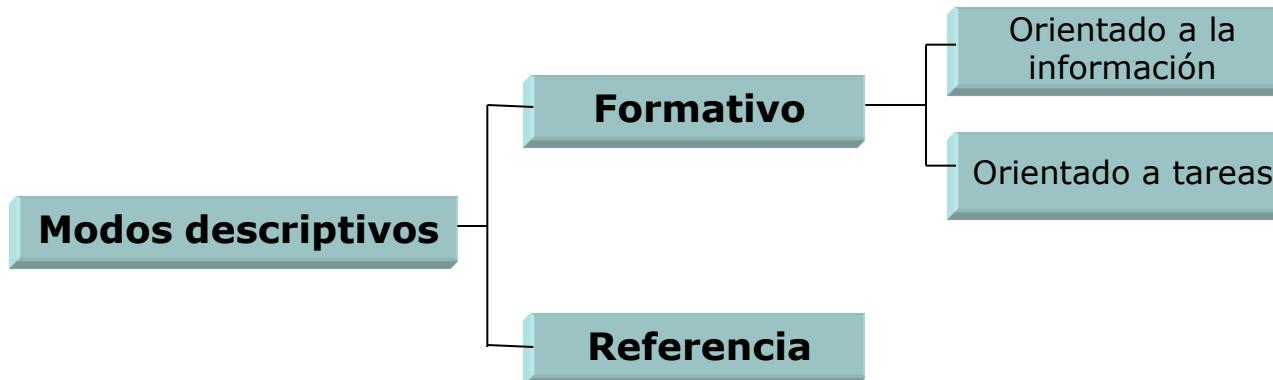
Conceptos generales

Modos descriptivos

La documentación de usuario puede adoptar dos modos narrativos diferentes: formativo o referencia, en función de la finalidad con la que el lector va a usar el texto:

- Para aprender a trabajar con el software (modo formativo)
- Para refrescar la memoria, realizando consultas puntuales (modo referencia).

A su vez, los textos formativos pueden orientar la exposición de sus contenidos para indicar al lector cómo realizar cada tarea paso a paso. (orientados a tareas), o para transmitirle la información y conocimientos técnicos necesarios para emplear el software de forma adecuada (orientados a la información).



Conceptos generales

Los factores que deben determinarse antes de desarrollar la documentación son:

- Cuáles son los documentos necesarios.
- Las características de la audiencia o audiencias de la documentación.
- El modo descriptivo de cada documento.



Documentos necesarios

En función de las características del sistema, de los usuarios e incluso de parámetros del proyecto, es necesario determinar cuáles son los documentos que deberán elaborarse.

Algunos factores que pueden resultar útiles en su determinación son:

- Naturaleza del producto, fin previsto, entorno en el que se empleará, complejidad de uso vista desde el punto de vista del usuario. Cómo de complejo es instalar, operar y mantener el sistema.
- Nivel de conocimientos de los usuarios, instaladores y personal de mantenimiento.
 - En el uso de sistemas informáticos.
 - En los procesos y negocio gestionados por el sistema.
- Tamaño y complejidad del sistema, junto con las tecnologías empleadas en su desarrollo y mantenimiento.
- Requisitos contratados.
- Ciclo de desarrollo del producto.

Así por ejemplo, un producto con desarrollo incremental puede tener como requisitos en el contrato la elaboración de manuales de usuario para cada subsistema entregado, o uno global para todo el sistema.

Conceptos generales

Características de la audiencia o audiencias

Audiencia: grupo de usuarios con características similares, tanto de operación con el sistema, como de conocimientos y experiencia informática y profesional.

Antes de comenzar el desarrollo de la documentación es importante clasificar a los usuarios del sistema por audiencias, identificando las características clave.

La documentación debe plantearse pensando en las características y necesidades de la audiencia.

Algunos criterios útiles para identificar las audiencias y sus características:

- **Educación:** ¿Cuál es el nivel educativo de la audiencia?
- **Actitud:** ¿Cuál es la actitud de la audiencia?. ¿Son reacios al uso de ordenadores?. ¿Presentan resistencia al cambio?
- **Nivel de sofisticación informática.** A título de ejemplo, Brockmann^[1] identifica cinco niveles de sofisticación informática de los usuarios, que se muestran en la página siguiente.
- **Familiaridad con los procesos** y negocio de la aplicación.

[1] Brockmann, R.J. (1990). *Writing Better Computer Documentation: From Paper to Hypertext*

Conceptos generales

Clasificación de usuarios

Nivel de sofisticación informática	Características
Inexperto^[1]	<ul style="list-style-type: none">Muy poca o ninguna experiencia con ordenadoresTratan volúmenes reducidos de informaciónNo confían en la informáticaTrabajadores concretos
Principiante	<ul style="list-style-type: none">Alguna experiencia con ordenadoresPueden comprender conceptos aisladosEmplean ejemplos concretosEmplean siempre las opciones por defecto
Intermedio	<ul style="list-style-type: none">Usuario novel con pocos meses de experiencia con ordenadoresComienza a enlazar conceptos aisladosEmplea acciones por defecto y sus opciones.
Experto	<ul style="list-style-type: none">Es la evolución de un usuario intermedio.Comprende las relaciones entre conceptos aislados.Tiene un nivel alto de autoconfianza.Comprende el lenguaje abstracto
Intermitente	<ul style="list-style-type: none">Puede ser inexperto, principiante, intermedio o experto. Trabaja muy poco con el sistema.Se conduce a través de los menús y mensajes del sistema

[1] La denominación original que hace Brockmann en su libro es "lorito" (parrot)

Conceptos generales

- Por estructura de la documentación se entiende la manera en la que la información se divide en apartados, y el orden en el que éstos se presentan.
- La estructura afecta tanto a documentos impresos como a documentos electrónicos.
- La documentación puede estructurarse en uno o varios documentos.
- La estructura debe ayudar a localizar y comprender la información.
- Cuando la documentación de un sistema se dirige a audiencias diferentes debe emplearse uno de los siguientes criterios:
 - Separar en secciones diferentes la información dirigida a audiencias diferentes, identificando en la introducción de cada sección la audiencia a la que va dirigida.
 - Separar en documentos diferentes la información para cada audiencia.

Estructura de la documentación de usuario

Recomendaciones del estándar IEEE 1063-2001 para la estructura

Estructura general

- La documentación de un sistema de software puede consistir en uno o más documentos, y cada documento puede comprender uno o varios volúmenes. Por ejemplo la referencia de comandos de un lenguaje de programación puede tener un volumen con la mitad de ellos y otro con la otra mitad.
- Son recomendables (aunque no obligatorio) las siguientes divisiones dentro de cada documento:
 - Documentos impresos: Capítulos, temas y sub-temas.
 - Documentos electrónicos: temas.

La unidad de presentación para los primeros es la página, y para los segundos la pantalla. Cada página o pantalla debe tener una identificación única (por ejemplo el título del capítulo y el nº de página), que debe aparecer al imprimirla el usuario.
- Los documentos impresos no deben tener más de tres niveles de subdivisión dentro de un capítulo. Así, por ejemplo, un sub-tema con nivel 1.2.3.4 debe ser el mayor nivel de sub-división.
- Los documentos electrónicos deben permitir acceder a cualquier información con menos de 3 saltos (links) desde la página inicial.
- Los documentos que contengan información en varios modos descriptivos (formativo y de referencia) deben estar claramente separados en capítulos diferentes, o al menos en temas diferentes o manteniendo formatos tipográficos distintos.
- La documentación en modo de referencia debe estar estructurada para facilitar la búsqueda y acceso a los diferentes elementos. Por ejemplo, ordenando alfabéticamente una lista de comandos, o de informes de error.

Estructura de la documentación de usuario

Recomendaciones del estándar IEEE 1063-2001 para la estructura

Cada documento debe incluir

INFORMACIÓN IDENTIFICATIVA

- Título del documento
- Versión del documento y fecha de publicación
- Nombre del producto de software y versión
- Organización que edita el documento

TABLA DE CONTENIDOS (índice)

INTRODUCCIÓN

- Audiencia
- Alcance y propósito del documento
- Descripción general del propósito y funcionalidad del software, así como del entorno de operación

Estructura de la documentación de usuario

Recomendaciones del estándar IEEE 1063-2001 para la estructura

Información crítica



- La información crítica debe aparecer en una ubicación destacada de la documentación.
- Las advertencias de carácter general deben incluirse en la introducción del documento.
- Las advertencias particulares deben aparecer en la misma página o pantalla en la que se da información del procedimiento implicado

Contenido



- La información debe ser completa
 - Si es en modo formativo debe incluir descripción suficiente para que los individuos con menos experiencia de la audiencia puedan realizar eficientemente las funciones descritas.
 - En modo referencia se deben incluir todas las instancias de los elementos seleccionados.
- La información debe ser actual y acorde a la versión del software indicada.

Estructura de la documentación de usuario

Recomendaciones del estándar IEEE 1063-2001 para la estructura

Componentes recomendados para la documentación de usuario

COMPONENTE	¿OBLIGATORIO?
Información identificativa	Sí
Tabla de contenidos	Sí, en documentos de más de 8 páginas
Lista de imágenes	Opcional
Introducción	Sí
Información para el uso de la documentación	Sí
Información conceptual de las funcionalidades generales	Sí

Estructura de la documentación de usuario

Recomendaciones del estándar IEEE 1063-2001 para la estructura Componentes recomendados para la documentación de usuario

COMPONENTE	¿OBLIGATORIO?
Procedimientos	Sí, en modo formativo
Información de comandos de software	Sí, en modo referencia
Mensajes de error y resolución de problemas	Sí
Glosario	Sí, si la documentación incluye términos desconocidos para la audiencia
Fuentes de información adicionales	Opcional
Índice	Sí, en documentos de más de 40 páginas
Capacidad de búsqueda	Sí, en documentación sobre formato electrónico

Estructura de la documentación de usuario

Recomendaciones generales

Legibilidad

La documentación impresa y electrónica debe resultar legible para el usuario, teniendo en cuenta la distancia que se empleará en las condiciones normales del entorno de consulta. Deben emplearse tipos de letra y colores fácilmente legibles sobre el color de fondo empleado. La documentación impresa debe mantenerse legible si el usuario agranda o reduce la ventana de visualización.

El estándar IEEE 1063, por ejemplo, da algunas recomendaciones específicas como:

- No abusar de las letras mayúsculas, indicando que no se emplee en más de 25 palabras seguidas.
- No emplear en los textos electrónicos letras menores de 3mm. (aprox. 7,5 puntos).

En la documentación electrónica deben considerarse también las combinaciones de colores previendo el caso de que el usuario vaya a imprimirla en una impresora monocromo.

Corrección

Los textos deben ser léxica, ortográfica y sintácticamente correctos.

Consistencia en la terminología y en el formato

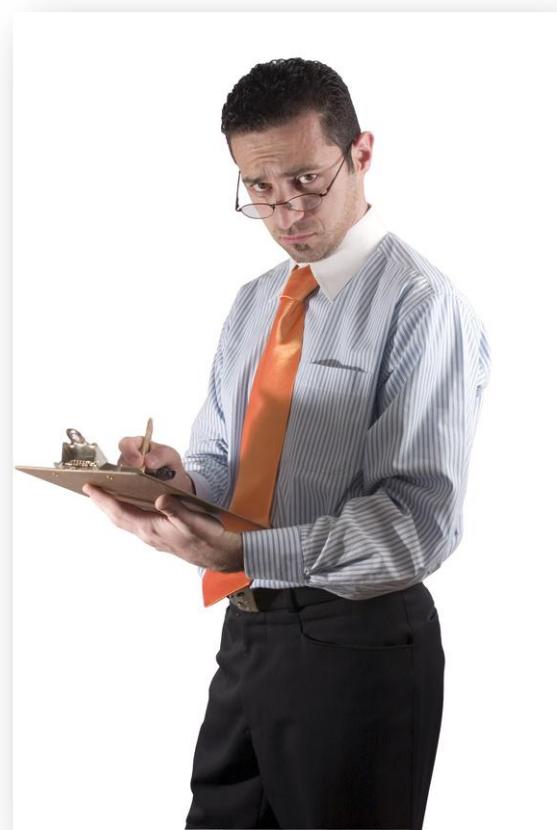
El documento debe emplear de forma consistente la terminología empleada para nombrar los elementos del interfaz de usuario, nombres de operaciones, funciones, procesos y conceptos claves del sistema.

Asimismo debe respetar a través de todo el documento unas características gráficas homogéneas: cabeceras, pies de página, estilos de títulos y párrafos, márgenes, estilos de viñetas, etc.

Las convenciones empleadas para mostrar las advertencias y notas resaltadas deben presentarse con las mismas características de estilo en todo el documento.

verificación y validación

1.0



Introducción

La complejidad del desarrollo de un sistema de software

Durante el desarrollo de un sistema de software, antes de producir el producto ejecutable final, se generan múltiples productos intermedios:

- Especificaciones de requisitos.
- Diseño.
- Planes de prueba.
- Código.

Al mismo tiempo el producto final se genera a través de las tareas y actividades realizadas en diferentes procesos:

- Adquisición.
- Suministro.
- Desarrollo.
- Etc.

Los errores introducidos en los productos intermedios se transmiten al producto final.

Las calidad del producto final depende de la calidad imprimida en las diferentes tareas, actividades y procesos.

Introducción

Verificación y validación

Aunque en el lenguaje coloquial estos términos pueden considerarse sinónimos, en el contexto de la ingeniería del software tienen significados diferentes:

- **Verificación:** Determinación con medios objetivos y repetibles de que un elemento satisface los requisitos.
- **Validación:** Determinación con medios objetivos y repetibles de que un elemento puede emplearse para el fin que tiene asignado.

Coste de la validación y verificación

Las actividades de validación y verificación en un proyecto requieren un esfuerzo, que debe estimarse y planificarse de forma apropiada en el plan del proyecto.

En función de las características del proyecto los costes directos e indirectos suelen situarse en rangos del 5% al 40%^[1] del coste total del proyecto.

[1] Boehm "Software Engineering Economics" 1981 – Marciniaj J.J. Encyclopedia of Software Engineering 1994 – Neal, R.D. "A Case Study of IV & V Cost Efectiveness" 1997

Conceptos

Verificación y validación

Es la disciplina de gestión y actividad técnica para garantizar que el software operará según lo especificado en los **requisitos** y las **necesidades del usuario**, que se lleva a cabo a través de:

- Proceso proactivo de análisis, revisión y pruebas.
- Gestión en paralelo con las actividades de desarrollo para garantizar que el software cumple los objetivos de corrección, calidad, rendimiento, agendas y usabilidad.

Verificación: Método empleado para garantizar que el producto resultante de una actividad o fase del desarrollo cumple con los requisitos de esa actividad o fase en lo relativo a corrección, calidad, rendimiento, cumplimiento de agendas y usabilidad.

Validación: Método que garantiza que el producto es conforme para el uso que tiene previsto.

Verificación

- ¿Se está construyendo adecuadamente el producto?
- La verificación se realiza “contra” el entorno de desarrollo o del proyecto.

Validación:

- ¿Se está construyendo el producto adecuado?
- La validación se realiza “contra” el entorno cliente, donde el producto debe cumplir su finalidad.

Conceptos

Verificación: Método empleado para garantizar que el producto resultante de una actividad o fase del desarrollo cumple con los requisitos de esa actividad o fase en lo relativo a corrección, calidad, rendimiento, cumplimiento de agendas y usabilidad.

Validación: Método que garantiza que el producto es conforme para el uso que tiene previsto.

Verificación y validación en los procesos de desarrollo

5. Procesos primarios

5.1 Adquisición

5.2 Suministro

5.3
Desarrollo

5.3
Operación

5.3
Mantenimiento

6.- Procesos de soporte

6.1 Documentación

6.2 Gestión de la configuración

6.3 Control de calidad

6.4 Verificación

6.5 Validación

6.6 Reuniones

6.7 Auditoría

6.8 Resolución de problemas

7. Procesos organizacionales

7.1 Gestión

7.2 Infraestructura

7.3 Mejora

7.4 Formación

Verificación y validación en los procesos de desarrollo

Procesos de soporte

Las actividades de verificación y validación pueden realizarse en diversas fases y sobre diversos productos del desarrollo.

Por esta razón están clasificados como procesos de soporte, que son llamados por otros procesos del ciclo de vida.

Así, por ejemplo, si el estándar 830 de IEEE se emplea para regular cómo debe hacerse el documento de especificación de requisitos del software, resulta posible y probable que durante el curso del desarrollo se revise el documento para ver si se ajusta a las características definidas en el estándar (verificación).

También resulta posible (y muy recomendable) que se contraste el documento generado con interlocutores del cliente para comprobar que lo escrito refleja sus necesidades (validación).

Si la agenda del plan de proyecto preveía disponer del diseño en la fecha X, parece lógico que regularmente se **verifique** si los procesos están inyectando causas de problemas en el proyecto (incumpliendo agendas, en este caso).

El esfuerzo de verificación y validación **debe ajustarse a las características del proyecto**. En algunos casos resultará aconsejable o necesario generar un “*plan de verificación y validación del software*” que se ajuste a estándares como el IEEE 1012-1998, y en otros casos bastará con tareas básicas de verificación y validación, contempladas y dimensionadas en el plan del proyecto.

Relación entre V&V y Aseguramiento de Calidad

Aseguramiento de la calidad

La función del Aseguramiento de la Calidad es garantizar que la organización realiza el trabajo conforme a los procedimientos y métodos establecidos para el proyecto.

IEEE Std. 730-1998

Relación con Verificación y validación.

Es frecuente encontrar cierta confusión entre estas dos áreas.

El Aseguramiento de la calidad (SQA) es una metodología interna cuya principal finalidad es garantizar que el flujo del trabajo cumple con las normas que tiene impuestas el desarrollador (por su normativa interna, por imposición del cliente, etc.).

SQA no evalúa el producto producido en esa fase o en ese proyecto, sino el proceso que lo ha producido. No mira el producto, mira el proceso.

Validación y Verificación enfocan su análisis en atributos del producto generado.

Adecuación del V&V a las características del proyecto

Definiendo los objetivos

Las consideraciones que deben contemplarse para evaluar la planificación de las actividades de Validación y Verificación son:

- **Nivel de integridad del proyecto.**

Concepto desarrollado en las páginas siguientes. Mide la “criticidad” del software.

- **Mínimo de tareas recomendadas para el nivel de integridad del proyecto.**

La regulación interna de la organización desarrolladora puede determinar qué tareas de V&V deben realizarse para cada nivel de integridad.

El estándar IEEE 1012 define 4 niveles de integridad e incorpora una tabla en la que se estipulan las actividades mínimas de V&V en función del nivel.

- **Intensidad y rigor necesarios** en las tareas de Validación y verificación.

El nivel de integridad no sólo determina qué tareas deben realizarse, sino también su intensidad y rigor. Por ejemplo, si lo realiza el propio personal de desarrollo, otro equipo de desarrollo diferente, o incluso una organización externa (auditora).

- Los **criterios** que se emplearán en las tareas de V&V para establecer los **parámetros** mínimos de corrección, consistencia, precisión

Adecuación del V&V a las características del proyecto

Criticidad del producto

El estándar IEEE 1012 establece que el “plan de validación y verificación del software” (SVVP) debe especificar un método para clasificar el nivel de integridad del software de cada subsistema de software del proyecto.

Adecuación del V&V a las características del proyecto

Análisis de criticidad

Proceso para identificar, evaluar y categorizar el grado de criticidad de los elementos del producto de software.

La definición formal incluida en el estándar IEEE 1012-1998 es:

"La evaluación estructurada de las características del software (p. ej. Seguridad, complejidad, rendimiento) para determinar la severidad del impacto de un fallo del sistema, de su degradación o de su no cumplimiento con los requisitos o los objetivos del sistema."

En otras palabras:

Si el sistema falla, se degrada o no consigue realizar las funciones de los requisitos, ¿qué impacto tiene en la seguridad o en el rendimiento?

Análisis de criticidad

Análisis de daños
(hazard analysis)

Análisis de riesgos
(risk analysis)

Adecuación del V&V a las características del proyecto

Análisis de criticidad: análisis de daños

La definición formal del análisis de daños (a nivel de sistema) es:

"Análisis de fuentes potenciales de daños o de situaciones que pueden generar daños en términos de daños a personas, daños a la salud, o al entorno, o una combinación de ellos".

IEC 60300-3-9, 1995

No obstante el estándar para validación y verificación IEEE 1012-1998 da una definición más amplia que incluye también pérdidas económicas, fallo en la misión del sistema, o impacto social adverso. Para nosotros "daño" en el marco de validación y verificación de software incluye por tanto:

- **Daños a las personas.**
- **Daños al medio ambiente.**
- **Pérdidas económicas.**
- **Fallo en la finalidad del sistema.**
- **Impacto social adverso.**

Para realizar el análisis de daños **deben identificarse las consecuencias que pueden ocasionar los fallos en el software**. Es posible que no generen daños físicos, pero sí en términos de pérdidas económicas (para el desarrollador, para el cliente, para los usuarios), o de impacto social adverso (desprestigio del cliente, del desarrollador, de los usuarios, de terceros).

Adecuación del V&V a las características del proyecto

Análisis de criticidad: análisis de riesgos

Riesgo: probabilidad de que se produzca un daño identificado

En el desarrollo de un sistema de software se pueden producir adversidades que afecten a:

- Los planes del proyecto.
- Al producto o subproductos del desarrollo.

Los riesgos inherentes a un proyecto suelen ser de tres naturalezas:

- Intrínsecos al sistema que se desarrolla
- Derivados de las particularidades de desarrollo del software.
- Propios del desarrollo de proyectos.

Adecuación del V&V a las características del proyecto

Análisis de criticidad: análisis de riesgos

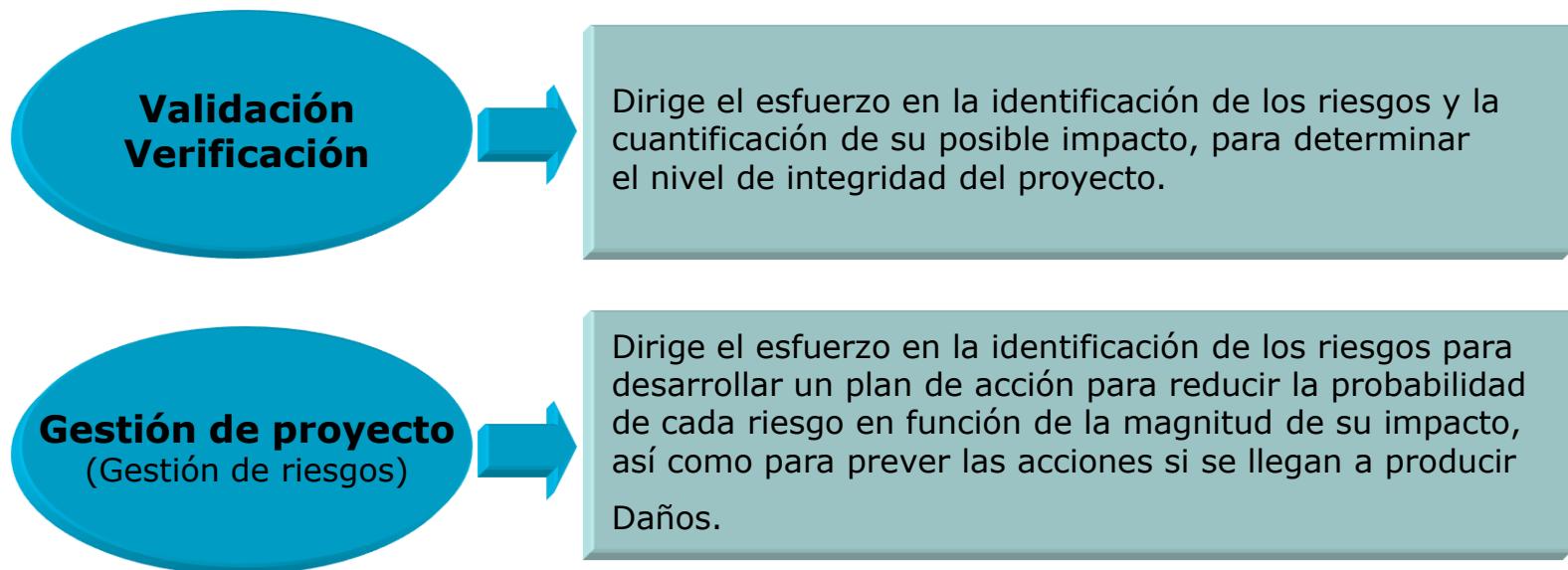
NATURALEZA DEL RIESGO	CAUSAS TÍPICAS
Propios del sistema	Los identificados en el análisis de daños
Propios del desarrollo de software	<ul style="list-style-type: none">▪ Complejidad innecesaria Complejidad intrínseca del diseño mayor de la necesaria▪ Baja calidad Incumplimiento de estándares necesarios▪ Inestabilidad de los requisitos▪ Problemas con herramientas y métodos Inestabilidad, bugs en compiladores, etc.▪ Comportamiento imprevisto de los interfaces Interfaces con hardw. y softw. externo en la implementac.▪ Inestabilidad y cambio rápido de las plataformas tecnológicas
Propios de los desarrollos por proyecto^[1]	<ul style="list-style-type: none">▪ Presión en costes y agendas.▪ Lagunas en planificación y gestión.▪ Retrasos en subcontrataciones.

[1] En los proyectos de software se suelen dar con mayor intensidad los riesgos típicos indicados.

Adecuación del V&V a las características del proyecto

Análisis de criticidad: análisis de riesgos

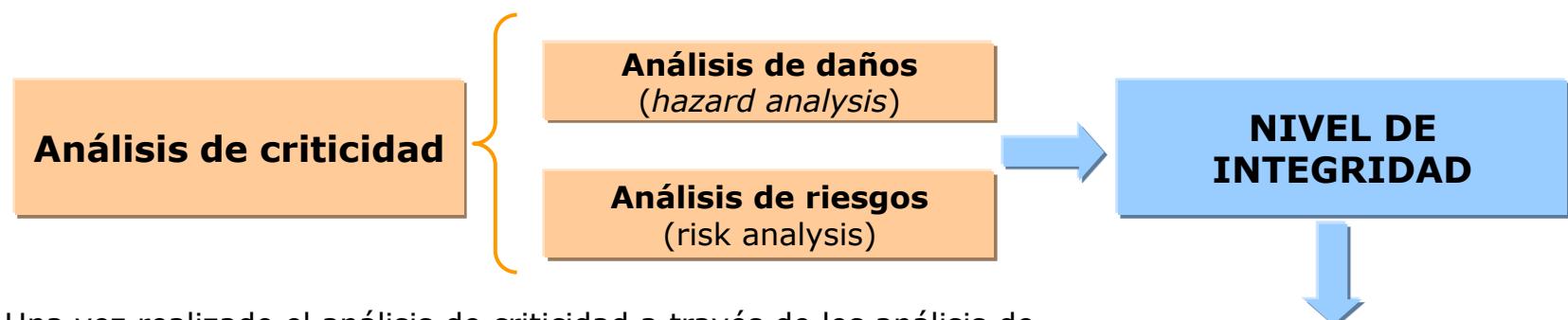
Validación y Verificación no gestionan las causas de los riesgos. Esa gestión pertenece a la “gestión de riesgos”, dentro de la “gestión del proyecto”.



[1] En los proyectos de software se suelen dar con mayor intensidad los riesgos típicos indicados.

Adecuación del V&V a las características del proyecto

Niveles de integridad



Una vez realizado el análisis de criticidad a través de los análisis de daños y de riesgos, resulta posible establecer el nivel de integridad del proyecto y adecuar a él las tareas de validación y verificación.



Adecuación de las tareas de VALIDACIÓN y VERIFICACIÓN

El nivel de criticidad depende de dos factores:

- MAGNITUD DEL DAÑO POSIBLE
- POSIBILIDAD DE MITIGACIÓN DEL DAÑO

Adecuación del V&V a las características del proyecto

Niveles de integridad

El estándar IEEE 1012-1998 define 4 niveles de integridad.

En el borrador de 2004 las definiciones que se recogen para cada uno son:

Nivel	Dimensión del daño por fallo del Software	Mitigación aplicable
4	<ul style="list-style-type: none">▪ Pérdida de vida▪ Pérdida del sistema▪ Graves pérdidas económicas o sociales	No es posible mitigar los daños producidos
3	<ul style="list-style-type: none">▪ El sistema no realiza el fin previsto ni en todo ni en parte.▪ Graves pérdidas económicas o sociales	Es posible una mitigación parcial de los daños producidos
2	<ul style="list-style-type: none">▪ No se pueden realizar funcionalidades parciales del sistema.▪ Pérdidas económicas o sociales importantes	Se pueden mitigar los daños producidos
1	<ul style="list-style-type: none">▪ Una determinada funcionalidad del sistema no se realiza.▪ Consecuencias mínimas	No es necesario mitigar los daños

El modelo del estándar resulta válido, pero cualquier modelo, adecuado a las circunstancias del sistema y del entorno de desarrollo, puede resultar igualmente válido.

Adecuación del V&V a las características del proyecto

Independencia

La determinación de las personas responsables de las tareas de verificación y validación, depende de:

- **Características y naturaleza del proyecto**
- **Nivel de integridad**

La independencia puede aplicarse a distintas áreas del proyecto:

- **Independencia de gestión**

Las personas que realizan las tareas de verificación y validación se gestion al margen de la organización que realiza el desarrollo. Tienen la autoridad para tomar decisiones sobre el trabajo de V&V, incluyendo qué elementos se van a analizar, con qué herramientas. Facilitan la información de sus conclusiones tanto a los desarrolladores como al adquiriente, pero sólo el adquiriente puede modificar la línea de trabajo de validación y verificación.

- **Independencia técnica**

Las personas que analizan el proyecto son ajenas al grupo de desarrollo. Emplean sus propias herramientas, métodos y recursos.

- **Independencia financiera**

Las tareas de verificación y validación cuentan con presupuesto propio, y la autoridad para modificar su presupuesto está fuera de la organización desarrolladora.

Adecuación del V&V a las características del proyecto

Tipos de independencia

La validación y verificación independiente (IV&V) se clasifica en 4 tipos, en función del rigor con el que se realiza: clásica, modificada, interna y doméstica. Los proyectos con nivel de integridad 4 requieren independencia rigurosa en todas sus áreas.

El estándar IEEE 1012-1998 muestra con la siguiente tabla el grado de independencia generalmente proporcionado por cada tipo, para cada faceta del proyecto.

Tipos de independencia	Áreas		
	Gestión	Técnica	Financiera
CLÁSICA	I	I	I
MODIFICADA	I-R	I	I
INTERNA	I-R	I-R	I-R
DOMÉSTICA	I-M	I-M	I-M

- I: Independencia rigurosa
- IR: Independencia con reparos
- IM: Independencia mínima

Adecuación del V&V a las características del proyecto

Tipos de independencia

IV&V CLÁSICA

Normalmente requerida para proyectos con nivel de integridad 4.
Exige rigurosa independencia en las 3 áreas del proyecto.

IV&V MODIFICADA

Suele emplearse en proyectos con nivel de integridad 3.
El desarrollo y la V&V lo realizan organizaciones diferentes, pero la responsabilidad de la gestión en el proyecto es única, y es la que recibe la información de ambas partes. No obstante, los presupuestos y el personal técnico están separados.

IV&V INTERNA

Se emplea cuando el equipo de V&V pertenece a la misma organización desarrolladora, pero en la forma de una entidad diferenciada del grupo de desarrollo del proyecto.

IV&V DOMÉSTICA

Se emplea personal de la organización desarrolladora para realizar la V&V. El personal de desarrollo y de validación y verificación trabaja conjuntamente. No se puede garantizar la independencia técnica, y la gestión y el presupuesto son únicos para desarrollo y V&V.

Verificación y validación en el ciclo de vida del software

Las tareas de verificación y validación se deben realizar en paralelo con los procesos del ciclo de vida, incluyendo también la gestión del proyecto.



GESTIÓN

El objetivo del trabajo de verificación y validación es garantizar que el software tiene la integridad requerida.

Este trabajo debe realizarse de forma integrada en la gestión del proyecto y puede comprender:

- Desarrollo del plan de validación y verificación.
- Valoración de las modificaciones.
- Supervisión de las actividades de verificación y validación
- Planificación, monitorización y control del trabajo de validación y verificación.
- Etc.

Verificación y validación en el ciclo de vida del software

ADQUISICIÓN

En el proceso de adquisición el trabajo de verificación y validación debe incluir siempre

- Revisión de la descripción del sistema.

En función del nivel de integridad del proyecto, puede cubrir también:

- Valoración de la dimensión y alcance de los trabajos de V&V.
- Planificación de la comunicación entre los trabajos de V&V y la organización desarrolladora.

SUMINISTRO

El proceso de verificación y validación comienza cuando un suministrador decide atender la petición de adquisición. V&V se enfoca en determinar si la documentación e información facilitada por el adquiriente es consistente y si, en opinión del suministrador, la solución satisfará las necesidades de los clientes.

Este proceso se denomina “**verificación del contrato**”.



Verificación y validación en el ciclo de vida del software

DESARROLLO

La verificación y validación en el desarrollo centra su actividad en 6 tareas, que corresponden a las 5 típicas de los ciclos de desarrollo en cascada, más instalación.

V&V EN EL PROCESO DE DESARROLLO

CONCEPTO

REQUISITOS

DISEÑO

IMPLEMENTACIÓN

PRUEBAS

INSTALACIÓN

Si en el ciclo de vida empleado por el proyecto, la incorporación de estas actividades está modificada, el proceso de verificación y validación también se adecuará a las características del proyecto.

V & V CONCEPTO

La verificación y validación del concepto trabaja sobre la descripción del sistema, lleva a cabo el análisis de criticidad, estudiando y evaluando daños y riesgos; y genera o actualiza el plan de validación y verificación.

Verificación y validación en el ciclo de vida del software

V & V REQUISITOS

En esta fase, la verificación y validación comprueba el principal producto generado en esta fase: la especificación de requisitos del software.

Se analizan las propiedades de calidad

DEL DOCUMENTO

- Completo
- Correcto
- Consistente
- Modificable
- Trazable

DE LOS REQUISITOS

- Posibles
- Necesarios
- Priorizados
- Correctos
- Verificables

V & V DISEÑO

Comprobación de que el diseño realizado comprende todos los requisitos sin omisiones y sin complejidad innecesaria. Los aspectos generales que se analizan son:

- Trazabilidad entre requisitos y diseño.
- No hay omisiones ni añadidos.
- El diseño es apropiado para los objetivos deseados del sistema.
- El diseño es conforme con los estándares, prácticas y convenciones acordadas para el proyecto
- El diseño es comprensible por el equipo de desarrollo y el posterior de mantenimiento.
- Contiene información suficiente para realizar las pruebas de unidad y de integración.
- La documentación está completa, incluyendo gráficas o especificaciones necesarias.

Verificación y validación en el ciclo de vida del software

V & V IMPLEMENTACIÓN

La implementación transforma la descripción del diseño en componentes que juntos integran el producto final de software. La labor de verificación y validación comprueba:

- Conformidad del código
 - Con las especificaciones del diseño
 - Con los estándares aplicables
- La idoneidad del código para obtener el producto con el nivel de integridad deseado.

V & V PRUEBAS

La verificación y validación de las pruebas garantiza que se han cumplido los requisitos del sistema y del software, alcanzando los niveles de integridad requeridos.

En sistemas con niveles de integridad 3 y 4 es necesario que el equipo de verificación y validación realice los planes y procesos de prueba, así como su ejecución.

Para niveles 1 y 2 es suficiente con verificar las pruebas realizadas por el equipo de desarrollo.

V & V INSTALACIÓN

Se comprueba el rendimiento del sistema de software al ejecutarse en el entorno del cliente, así como que los procedimientos de instalación son correctos.

Verificación y validación en el ciclo de vida del software

V & V OPERACIÓN

Una vez instalado y puesto en servicio el sistema de software, la verificación y validación valora el impacto que los cambios pueden suponer en el nivel de integridad del sistema, o los riesgos o daños que pueden introducir.

Incluye la monitorización y evaluación del entorno de operación.

V & V MANTENIMIENTO

Una vez puesto en servicio el sistema de software, las modificaciones del entorno, o la presencia de errores, o la necesidad de ampliar su funcionalidad requerirán emprender tareas de mantenimiento, que en esencia, y a menor escala son pequeñas acciones de desarrollo que pueden introducir modificaciones en el nivel de integridad, y requerir revisiones en los análisis de criticidad, daños y riesgos, así como requerir posteriores acciones de verificación y validación sobre las modificaciones de requisitos, diseño, desarrollo y pruebas.

Mantenimiento

1.0



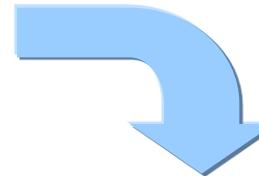
Introducción

La complejidad del mantenimiento de un sistema de software



CONSUME MUCHOS RECURSOS

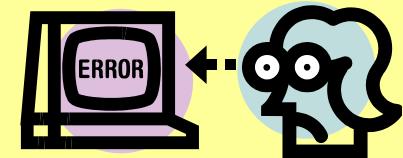
El mantenimiento consume más del 60% del coste de todo el ciclo de vida



EL SISTEMA YA ESTÁ EN USO

Las actividades de mantenimiento resultan muy visibles para el cliente.

Pueden afectar negativamente a la imagen de la organización.



**ES UNA ACTIVIDAD CRÍTICA
GENERALMENTE INFRACONSIDERADA**

Introducción

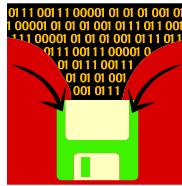
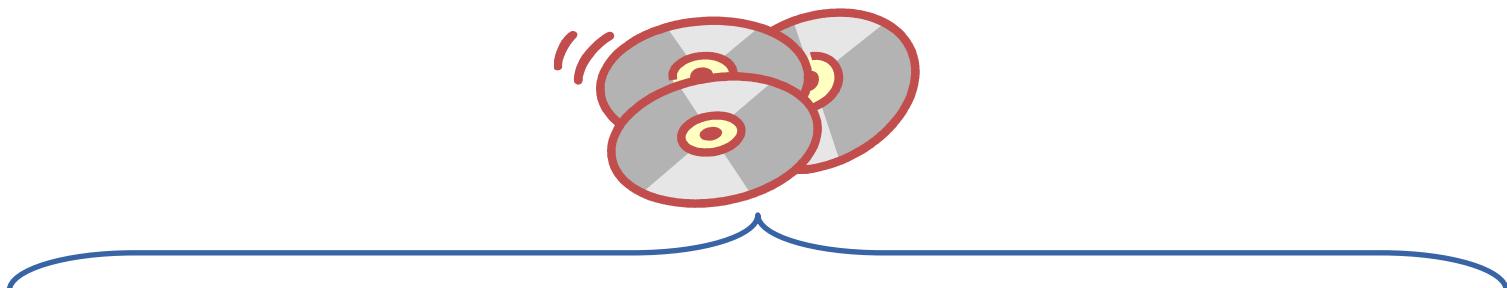
Definición

Modificación de un *producto de software*, después de su entrega para corregir errores, mejorar el rendimiento u otros atributos o adaptar el producto a cambios del entorno.

IEEE Std. 1219-1998

Producto de software no comprende sólo el código. Incluye también la documentación y los datos de configuración

PRODUCTO DE SOFTWARE



Código



Datos de configuración
y estructuras de datos



Requisitos, documentos
de análisis, plan de V&V...



Manuales y
documentación de usuario

Tipos de mantenimiento

El mantenimiento del software se clasifica generalmente en tres categorías, en función de cuál es la causa que motiva el cambio:

- Adaptativo
- Correctivo
- Perfectivo

ADAPTATIVO

Permite al software continuar en funcionamiento, adaptándose a cambios producidos en su entorno de operación.

Los cambios típicos se suelen centrar en el hardware con el que interactúa, en el sistema operativo, o en formatos de datos que recibe o envía.

CORRECTIVO

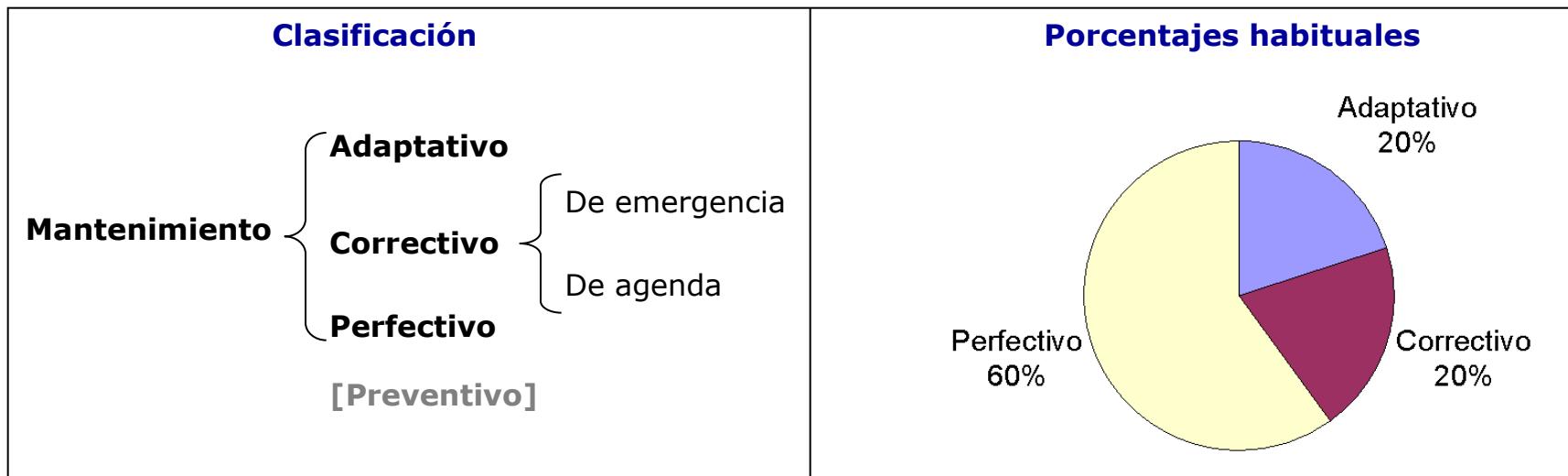
Tiene como finalidad corregir fallos o problemas. Dentro del mantenimiento correctivo se suele diferenciar entre “de emergencia” o “de agenda”; en función de la urgencia con la que deba aplicarse la solución.

En algunas ocasiones el cliente necesita urgentemente la reparación del fallo, y en otras, puede seguir operando con el error presente, y esperar a la próxima versión que normalmente incluye cambios acumulados en la agenda de mantenimiento, tanto de tipo adaptativo, como correctivo y perfectivo.

Tipos de mantenimiento

PERFECTIVO

Se realiza para dar respuesta a nuevos requisitos del cliente, o para mejorar el rendimiento del sistema.



PREVENTIVO

En su versión de 1993, el estándar IEEE 1229 incluía también en su clasificación el mantenimiento preventivo como aquel que se realiza para evitar la aparición futura de errores, o para mejorar la integridad de producto en prevención de éstos. Algunos textos lo consideran como un 4º tipo de mantenimiento, y otros lo incluyen como mantenimiento correctivo.

Dificultad del mantenimiento

El mantenimiento es una de las fases más difíciles del ciclo de vida, y generalmente no está lo suficientemente reconocida.

Las principales razones de esta situación son:

- En las organizaciones de software no aparece asociada a nuevas oportunidades de negocio, que es sin duda un aspecto mucho más atractivo para sus gestores.
- Los trabajos de mantenimiento suelen tener asignados sus propios presupuestos pre-establecidos, y se ven como un “negocio normal”, por lo que no suelen atraer la atención de la actividad del negocio.
- El personal técnico suele preferir trabajar en proyectos nuevos y no en mantener sistemas ya desarrollados.

En muchos sentidos, el mantenimiento resulta más difícil tanto desde el punto de vista técnico como de gestión del proyecto. Algunos de los factores que hacen del mantenimiento un proceso difícil son:

- Posibilidad de introducción de errores en cascada o distorsionar funcionalidades ya implementadas al insertar nuevas modificaciones.
- El equipo de mantenimiento debe tener un conocimiento global del producto.
- Las pruebas suelen resultar especialmente complicadas porque generalmente las limitaciones de tiempo no hacen posible ejecutar pruebas completas de regresión.
- Desde el punto de vista de gestión, las tres categorías de mantenimiento (correctivo, perfectivo y adaptativo) suelen realizarse de manera simultánea, con gestión de prioridades de cada petición de cambio, y respetando la gestión de la configuración del sistema.

Dificultad del mantenimiento

Dificultad por degradación del sistema

Cuanto mejor diseñado, codificado y documentado está un sistema, más fácil resulta su mantenimiento.

Las propias tareas de mantenimiento tienden a degradar el diseño, la limpieza del código y su documentación, generando de esta forma una espiral que se retroalimenta y que con el tiempo incrementa la dificultad de mantenimiento de un sistema.

Los factores que favorecen esta situación, y que por tanto es necesario gestionar adecuadamente son:

- Los mitos ya apuntados de no otorgar al mantenimiento la importancia y rigor necesarios.
- Las presiones de tiempo y recursos con las que suelen ejecutarse.
- La consideración por parte del personal técnico de tareas de "segunda división"

Áreas de degradación creciente

Diseño

Código

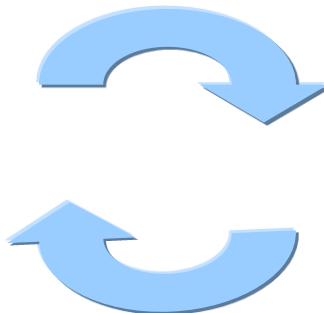
Datos

Documentación

Dificultad del mantenimiento

Dificultad por degradación del sistema

DIFICULTAD DEL MANTENIMIENTO



SISTEMA MÁS DIFÍCIL DE MANTENER

Degradación del sistema

- Diseño cada vez más turbio
- Código “ parcheado” cada vez más sucio
- Estructurad de datos van perdiendo su normalización e integridad referencial
- Actualización deficiente o nula de la documentación

Factores agravantes

- Escasa concienciación organizacional de la relevancia del mantenimiento.
- Peticiones de cambios con presión de fechas y presupuestos.
- Consideración del personal técnico de que se trata de un trabajo de “segunda categoría”

Dificultad del mantenimiento

Las tareas de mantenimiento deben ejecutarse dentro de un marco de gestión, de igual forma que si se trata el desarrollo de un sistema nuevo.

También es frecuente que en este aspecto también el mantenimiento suele ser tratado como "la cenicienta" en los proyectos de software, y generalmente resulta más difícil la gestión de un proyecto de mantenimiento que la de un desarrollo nuevo. De hecho puede ocurrir que dentro del mantenimiento de un sistema se incluya también el desarrollo de un nuevo sub-sistema para ampliar su funcionalidad.

Por tanto todas las tareas e indicaciones propias de la gestión de proyectos de software son aplicables a los proyectos de mantenimiento: estimación del esfuerzo necesario, identificación de procesos necesarios, planificación de costes y agendas, gestión de riesgos, etc.

Las actividades de mantenimiento deben realizarse con técnicas de gestión de proyectos

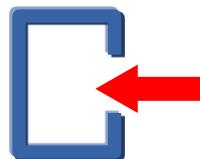
Las 7 fases del mantenimiento

Para identificar y comprender las actividades que deben tenerse en cuenta en el mantenimiento, los pasos que deben seguirse y las herramientas y métodos que deben emplearse, resulta muy útil considerar que los procesos de cambio o modificaciones de un sistema de software comprenden 7 fases:

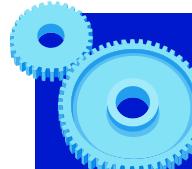
- Identificación clasificación y priorización del problema o de la modificación.
- Análisis.
- Diseño.
- Implementación.
- Pruebas de sistema y de regresión.
- Pruebas de aceptación.
- Entrega.

Al realizar tareas de mantenimiento, en cada una de estas fases deben considerarse los siguientes elementos:

EN CADA FASE



Entradas



Procesos



Control



Salidas

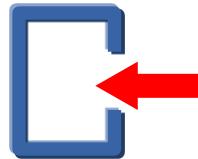


Métricas

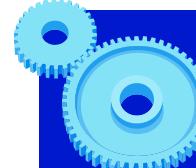
Las 7 fases del mantenimiento

1.- Identificación del problema, clasificación y priorización

Cada petición de cambio debe identificarse, clasificarse y asignarle una prioridad, teniendo en cuenta qué tipo de mantenimiento implica (correctivo, adaptativo, perfectivo y si es de emergencia)



Entradas



Procesos



Control



Salidas



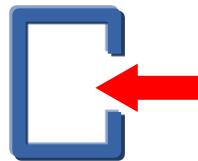
Métricas

- | | | | | |
|--|--|--|--|--|
| <ul style="list-style-type: none">▪ Petición de cambio | <ul style="list-style-type: none">▪ Asignar nº de identificación▪ Clasificar por tipo de mantenimiento▪ Analizar y determinar se se acepta rechaza o pospone▪ Primera estimación de su magnitud▪ Priorizar la modificación▪ Asignar la petición a qué bloque de modificaciones prevista va a ir | <p>Una vez identificada la petición de cambio, debe quedar registrada en un registro de peticiones de cambio</p> | <p>Petición de cambio validada que queda en un registro con la siguiente información:</p> <ul style="list-style-type: none">▪ Definición del problema o del nuevo requisito▪ Evaluación▪ Tipo de mantenim.▪ Prioridad inicial▪ Estimación inicial de recursos necesarios | <ul style="list-style-type: none">▪ Nº de omisiones en la P.C.▪ Nº de P.C. enviadas▪ Nº de peticiones de cambio duplicadas▪ Tiempo invertido en la validación. <p>Factores medidos: corrección y mantenibilidad</p> |
|--|--|--|--|--|

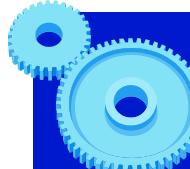
Las 7 fases del mantenimiento

2.- Análisis

La fase de análisis emplea la relación de peticiones de cambio registradas y validadas para analizar su viabilidad, alcance de las modificaciones y preparar un plan preliminar de diseño implementación y entrega

**Entradas**

- Petición de cambio validada
- Estimación inicial de recursos y demás información registrada.
- Documentación del proyecto (si la hay).

**Procesos**

- Análisis de viabilidad (impacto, soluciones alternativas, implicaciones de seguridad, coste y beneficio de la modificación...)
- Análisis detallado (SRS de la modificación, elementos a modificar, estrategia de pruebas...)

**Control**

- Realizar revisiones técnicas y revisar
 - Estrategia de pruebas.
 - Que la documentación está completa y actualizada e incluye parámetros de seguridad

**Salidas**

- Informe de viabilidad de las P.C.
- Informe del análisis detallado.
- Requisitos actualizados (y trazables)
- Lista preliminar de modificaciones.
- Plan de pruebas
- Plan de implementación

**Métricas**

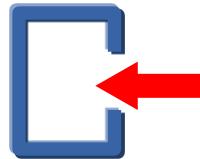
- Modificaciones de requisitos
- % de errores en la documentación
- Esfuerzo por área (SQA, SE, etc.)
- Tiempo empleado
- % de errores generados por prioridad y tipo.

Factores: flexibilidad
trazabilidad
usabilidad
mantenibilidad
reusabilidad

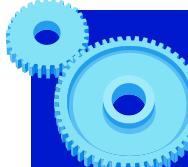
Las 7 fases del mantenimiento

3.- Diseño

En esta fase se emplea toda la documentación del sistema, del proyecto y la generada en la fase anterior (análisis) para diseñar la modificación del sistema.

**Entradas**

- Salidas de la fase de análisis.
- Documentación del sistema y del proyecto
- Código, comentarios y bases de datos del sistema.

**Procesos**

- Identificación de los módulos afectados.
- Documentación de las modificaciones
- Creación de casos de prueba para las modificaciones
- Identificación y creación de pruebas de regresión
- Actualización de documentación (SRS manuales...)

**Control**

- Inspección / verificación del diseño
- Inspección / verificación de la documentación asociada

**Salidas**

- Revisados:
- Lista de modificaciones
- Análisis detallado
- Plan de implementación actualizado
- Línea base de diseño
- Planes de pruebas

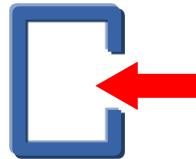
**Métricas**

- Complejidad del software
- Cambios diseño
- Esfuerzo por área
- Cambios en planes de prueba
- Número de módulos
- Nº líneas de código añadidas o modificadas

Las 7 fases del mantenimiento

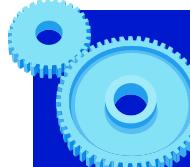
4.- Implementación

A partir del diseño realizado y verificado, el código y la documentación del sistema y del proyecto se lleva a cabo el trabajo de implementación.



Entradas

- Resultados de la fase de diseño.
- Código fuente y bases de datos del sistema.
- Documentación del sistema y del proyecto.



Procesos

- Codificación y pruebas unitarias
- Integración
- Análisis de riesgos
- Revisión de pruebas



Control

- Revisiones de código
- Verificación de la integración.
- Verificación de modificaciones y actualizaciones de documentación.
- Gestión de riesgos y supervisión durante las pruebas



Salidas

- Revisados:
- Software actualizado.
- Documentación de diseño, pruebas, manuales documentación de formación actualizados.
- Definición de riesgos e impactos.
- Informe de revisión de las pruebas



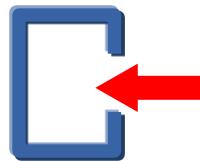
Métricas

- Volumen (puntos de función / líneas de código)
- Porcentaje de errores generados.

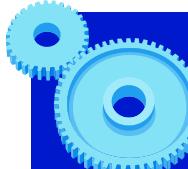
Las 7 fases del mantenimiento

5.- Pruebas de sistema y de regresión

Tras la implementación deben realizarse las pruebas del sistema modificado. Las pruebas de regresión son parte de las pruebas del sistema que comprueban que el código modificado no ha introducido errores nuevos.

**Entradas**

- Informe de las pruebas.
- Documentación de los planes de prueba, casos de prueba, procedimientos de prueba, manuales de usuario, diseño.
- Sistema actualizado

**Procesos**

- Prueba funcional del sistema.
- Pruebas de interfaz.
- Pruebas de regresión.

**Control**

- Las pruebas del sistema se han realizado según los planes SQA.
- Control de la gestión de la configuración de: código, peticiones de cambio, documentación de pruebas

**Salidas**

- Revisados:
- Sistema revisado.
 - Informes de pruebas.

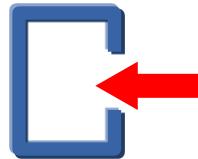
**Métricas**

- Porcentajes de errores por prioridad y tipo: Generados y corregidos.

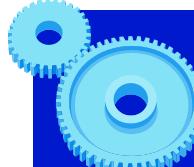
Las 7 fases del mantenimiento

6.- Pruebas de aceptación

Sobre el sistema completamente integrado, el cliente, los usuarios o un tercero nombrado por el cliente lleva a cabo las pruebas de aceptación

**Entradas**

- Informes de pruebas.
- Sistema completamente integrado.
- Planes de pruebas de aceptación.
- Casos de prueba de aceptación.
- Procedimientos de aceptación

**Procesos**

- Ejecución de las pruebas de aceptación a nivel funcional.
- Ejecución de pruebas de interoperabilidad.
- Ejecución de pruebas de regresión.

**Control**

- Ejecución de planes de aceptación.
- Auditoría funcional.
- Puesta bajo control de configuración de la nueva documentación.
- Establecimiento de la nueva línea base del sistema.
- Informe de los resultados de auditoría funcional.

**Salidas**

- Nueva línea base del sistema.
- Informe de auditoría funcional.
- Informe de pruebas de aceptación.

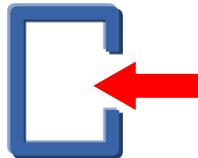
**Métricas**

- Porcentajes de errores por prioridad y tipo: Generados y corregidos.

Las 7 fases del mantenimiento

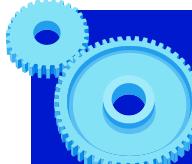
7.- Entrega

Entrega del sistema modificado.



Entradas

- El sistema modificado según se represente en la nueva línea base.



Procesos

- Auditoría física de la configuración.
- Notificación a la comunidad de usuarios.
- Desarrollo y archivo de una copia de seguridad del nuevo sistema.
- Instalación y formación de usuarios.



Control

- Ejecución de la auditoría física de la configuración.
- Documento de descripción de la versión.



Salidas

- Informe de la auditoría física.
- Documento de descripción de la versión.



Métricas

- Cambios en la documentación (manuales de usuario, de operación, documento descripción de versión, etc.)

Mantenibilidad

Con este término, que aunque inexistente en el léxico español se ha hecho hueco en nuestra jerga, se define una propiedad del software que se puede definir como:

la medida cualitativa de la facilidad para comprender, corregir y adaptar o mejorar el software.

Los factores que conforman la mantenibilidad de un sistema de software son:

- Mayor o menor profesionalidad en las fases de diseño, codificación y prueba.
- Adecuada cualificación del equipo desarrollador del software.
- Facilidad de comprensión de la estructura del software.
- Facilidad de uso del sistema.
- Uso de lenguajes de programación y sistemas operativos estandarizados.
- Grado de normalización de la documentación.
- Disponibilidad de la documentación de los casos de prueba.
- Facilidades de depuración con las que cuenta el sistema.
- Disponibilidad de medios e infraestructura para realizar el mantenimiento.
- Madurez en la planificación del mantenimiento.

Mantenibilidad

Medición

Vista la definición de mantenibilidad, y los factores que la forman...

¿Cómo se mide la mantenibilidad?. ¿Es posible afirmar que este sistema tiene una mantenibilidad de 6, o alta, o peor que la de aquel otro sistema?.

No es un atributo ni físico ni simple. No puede medirse directamente.

Las mediciones siempre tendrán carácter de aproximación, y se pueden realizar indirectamente midiendo aspectos relacionados:

- Tiempos invertidos en las tareas de mantenimiento
Para indentificar el problema, para analizarlo, para modificar x líneas de código, etc.
- Midiendo la complejidad del sistema de software.
En esta línea las propuestas de medición apuntan a la medición de la complejidad ciclotómica, la legibilidad, etc.
Esta línea presenta el problema de utilizar atributos indirectos que también son de difícil medición.

La mantenibilidad es un atributo de calidad del software.

Mantenibilidad

Reingeniería

¿Cómo abordar el mantenimiento de un sistema de software con problemas de mantenibilidad?



- No se dispone de documentación (diseño, requisitos...)
- Con deficiente gestión de la configuración.
- Que ha sufrido múltiples y cambios que han degradado el sistema, o desfasado la documentación.

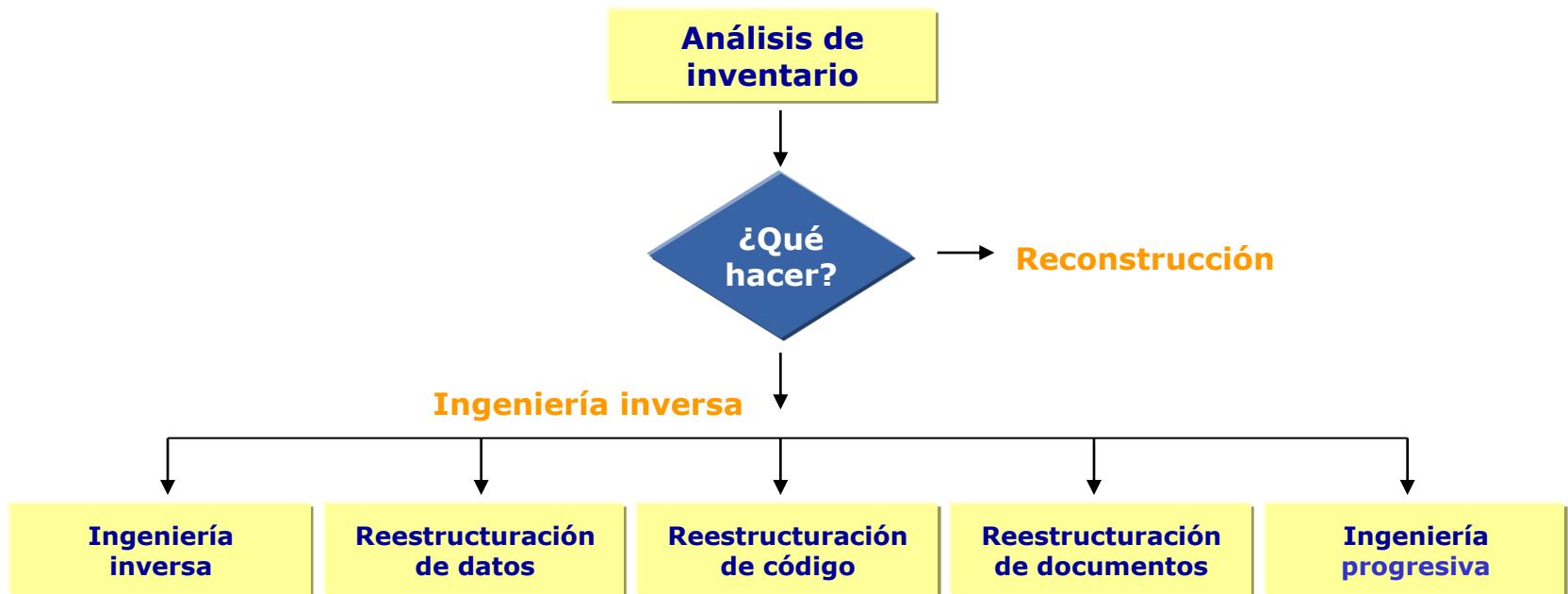


Analizar el sistema y decidir si conviene rehacerlo de nuevo, o por el contrario resulta más apropiado aplicar técnicas de reingeniería.

Mantenibilidad

Modelo de reingeniería del software

El modelo comprende 6 actividades. La primera es un análisis de inventario del que se decidirá si se aplica reingeniería, y en caso afirmativo se emplearán alguna o todas de las cinco actividades restantes.



Mantenibilidad

Modelo de reingeniería del software

Análisis de inventario

El inventario del sistema comprende la información necesaria para el análisis que servirá para decidir si resulta más conveniente rehacer de nuevo el sistema, o aplicar técnicas de reingeniería:

- Identificación del sistema de software
- Año de creación
- Número de cambios importantes realizados
- Esfuerzo invertido en esos cambios
- Fecha y esfuerzo del último cambio importante
- Sistema o sistemas en los que se integra el software
- Sistemas con los que se relaciona
- Bases de datos a las que accede
- Errores detectados en los últimos x meses (12)
- Número de usuarios
- Complejidad de la arquitectura, código y documentación
- Calidad de la documentación
- Mantenibilidad general
- Longevidad acumulada y previsible del proyecto
- Número de cambios previstos en los próximos x meses
- Coste anual del mantenimiento
- Valor actual del negocio que gestiona
- Importancia estratégica para el negocio del cliente y del desarrollador

Mantenibilidad

Modelo de reingeniería del software

Reestructuración de documentos

Los sistemas en los que se cuestiona aplicar reingeniería suelen tener deficiencias en su documentación.

En función de las características del proyecto, tras el análisis del inventario las opciones son:

- Dejarlo como está

Razones:

- Se trata de un sistema con escasa previsión de cambios futuros.
- Se trata de un sistema que se encuentra cercano al fin de su ciclo de vida.
- Los recursos necesarios para crear la documentación no compensan con el beneficio obtenido.

- Documentar sólo las partes que se modifican

Razones:

- Se dispone de recursos limitados.
- Tras el análisis de inventario resulta necesario actualizar la documentación.

- Reducir la documentación al mínimo imprescindible

Razones:

- Se trata de un sistema crítico para el negocio.
- Es preciso volver a documentarlo

Mantenibilidad

Modelo de reingeniería del software

Ingeniería inversa

La ingeniería inversa realiza un análisis de un sistema de software para conseguir especificar su documentación; generalmente su diseño.

Obviamente se aplica cuando no se dispone del diseño, o éste está obsoleto.

Un proceso de ingeniería inversa debe ser capaz de:

- Derivar las representaciones de diseño de procedimientos.
- Extraer la estructura de datos.
- Representar el modelo de los flujos de datos y de control.
- Representar el modelo de entidades y relaciones.

Mantenibilidad

Modelo de reingeniería del software

Reestructuración de código

Los sistemas que tras un análisis de inventario quedan como candidatos a una reestructuración de código suelen presentar una arquitectura de programa relativamente sólida, pero presentan módulos individuales que por haber sufrido modificaciones poco ortodoxas, o por las razones que sean presentan un código “sucio” de difícil comprensión, comprobación y mantenibilidad.

Reestructuración de datos

Las deficiencias en las estructuras de datos son una de las principales causas de errores. Es necesario realizar reestructuración (rediseño y posterior migración de la información al nuevo diseño) en las bases de datos que por no tener un diseño normalizado, o sin integridad relacional presentan un riesgo de error cuyo impacto aconseje su reestructuración.

La reestructuración de datos suele implicar también modificaciones de código.

*Enséñame tu código y mantén ocultas tus estructuras de datos, y me seguirás engañando.
Muéstrame tus estructuras de datos y normalmente no necesitaré que me enseñes tu código:
resultará evidente*

Fred Brooks

Mantenibilidad

Modelo de reingeniería del software

Ingeniería progresiva

Por el estado actual de las herramientas CASE se trata de un modelo ideal de proceso, más que de un proceso que se pueda aplicar directamente.

Su objetivo es ejecutar ingeniería inversa y reestructuración de código de forma automática a través de herramientas CASE que analicen el código y generen su diseño, así como su reestructuración.

Gestión de la configuración

1.0



Introducción

El problema

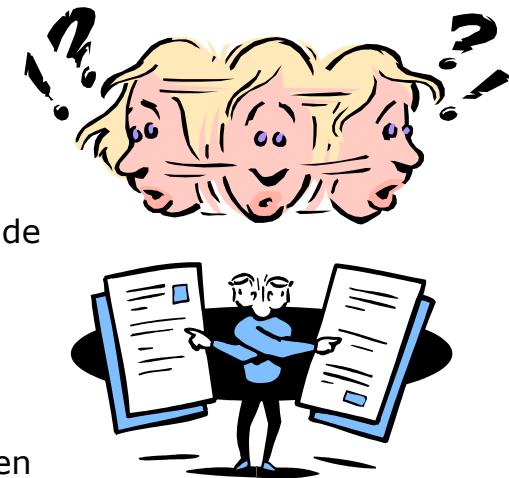
Entorno de desarrollo de un sistema de software de tamaño medio:

- Equipo de 10 programadores.
- 75 módulos de programa.
- Media de dos versiones de cada módulo.
- Documentación del proyecto: descripción del sistema, SRS, plan de proyecto, análisis, etc.
- Cada programador modifica un módulo cada día.
- Modificaciones de requisitos
- Varios programadores deben trabajar de forma concurrente sobre el mismo módulo.
- Etc.



Consecuencias

- La versión del programa no coincide con la documentación.
- Estamos en la versión 2.3, y debemos revisar un error que se ha producido en una instalación de la versión 1.7. ¿Dónde está el código de esa versión?
- Ese error ya se corrigió hace un mes.. ¿Porqué ha vuelto a aparecer?
- ¿Quién aprobó esa modificación de requisitos, y porqué no está en la versión actual de SRS?
- Se está dando mantenimiento a usuarios con diferentes versiones del sistema... ¿Qué versión del componente de acceso a datos se integró en la versión 2.0 del sistema?.
- Etc.



Definición

Gestión de la configuración del software es una disciplina formal de la ingeniería del software que proporciona métodos y herramientas para identificar y controlar el software durante su desarrollo y posterior uso.

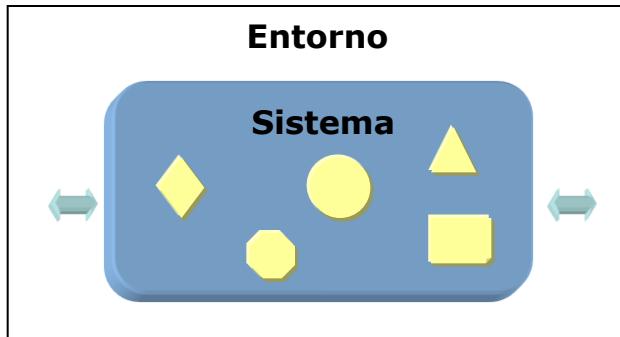
Comprende las siguientes actividades:^[1]

- Identificación y establecimiento de las “líneas base”.
- Revisión, aprobación o rechazo, control y seguimiento de los cambios.
- Auditorías y revisiones de la evolución de los productos de software.
- Control de la relación del sistema de software en su interfaz de operación.

Ámbito de la gestión de la configuración

De aplicación

Los componentes del sistema y su relación con el entorno



Temporal

Desde el inicio del proyecto hasta que se deja de usar y se retira.



[1] En la exposición del capítulo se abordan con detalle cada una de ellas.

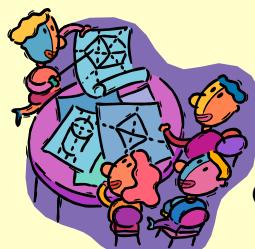
Conceptos clave

Línea base



Petición de cambio

Librería



Comité de control de la configuración

Elementos de configuración del software



Conceptos clave

Línea base



Especificación de un producto que ha sido formalmente **revisada y aceptada** para servir como **punto de referencia** para su posterior desarrollo, y sólo puede modificarse a través de un procedimiento formal de control de cambio.

El número y tipo de líneas base de un proyecto puede ser diferente en función de las características y modelo de ciclo de vida del mismo, pero las habituales son:

- **Línea base funcional**

Describe las funcionalidades que realizará el sistema, y se establece después de la revisión de la descripción del sistema y del diseño del sistema.

- **Línea base de requisitos** (también línea base asignada)

Documenta las funciones que desarrollará el software y se establece después de la revisión de la especificación de requisitos del software (SRS). También se denomina Línea base asignada, porque en ella se asignan al software los requisitos de la descripción del sistema.

- **Línea base de desarrollo**

Esta línea base crece y evoluciona durante el desarrollo del sistema y recoge la configuración en cada fase del diseño, codificación y pruebas. Los elementos contenidos en esta línea base van incrementando y son normalmente revisados por el equipo del desarrollo.

- **Línea base de producto**

Contiene el producto completo en su versión final. Se establece tras comprobar con la validación y verificación del producto que éste es conforme a las especificaciones de los requisitos.

Conceptos clave

Elemento de configuración del software

Un elemento de configuración del software (SWCI) es un conjunto de **productos de trabajo documentados** que se han producido en los procesos del ciclo de vida, o que se emplean en los mismos.

Por producto de trabajo se entiende a un elemento tangible que es el resultado de determinadas actividades o tareas de desarrollo: planes de pruebas, documentos de requisitos, documentos de diseño, código, manuales de usuario, etc.



Los elementos de configuración del software son cualquier parte del desarrollo o del producto entregable y necesitan poder identificarse, almacenarse, cambiarse, revisarse o mantenerse de forma independiente.

Estos elementos no comprende sólo los productos de software que se entregan al cliente, también incluyen los elementos que son necesarios para crear esos productos.

Categorías

Producto: Productos intermedios o finales desarrollados durante el proyecto.

Software adquirido: Módulos o componentes adquiridos o subcontratados.

Software suministrado: Software proporcionado por el cliente para que se integre en el sistema.

Software de pruebas: Software empleado para realizar las pruebas.

Software de apoyo: Software empleado para desarrollar el sistema de software (compiladores, etc.)

Conceptos clave

Petición de cambio

Las peticiones de cambio documentan la necesidad de modificar un elemento de configuración del software.

Las peticiones de cambio deben incluir:

- Razón por la que hay que realizar el cambio (detección de un fallo, modificación de requisitos, etc.)
- Elemento de configuración afectado y línea base a la que pertenece.
- Urgencia del cambio.
- Persona que lo solicita e indicación de si el origen es interno o externo.

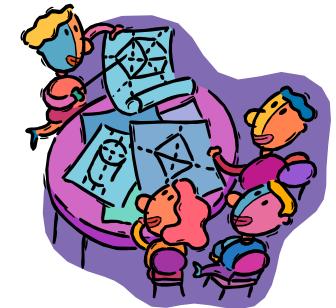


Conceptos clave

Comité de control de la configuración

Para conseguir que las peticiones de cambio se procesen de forma ordenada, correcta y a tiempo, el proyecto debe establecer quién o quienes configuran el comité de control de la configuración.

En función del tamaño y características del proyecto puede ser que lo forme una sola persona (p. ej. el analista o el gestor del proyecto), o que esté formado por varias: gestor de proyecto, cliente, gestor de calidad, etc.



Las funciones del comité incluyen:

- Sopesar las ventajas e inconveniente de la introducción del cambio (beneficios esperados, coste de la implementación)
- Evaluar el impacto de la modificación sobre los parámetros del proyecto (agenda, costes, riesgos, etc.).

El comité no sólo decide si debe realizarse el cambio, también determina su prioridad, cuándo y cómo debe llevarse a cabo y cómo deberá comprobarse y verificar su implementación.

Conceptos clave

Librerías

Las librerías constituyen los dispositivos de almacenamiento necesarios para llevar a cabo los cambios y el control histórico de los mismos que requiere la gestión de la configuración, de forma que queden guardadas y puedan recuperarse las diferentes líneas base en cualquiera de sus versiones.

El número y tipo de librerías puede variar en función de las características del proyecto, pero generalmente son 3:



- **Librería dinámica**

Es el entorno de almacenamiento usado y gestionado por el equipo de programación en el que se ubican los elementos con los que están trabajando.

- **Librería controlada**

Es la librería empleada para guardar las líneas base y controlar los cambios que sobre ellas se realizan. Los elementos se almacenan en esta librería después de haber sido identificados como elementos de configuración asignados a su línea base, documentados y aceptados por el comité de gestión de la configuración.

- **Librería estática**

También llamada repositorio de software. Guarda las líneas base una vez que han sido validadas y verificadas para su distribución y uso final.

Conceptos clave

Librerías

LIBRERÍA DINÁMICA

También llamada “Directorio de programación”.
Controlada por el equipo de programación.



LIBRERÍA CONTROLADA

También llamado “Directorio maestro”.
Contiene todas las líneas base del proyecto.



LIBRERÍA ESTÁTICA

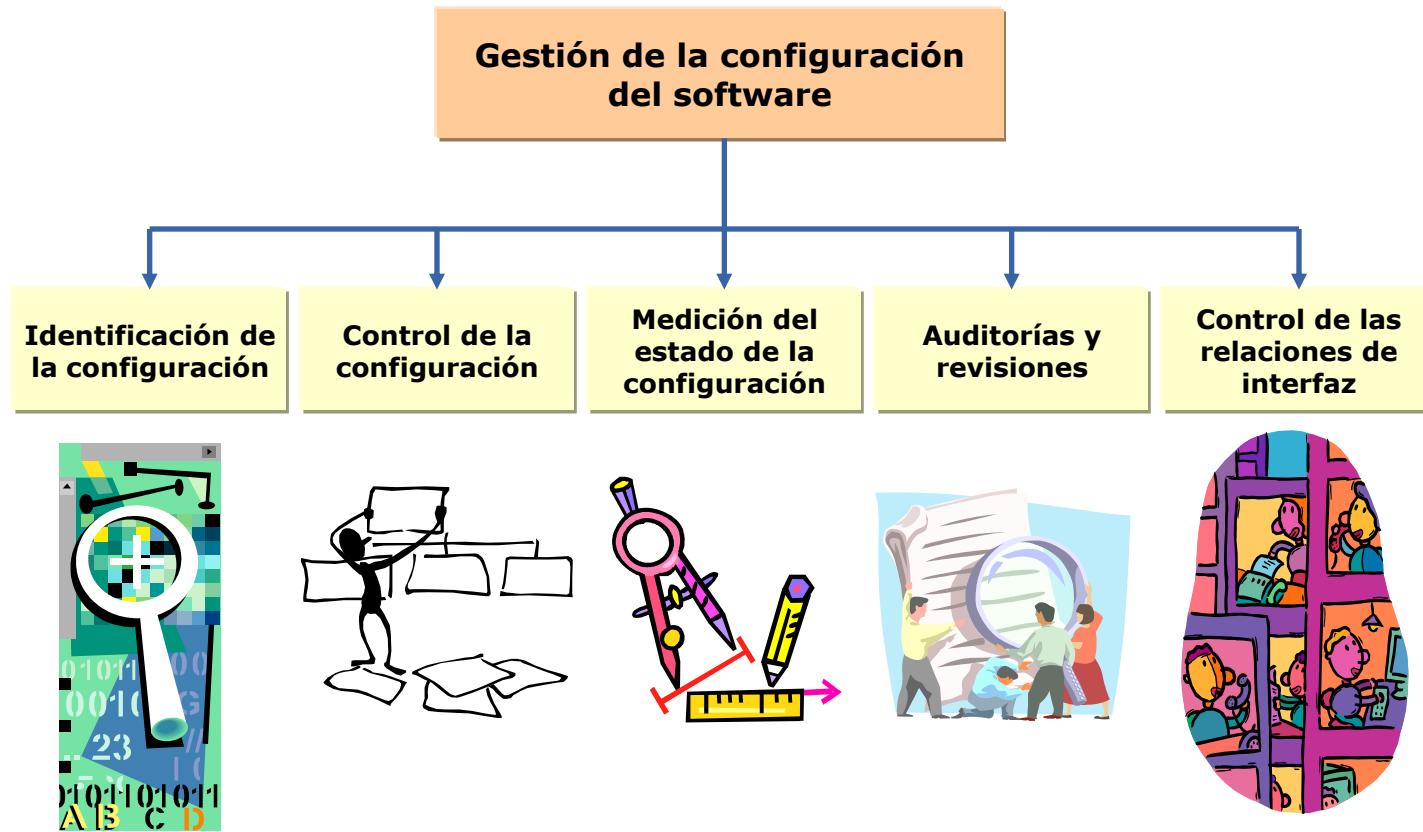
También llamado “Repositorio de software”
Comprende las líneas base que finalmente se entregan.



Versión 1.0

Versión 1.1

Áreas clave



Áreas clave

Identificación de la configuración

Determinación de los elementos de configuración del software y de las líneas base a las que pertenecen.

Selección de los elementos de configuración y agrupación en líneas base. Deben considerarse productos que puedan diseñarse, desarrollarse, probarse y modificarse de forma independiente.



No deben identificarse muy pocos, ni tampoco demasiados. Los criterios de selección deben ser acordes con las características del proyecto.

Actividades

Identificación

Nomenclatura y adquisición



Nomenclatura: Cada elemento de configuración debe nombrarse con un identificador único. Es habitual que el identificador contenga:

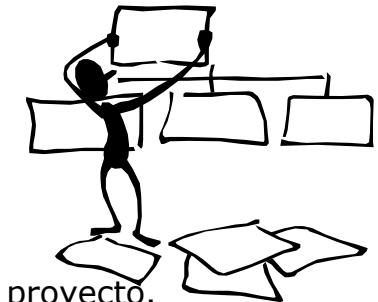
- NOMBRE: descriptivo del elemento.
- IDENTIFICADOR DE CONFIGURACION: Usado en la gestión interna de la librería.
- IDENTIFICADOR DE VERSION: Usado para identificar las diferentes versiones.

Adquisición: Una vez identificado cada elemento, debe incorporarse a su respectiva librería.

Áreas clave

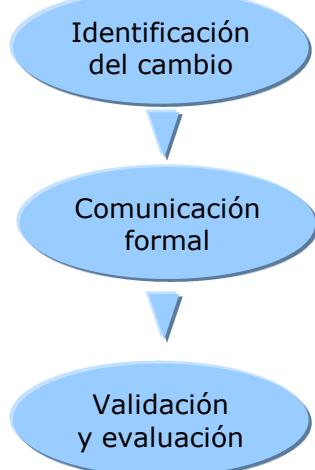
Control de la configuración

Comprende la gestión de las revisiones y de los procesos de aprobación, para evitar que se produzcan cambios de forma descontrolada.



Garantiza

- Que para cada cambio se evalúa y considera el impacto en el proyecto.
- Que sólo se implementan los cambios aprobados.
- Que todos los cambios aprobados se implementan.
- Que las líneas base se mantienen controladas y actualizadas.



CLASIFICACIÓN

- Por urgencia
- Por Naturaleza (error, mejora, mod. Requisitos...)
- Por categoría de elementos modificados (producto, Software adquirido, Software suministrado, software de pruebas o software de apoyo).

EVALUACIÓN

- Técnico
- En los interfaces de configuración
- En la agenda
- En el presupuesto



- Check-out línea base
- Ejecución de cambios
- Pruebas y verificación
- Aprobación de la ejecución
- Chech-in línea base

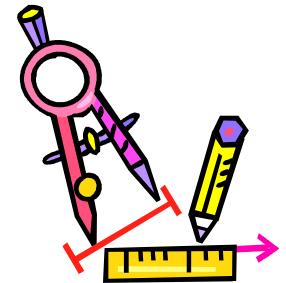
Áreas clave

Medición del estado de la configuración

Medición y registro de los cambios, contenidos e históricos de la gestión de la configuración



- Versión inicial aprobada de los elementos de la configuración.
- Estado de las peticiones de cambio.
- Estado de implantación de los cambios aprobados.



Esta es la información mínima que debería registrarse (Std. IEEE 828-1998).

Auditorías y revisiones

Con menor o mayor rigor, según se trate de revisiones o auditorías, estos procesos también se deben aplicar sobre la gestión de la configuración para garantizar:



- Que los elementos de la configuración se encuentran en el estado que deberían estar.
- Que las actividades, las tareas y los resultados de la gestión de la configuración son correctos.

Áreas clave

Control de las relaciones de interfaz

El desarrollo y mantenimiento de sistemas de software no suele ser “auto-contenido”. Normalmente el software debe relacionarse con hardware y con otro software. El control de las relaciones de Interfaz contempla y gestiona las situaciones posibles:



SITUACIONES

- El software debe ejecutarse sobre plataformas operativas comerciales
- El producto de software debe integrar componentes externos
- El desarrollo de partes del software se subcontrata a un proveedor externo.
- Evolución paralela del hardware del sistema global

IMPLICACIONES DE INTERFAZ

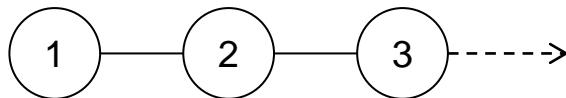
La gestión de la configuración debe registrar también las plataformas y componentes externos, evaluando las posibles evoluciones y cambios.

Las gestiones de configuración del proyecto de software y del subcontratado deben comunicarse y gestionar las implicaciones de cambio derivadas de uno a otro.

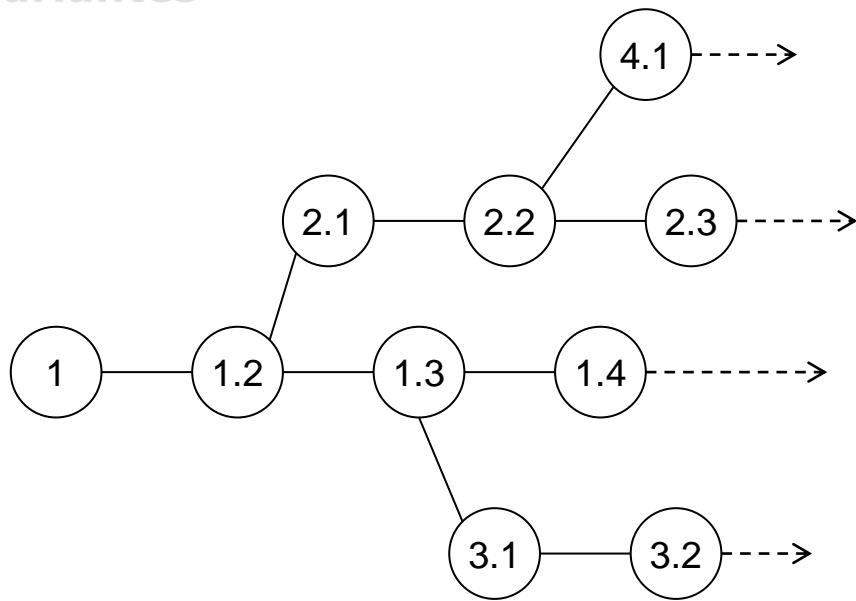
La gestión de la configuración del sistema global debe relacionarse con la del proyecto de software por las implicaciones de cambios que pueden derivarse en ésta de aquella.

Control de versiones

Evolución simple



Variantes



Tronco: 1.1 – 1.2 – 1.3 ...

Cabeza: Última versión del tronco

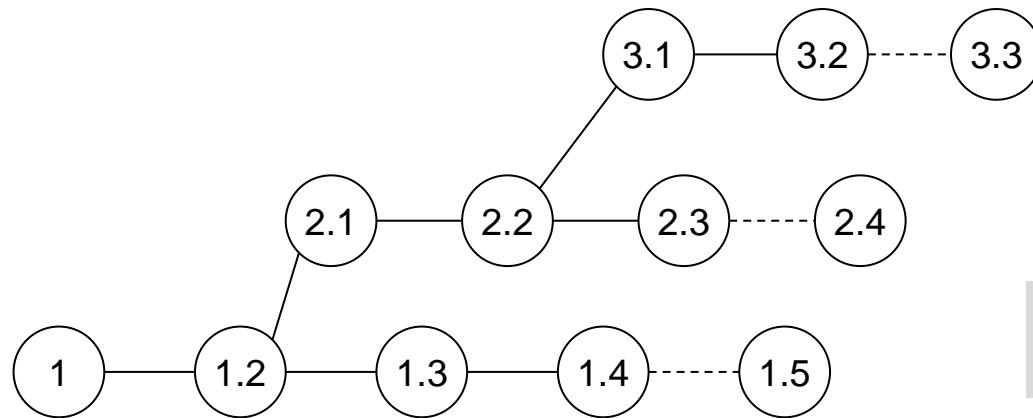
Ramas: 2.1... 3.1..... 4.1....

Delta: Cambio de una versión respecto a su anterior: 3.2<- 3.1 / 2.3 <- 2.2

Control de versiones

Propagación de cambios

Cambio aplicable a varias ramas



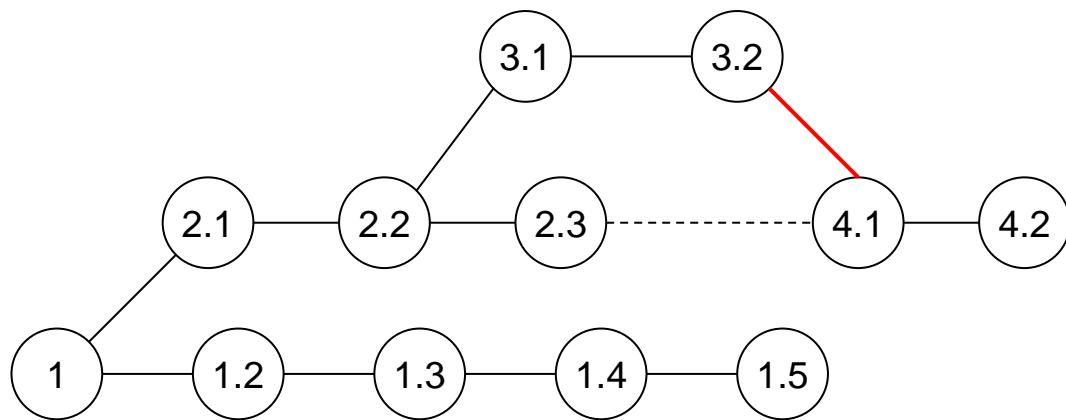
$$2.4 = 2.3 + (1.5 - 1.4)$$

$$3.3 = 3.2 + (1.5 - 1.4)$$

La herramientas de control de versiones suelen incorporar la funcionalidad “Diff-Merge” para gestionar la propagación de cambios.

Control de versiones

Fusión de ramas



$$4.1 = 2.3 + (3.2 - 2.2)$$

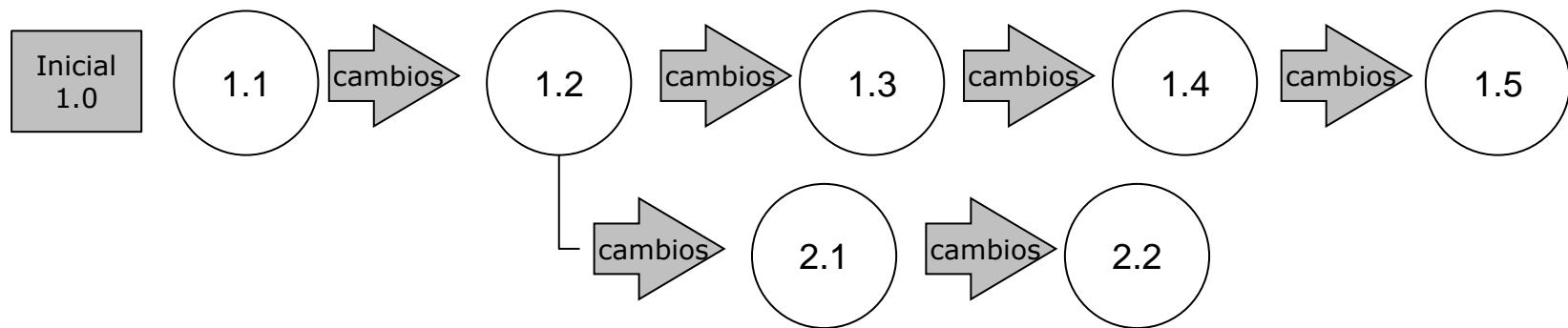
$$4.1 = 3.2 + (2.3 - 2.2)$$

La herramientas de control de versiones suelen incorporar la funcionalidad “Two-way merge” para gestionar la fusión de dos ramas.

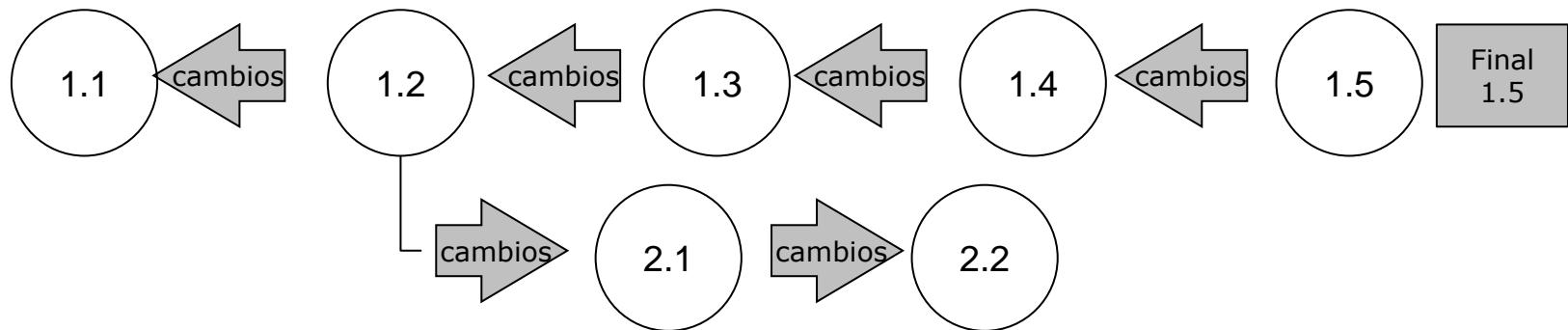
Control de versiones

Técnicas de almacenamiento

DELTAS DIRECTOS



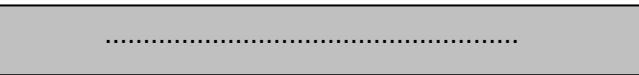
DELTAS INVERSOS



Control de versiones

Técnicas de almacenamiento

MARCADO SELECTIVO

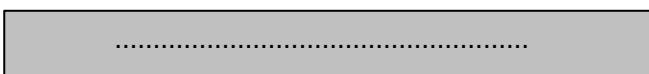


<< 1.3, 1.2



>>

<< 1.3



>>

.....

La herramientas de control de versiones suelen incorporar funcionalidades diff o diff-merge para recuperar el contenido de una versión, empleando una de estas técnicas.

AGILIDAD

Entrega temprana

Incremento continuo



cc-by [fox_kiyo](#)

AGILIDAD: entrega temprana, incremento continuo

17.02.03 Buscar Mi esPublico Carrito Portapapeles Correo Ayuda

La Gestión del Conocimiento para la Administración Pública

Gestión
Expedientes
Subvenciones

El Asesor
Consultas resueltas
Preguntar
Ver mis consultas

Profesional
Legislación
Jurisprudencia

Formación
Aula Virtual

Actualidad
Revista
Noticias
Entrevistas

Información
Admón. Pública

Comunidad
Experiencias
Foros
Links
Encuestas

Tienda Virtual
Tienda Virtual

noticias
13/02/2003 - COMUNIDAD VALENCIANA: LAS CORTES APRUEBAN LA NUEVA LEY DE ESPECTÁCULOS

formación
13/02/2003 - CASTILLA-LA MANCHA: LAS CORTES REGIONALES APRUEBAN LA LEY DE COOPERACIÓN INTERNAZIONAL PARA EL DESARROLLO

entrevistas
13/02/2003 - APROVADA LA LLEI D'UNIVERSITATS DE CATALUNYA (L.U.C.)

revista
- La Ordenación Jurídica de las Políticas Locales en España: Una Visión de Conjunto
- Sentido y Justificación Actual de la Concesión de Servicios Municipales

profesional LEGISLACIÓN
Búsqueda práctica Búsqueda clásica
Palabra clave: Administración: Cualquier administración
Resultados por página: 10

Usuario:
Contraseña:
«La olvidé» ENVIAR

SUSCRIPCIONES
Elige una de nuestras modalidades de suscripción
Registro gratuita Suscripciones

Mis últimas búsquedas Alertas personalizadas

27/1/2003 - LEGISLACIÓN: Ley 52/2002, de 30 de diciembre, de Presupuestos Generales del Estado para el año 2003.
22/1/2003 - LEGISLACIÓN: Ley 35/2002, de 20 de diciembre, de Ordenación del Territorio y Urbanismo.
21/1/2003 - LEGISLACIÓN: Ley 4/2002, de 17 de diciembre, de Ordenación Urbanística de Andalucía.

novedades
27/1/2003 - LEGISLACIÓN: Ley 52/2002, de 30 de diciembre, de Presupuestos Generales del Estado para el año 2003.
22/1/2003 - LEGISLACIÓN: Ley 35/2002, de 20 de diciembre, de Ordenación del Territorio y Urbanismo.
21/1/2003 - LEGISLACIÓN: Ley 4/2002, de 17 de diciembre, de Ordenación Urbanística de Andalucía.

27/1/2003 - LEGISLACIÓN: Ley 52/2002, de 30 de diciembre, de Presupuestos Generales del Estado para el año 2003.
22/1/2003 - LEGISLACIÓN: Ley 35/2002, de 20 de diciembre, de Ordenación del Territorio y Urbanismo.
21/1/2003 - LEGISLACIÓN: Ley 4/2002, de 17 de diciembre, de Ordenación Urbanística de Andalucía.

noticias
13/02/2003 - COMUNIDAD VALENCIANA: LAS CORTES APRUEBAN LA NUEVA LEY DE ESPECTÁCULOS

formación
13/02/2003 - CASTILLA-LA MANCHA: LAS CORTES REGIONALES APRUEBAN LA LEY DE COOPERACIÓN INTERNAZIONAL PARA EL DESARROLLO

entrevistas
13/02/2003 - APROVADA LA LLEI D'UNIVERSITATS DE CATALUNYA (L.U.C.)

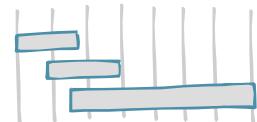
revista
- La Ordenación Jurídica de las Políticas Locales en España: Una Visión de Conjunto
- Sentido y Justificación Actual de la Concesión de Servicios Municipales

Inicio. Suscripciones. Quiénes somos. Colaboradores. Contáctenos con nosotros. Copyright © 2003 Aulice S.A. Todos los derechos reservados. Aviso Legal. Política de Privacidad.

espublico.com



2000



PROTEGEMOS TU CREATIVIDAD BIENVENIDO | CONECTARSE | DARSE DE ALTA

Qué es Safe Creative | Un paseo por Safe Creative | Darse de Alta

toma nota e! AYUDA ON CLICK ¿Necesita ayuda? HAGA CLICK

EL AUTOR ES QUIEN DETERMINA EL MODELO DE GESTIÓN de sus OBRAS

folios obras plásticas multimedia software

registro creativo como registrar tus obras de una forma fácil

Buscar en safecreative Lo que necesitas saber para registrar tus obras

Mi cuenta Tu nombre: Entrar Tu password:

Ayuda Lo que necesitas saber para entender como utilizar safecreative

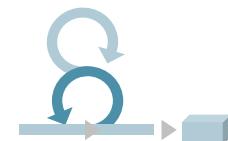
Privacy Policy | Terms of Use | Abusos de copyright | Cláusula de responsabilidades

safecreative® Copyright © 2007 AAR Future Corp.

safecreative.org



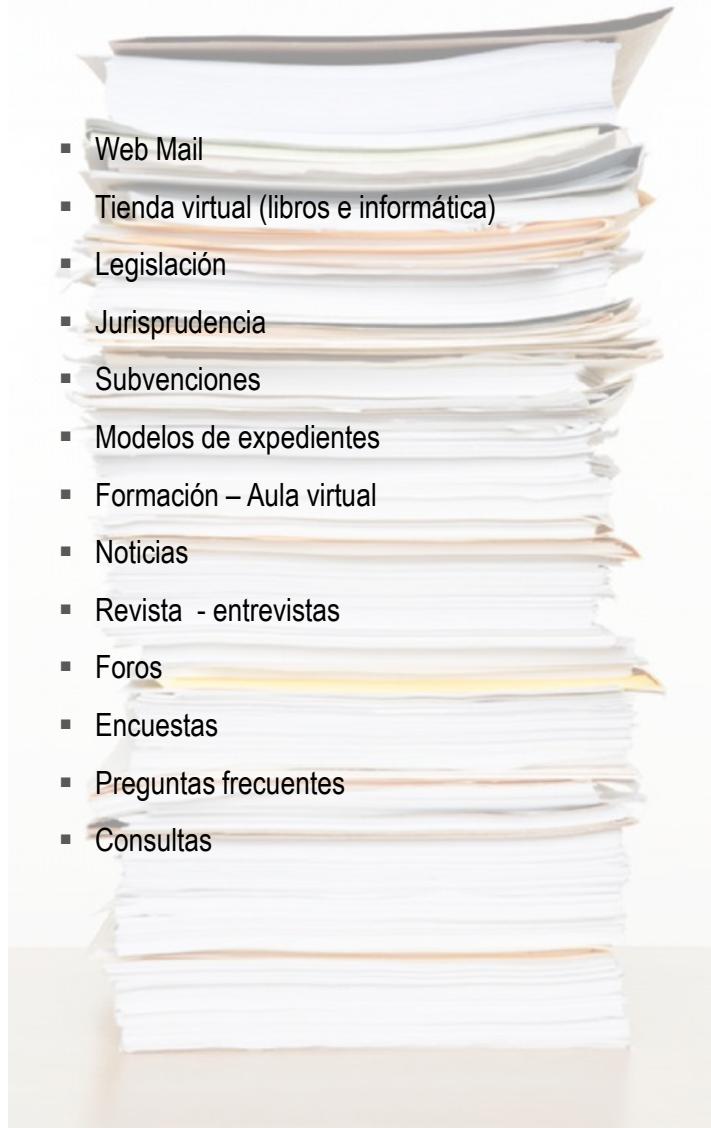
2007



AGILIDAD: entrega temprana, incremento continuo

The screenshot shows the homepage of espublico.com. On the left, there's a sidebar with links for Gestion, Asesor, Professional, Formacion, Actualidad, and Tienda Virtual. The main content area has tabs for profesional, LEGISLACIÓN, noticias, formación, and revista. Under profesional, there's a search bar and a list of news items. One item is highlighted: "13/02/2003 - COMUNIDAD VALENCIANA APRUEBAN LA NUEVA LEY DE ESPECTACULOS". Other news items include "13/02/2003 - CASTILLA-LA MANCHA: LAS CORTEZ REGIONALES APRUEBAN LA LEY DE COOPERACION INTERAUTONOMA PARA EL DESARROLLO" and "13/02/2003 - APROVADA LA LLEI D'UNIVERSITATS DE CATALUNYA (LUOC)". The formación tab shows an interview with Adrós Andreu López. The revista tab has an article about the electronic administration. At the bottom, there's a footer with links to Inicio, Suscripciones, Quiénes somos, Colaboradores, Contacte con nosotros, Copyright © 2001, Aulocis S.A., Todos los derechos reservados, Aviso Legal, and Política de Privacidad.

- Web Mail
- Tienda virtual (libros e informática)
- Legislación
- Jurisprudencia
- Subvenciones
- Modelos de expedientes
- Formación – Aula virtual
- Noticias
- Revista - entrevistas
- Foros
- Encuestas
- Preguntas frecuentes
- Consultas



AGILIDAD: entrega temprana, incremento continuo

The screenshot shows the espublico.com website's search interface for legislation. The search bar contains the word 'legislación'. The results page displays several news items related to legislation, such as:

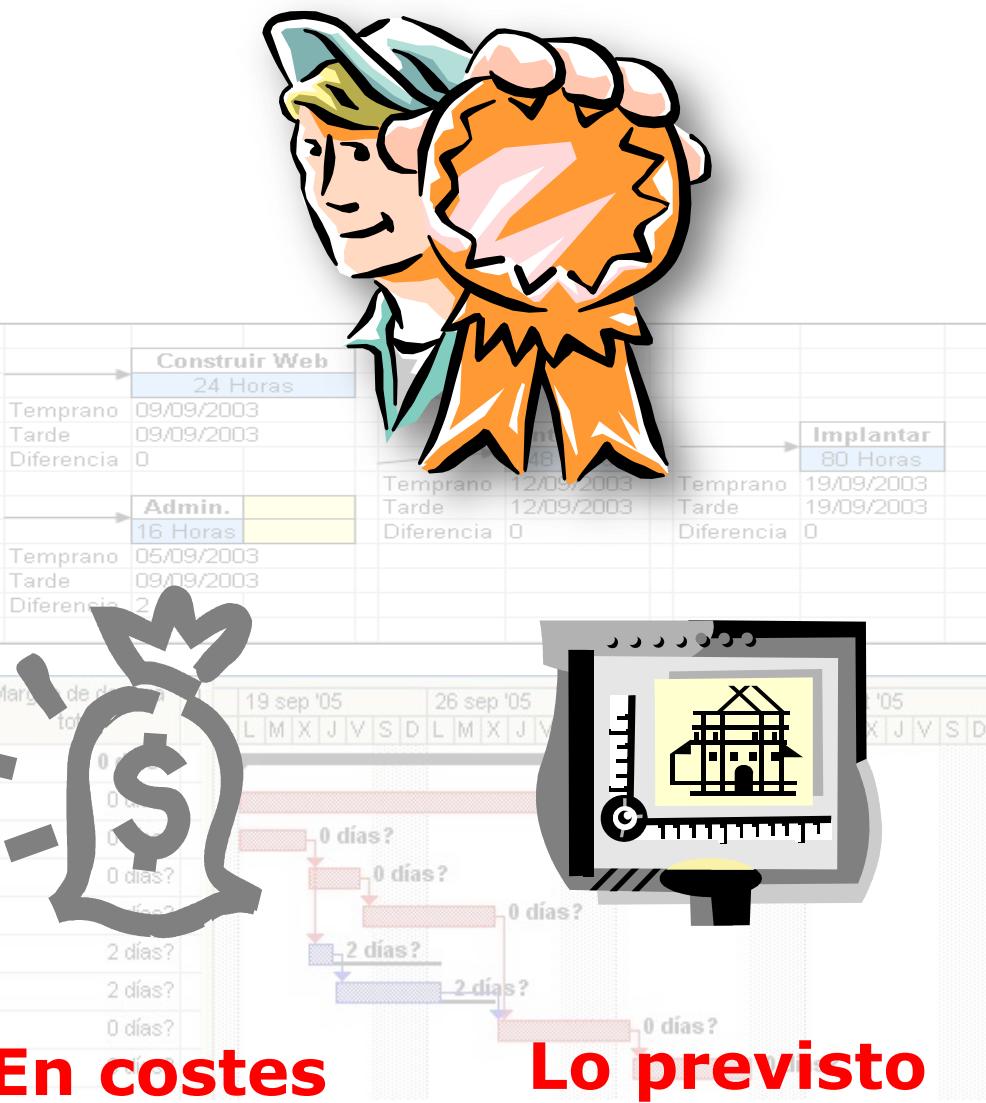
- 27/1/2003 - LEGISLACIÓN: Ley 52/2002, de 30 de diciembre, de Presupuestos Generales del Estado para el año 2003.
- 22/1/2003 - LEGISLACIÓN: Ley Foral 35/2002, de 20 de diciembre, de Ordenación del Territorio y Urbanismo.
- 21/1/2003 - LEGISLACIÓN: Ley 4/2002, de 17 de diciembre, de Ordenación Urbánica de Andalucía.

	Nombre de tarea	Fin anticipado	Límite de finalización	Margen de tiempo total
1	Expedientes	vie 07/10/05	vie 07/10/05	0 días?
2	Gestión de Proy.	vie 07/10/05	vie 07/10/05	0 días?
3	Análisis	mié 21/09/05	mié 21/09/05	0 días?
4	Diseño web	vie 23/09/05	vie 23/09/05	0 días?
5	Construir web	mié 28/09/05	mié 28/09/05	0 días?
6	Diseño admin	jue 22/09/05	lun 26/09/05	2 días?
7	Construir admin	lun 26/09/05	mié 28/09/05	2 días?
8	Integrar	lun 03/10/05	lun 03/10/05	0 días?
9	Implantar	07/10/05	vie 07/10/05	0 días?

A tiempo

En costes

Lo previsto

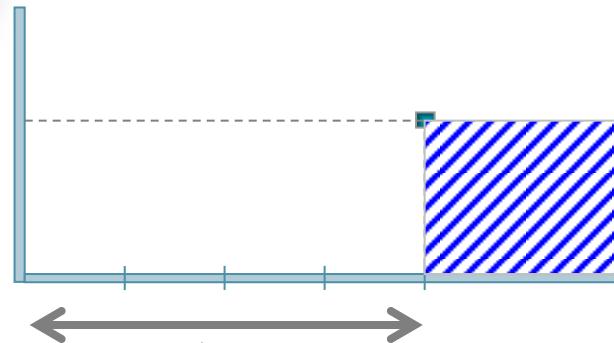


AGILIDAD: entrega temprana, incremento continuo

The screenshot shows the homepage of espublico.com. At the top, there's a navigation bar with links like '17.02.03', 'Documentos', 'Gestión del Conocimiento', 'Correo', and 'Ayuda'. Below the navigation is a banner for 'La Gestión del Conocimiento para la Administración Pública'. The main content area has a sidebar on the left with categories such as 'Gestión' (Expedientes, Subvenciones), 'El Asesor' (Consultas resueltas, Preguntar, Ver más consultas), 'Profesional' (Legislación, Jurisprudencia), 'Formación' (Aula Virtual), 'Actualidad' (Revista, Noticias, Entrevistas), 'Información' (Administración Pública), 'Comunidad' (Experiencias, Foros, Links, Encuestas), and 'Tienda Virtual' (Tienda Virtual). The main content area features a search form for 'LEGISLACIÓN' with fields for 'Palabra clave' and 'Administración'. It also includes sections for 'noticias' (with news items like '13/02/2003 - COMUNIDAD VALENCIANA APROUEAN LA NUEVA LEY DE ESPECTÁCULOS'), 'formación' (with an item like '13/02/2003 - CASTILLA-LA MANCHA: LAS CORTEZ REGIONALES APRUEBAN LA LEY DE EDUCACIÓN INTERNAZIONAL PARA EL DESARROLLO'), 'entrevistas' (with an item like '13/02/2003 - APROVADA LA LLEI D'ESTATUTS DE CATALUNYA (L-U-C)'), and 'revista' (with an item like 'Le Conferencia Jurídica de las Políticas Locales en España: Una Visión de Conjunto'). There's also a sidebar for 'SUSCRIPCIONES' and 'AYUDA & CLICK'.



Time to market



Desperdicio: 80%



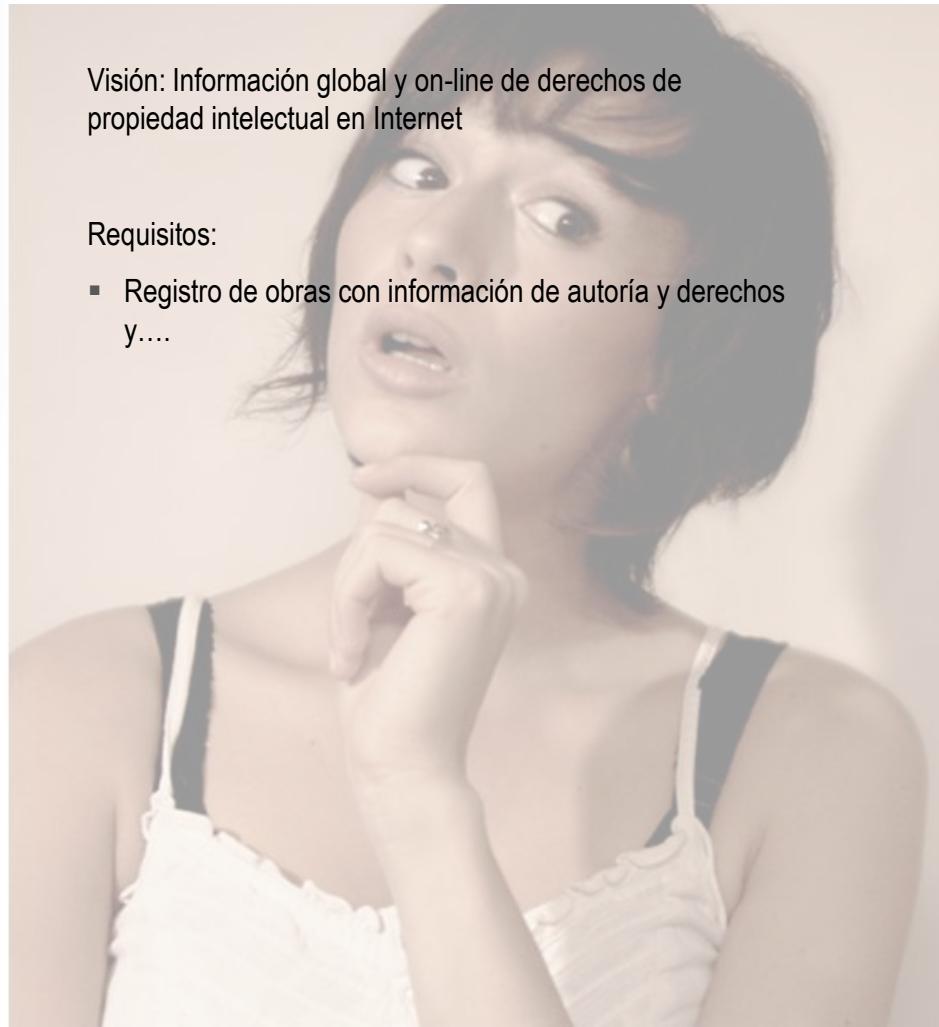
AGILIDAD: entrega temprana, incremento continuo



Visión: Información global y on-line de derechos de propiedad intelectual en Internet

Requisitos:

- Registro de obras con información de autoría y derechos y....



Fotografía cc-by: [Frédéric Dupont](#)

AGILIDAD: entrega temprana, incremento continuo

INCERTIDUMBRE



Fotografía cc-by: [Frédéric Dupont](#)

AGILIDAD: entrega temprana, incremento continuo

VELOCIDAD



Fotografía cc-by:[Amnemona](#)

AGILIDAD: entrega temprana, incremento continuo



2.007



2.008



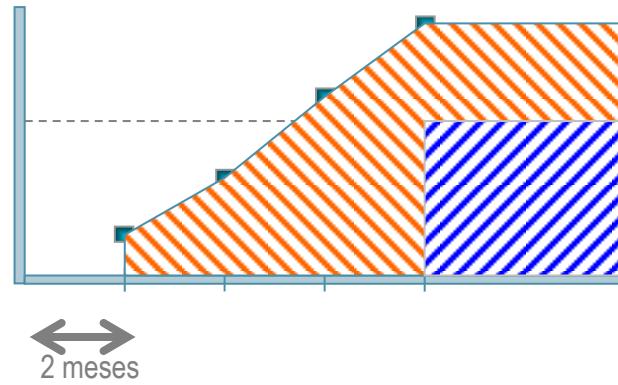
2.009



2.011



Time to market



Desperdicio: 20%

AGILIDAD: entrega temprana, incremento continuo



- Web Mail
- Tienda virtual (libros e informática)
- Legislación
- Jurisprudencia
- Subvenciones
- Modelos de expedientes
- Formación – Aula virtual
- Noticias
- Revista – entrevistas
- Foros
- Encuestas
- Preguntas frecuentes
- Consultas



- Certificados y notas informativas
- Buscador básico
- Registro desde RSS.
- Multiautoría
- Buscador avanzado
- Notificaciones cesae & desist.
- Interfaz API
- Programa cliente de registro (ART)
- Plataforma de licenciamiento
- Monitor de ventas y liquidaciones
- Información de consultas y descargas
- Asistente para generación de contratos
- Licencias personalizadas
- Licencias temporales
- API y programa de registro remoto.
- Asesoría jurídica
- Registro en U.S. Copyright Office
- Directorio de contratos y consultas
- Etc.

b.



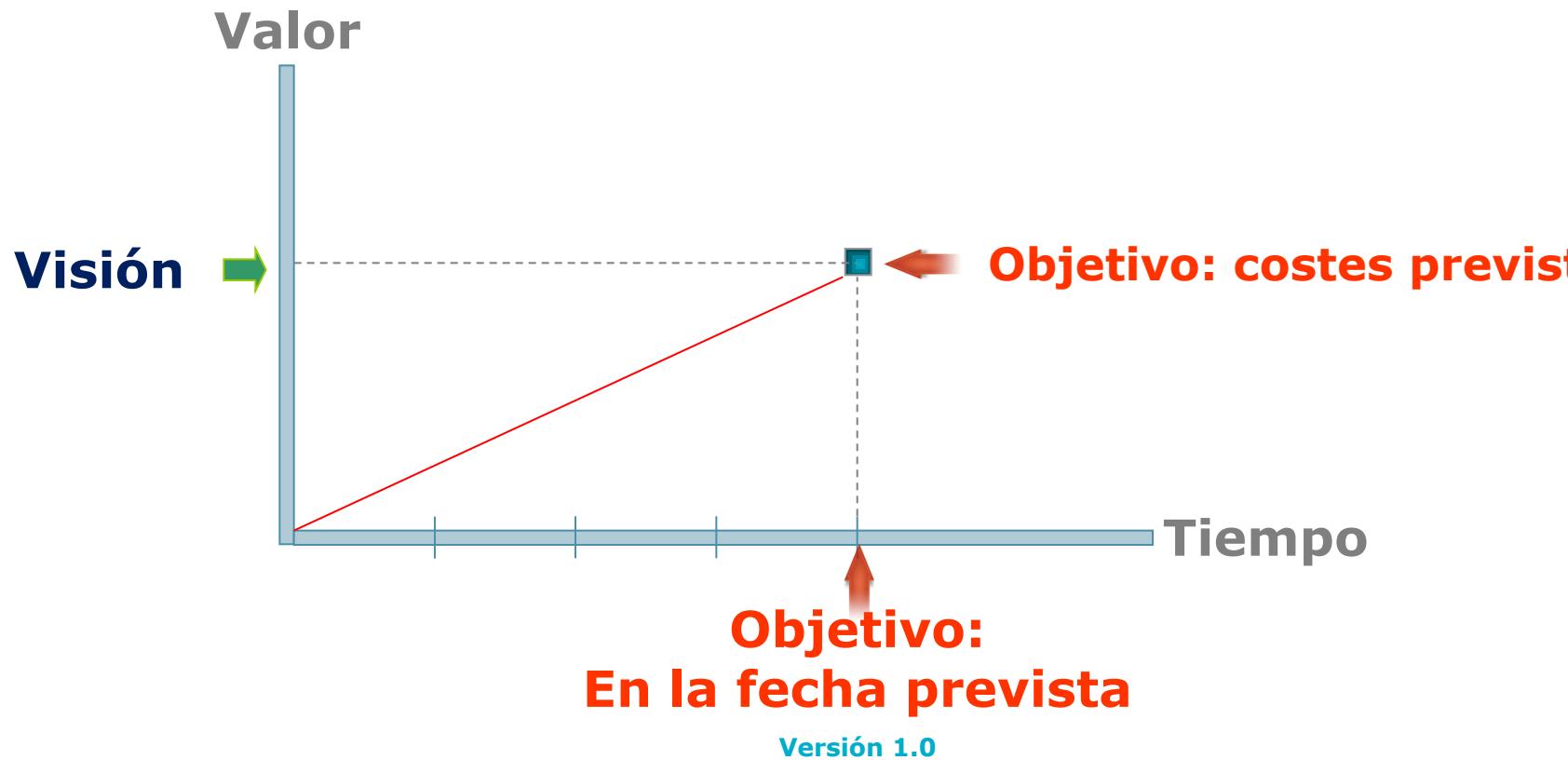
La gestión predictiva tiene como objetivo construir el producto planificado en el tiempo y con los costes previstos

	Nombre de tarea	Fin anticipado	Límite de finalización	Margen de demora total	Pri	19 sep '05	26 sep '05	03 oct '05	10 oct '05												
						D	L	M	X	J	V	S	D	D	L	M	X	J	V	S	D
1	Proyecto-CIS	vie 07/10/05	vie 07/10/05	0 días?																	
2	Gestión de Proyecto	vie 07/10/05	vie 07/10/05	0 días?																	
3	Ánalisis	mié 21/09/05	mié 21/09/05	0 días?																	
4	Diseño web	vie 23/09/05	vie 23/09/05	0 días?																	
5	Construir web	mié 28/09/05	mié 28/09/05	0 días?																	
6	Diseño admin	jue 22/09/05	lun 26/09/05	2 días?																	
7	Construir admin	lun 26/09/05	mié 28/09/05	2 días?																	
8	Integrar	lun 03/10/05	lun 03/10/05	0 días?																	
9	Implantar	vie 07/10/05	vie 07/10/05	0 días?																	

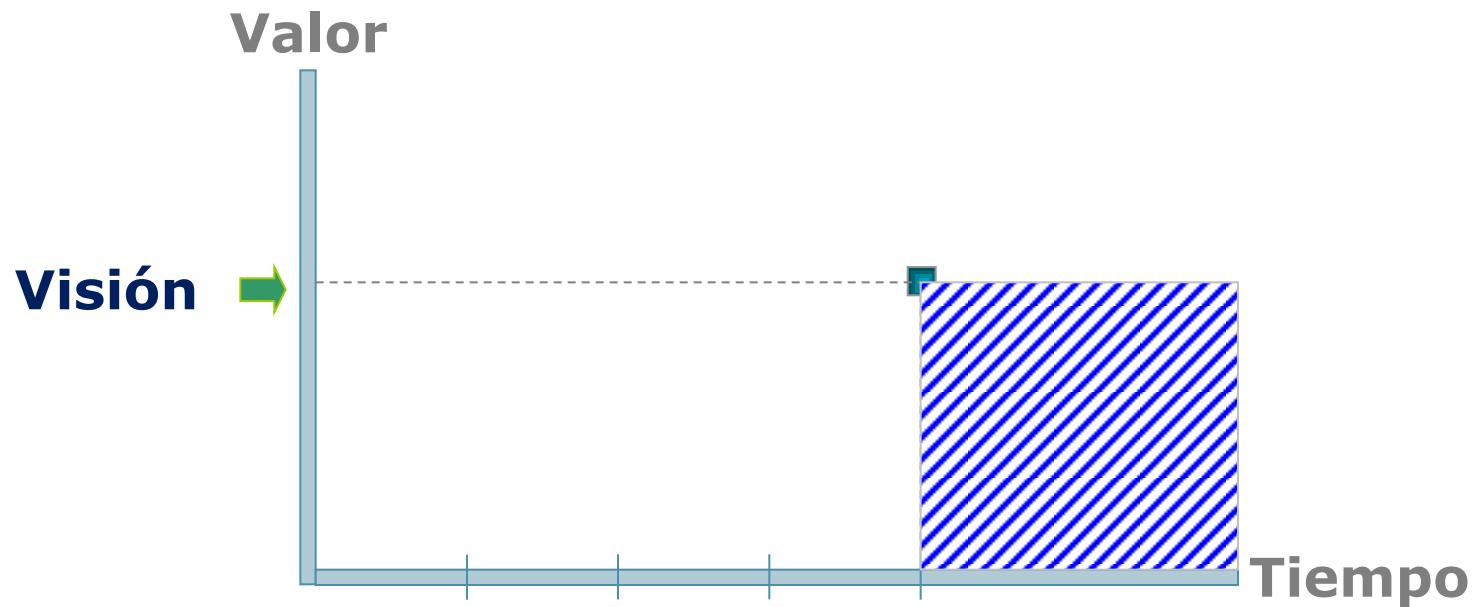
La gestión ágil tiene como objetivo entregar el mayor valor posible en el menor tiempo, e incrementarlo de forma constante



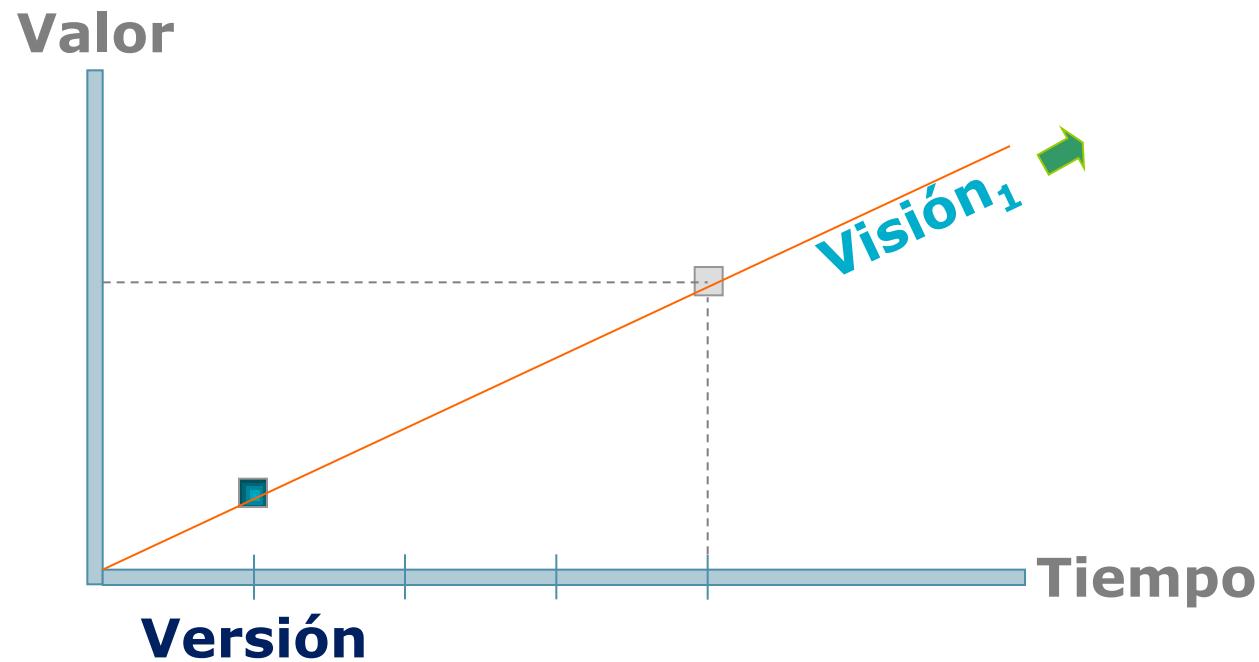
GESTIÓN PREDICTIVA: entrega de valor al cliente



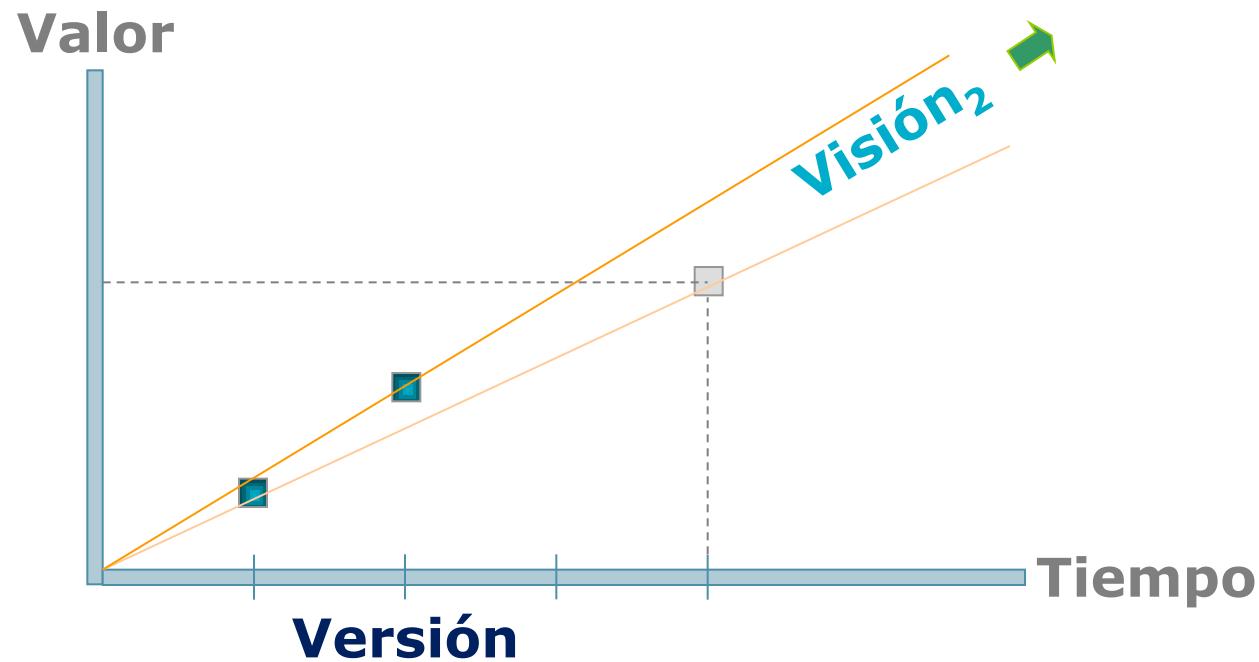
GESTIÓN PREDICTIVA: entrega de valor al cliente



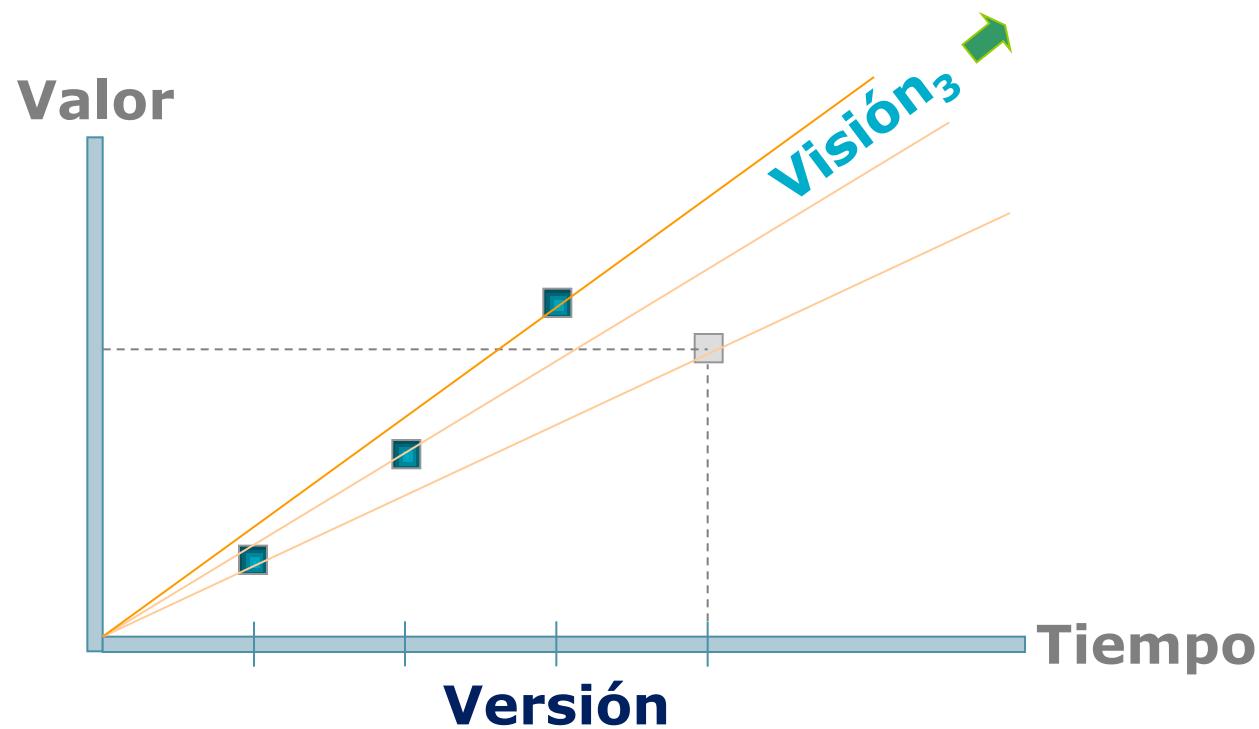
GESTIÓN ÁGIL: entrega de valor al cliente



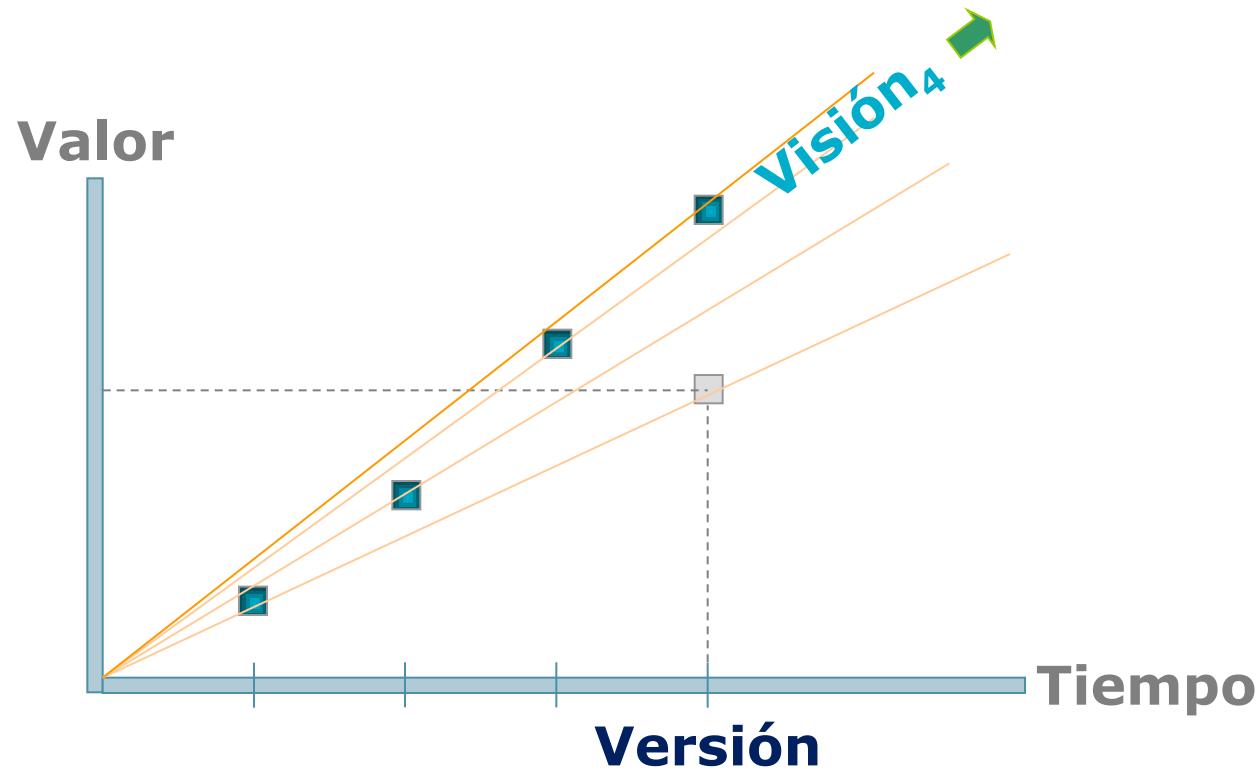
GESTIÓN ÁGIL: entrega de valor al cliente



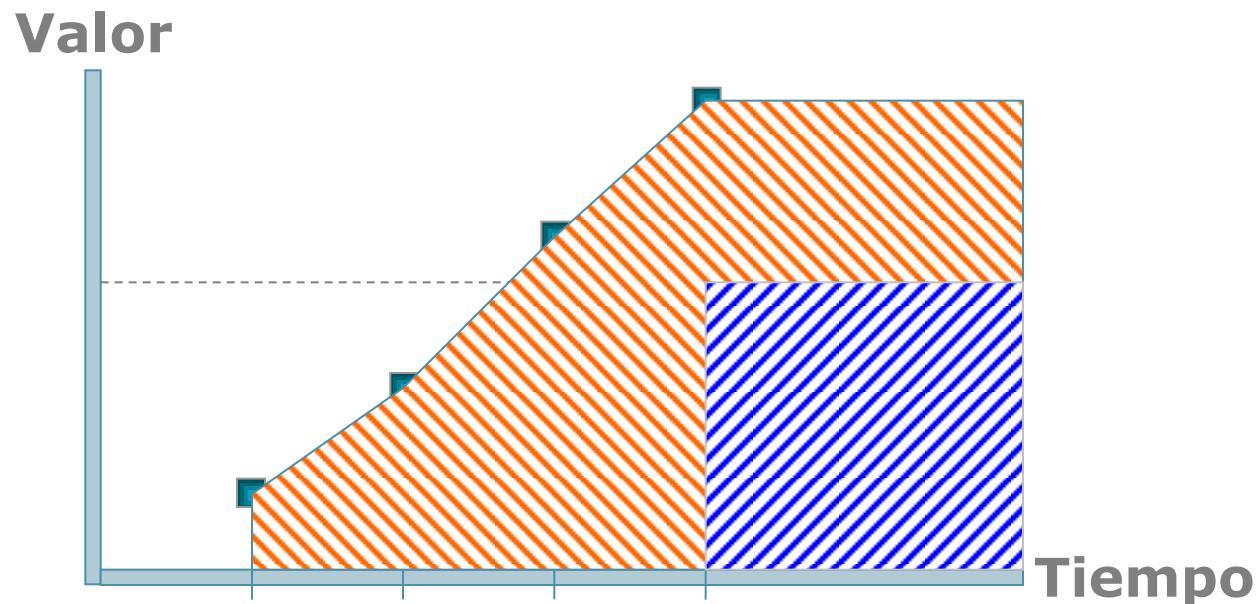
GESTIÓN ÁGIL: entrega de valor al cliente



GESTIÓN ÁGIL: entrega de valor al cliente



GESTIÓN ÁGIL: entrega de valor al cliente



- Se anticipa la presencia en el mercado.
- Las previsiones inversión / ROI se suavizan.
- Se mantiene la ventaja innovadora del producto en un mercado en movimiento.

AGILIDAD: entrega temprana, incremento continuo



2.007



2.008



2.009



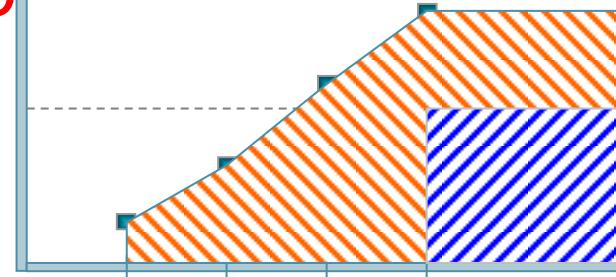
2.011

AGILIDAD

Valor temprano



Time to market



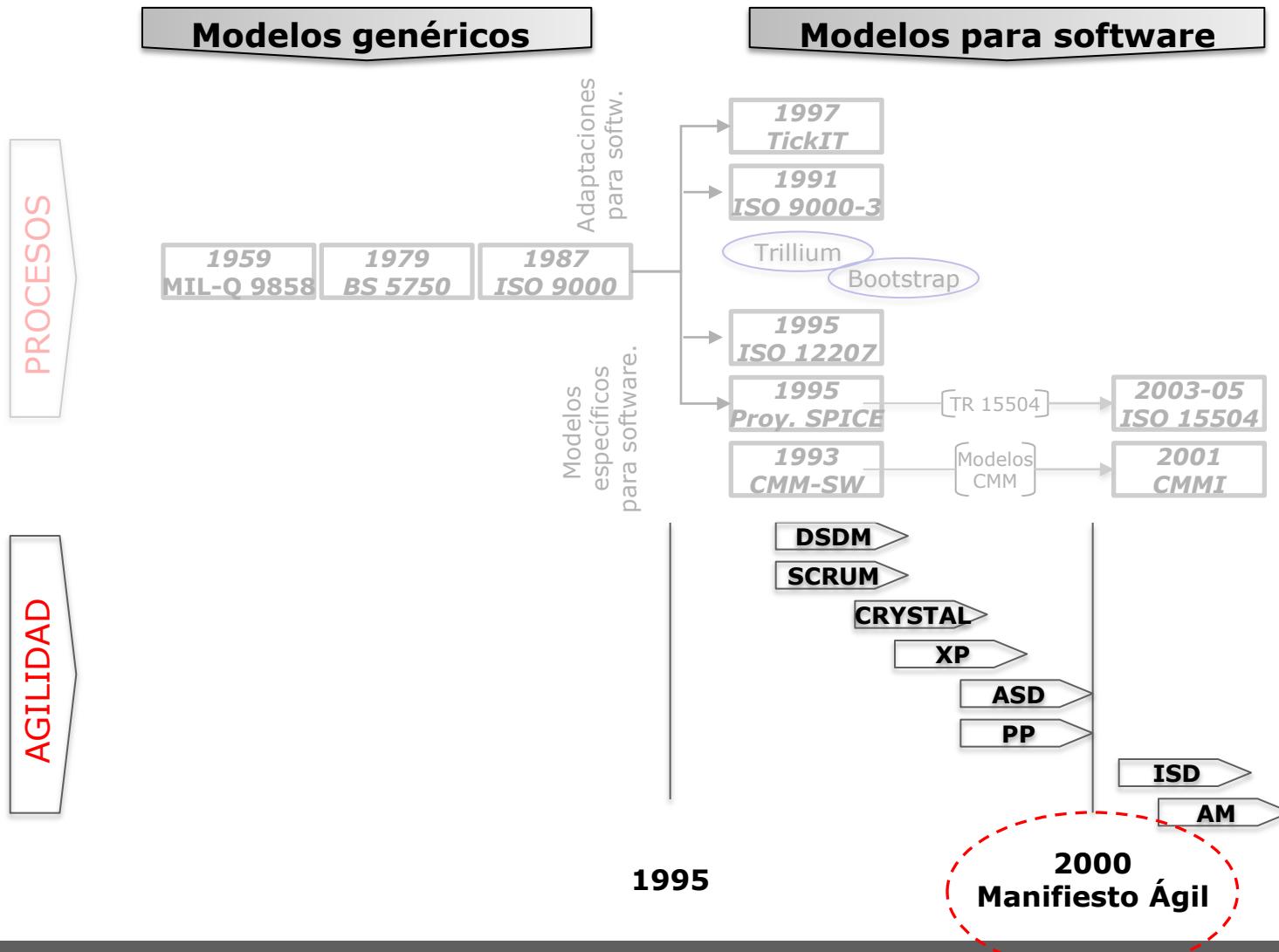
↔
2 meses



Desperdicio: 20%

Incremento continuo y
frecuente

POSTERIOR ANTÍTESIS A LOS MODELOS DE PROCESOS: AGILIDAD



MANIFIESTO ÁGIL

Marzo - 2001



Estamos poniendo al descubierto mejores
métodos para desarrollar software...

www.agilemanifesto.org

MANIFIESTO ÁGIL

VALOR



Cc by [Santi Siri](#)

HERRAMIENTAS Y PROCESOS

PERSONAS Y SU INTERACCIÓN

Communications Matrix							
Name/Nature of Communication	From	To	Content Provided By	Type	Frequency	Format Used	
Spokesperson							
Team Members							
Stakeholders							

Responsibility Matrix											
Document Owner:											
Responsibility											
Communication Task											
Identify tasks											
Define tasks											
Assign tasks											
Monitor tasks											
Review tasks											

Cc by [Tech Writer Boy](#)

MANIFIESTO ÁGIL



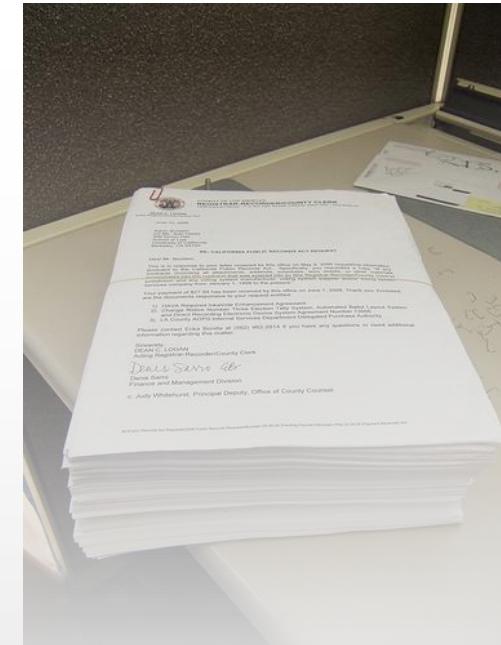
VALOR



Cc by [Thor](#)

DOCUMENTACIÓN
EXHAUSTIVA

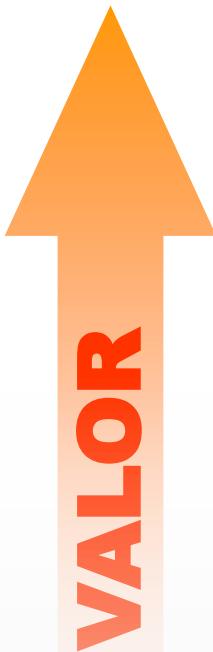
SOFTWARE QUE
FUNCIONA



Cc by [Joe Hall](#)

MANIFIESTO ÁGIL

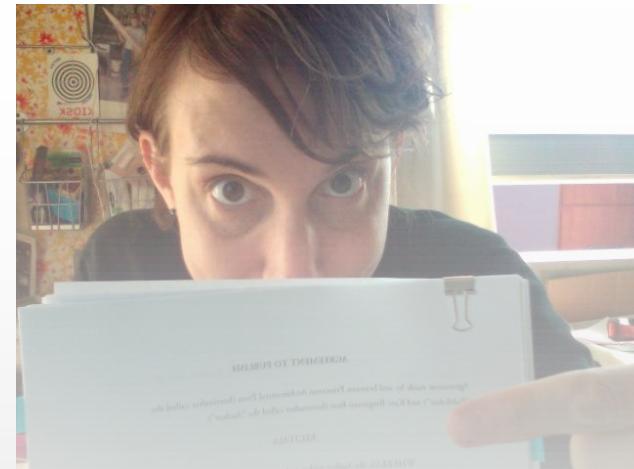
VALOR



Cc by [Karsten Konrad](#)

NEGOCIACIÓN CONTRACTUAL

COLABORACIÓN CON CLIENTE



Cc by [Kate Bingaman](#)

MANIFIESTO ÁGIL

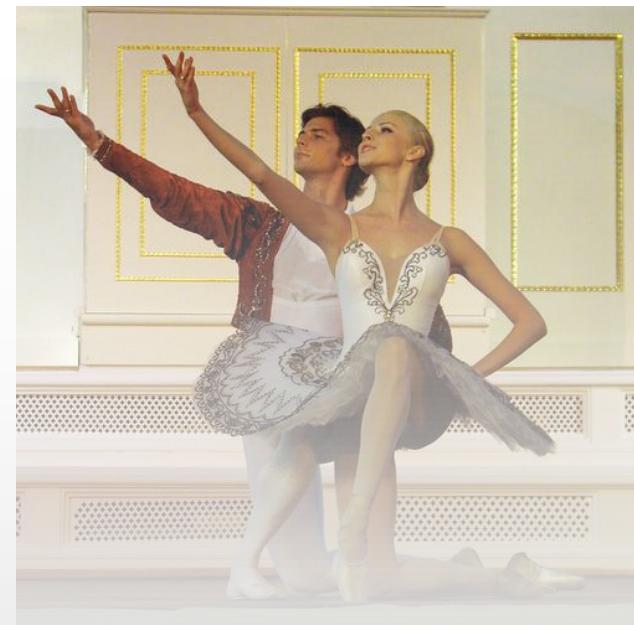
VALOR



Cc by [Jonny Hunter](#)

**SEGUIMIENTO
DE UN PLAN**

**RESPUESTA
AL CAMBIO**



Cc by [J.P. Dalbéra](#)

Los 12 principios del manifiesto ágil



1.- Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

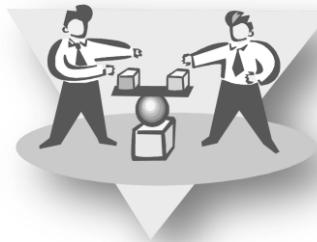


2.- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

Los 12 principios del manifiesto ágil



3.- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.



4.- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

Los 12 principios del manifiesto ágil



5.- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

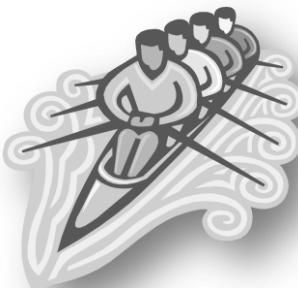


6.- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara

Los 12 principios del manifiesto ágil



7.- El software que funciona es la principal medida del progreso.



8.- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

Los 12 principios del manifiesto ágil



9.- La atención continua a la excelencia técnica enaltece la agilidad.



10.- La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.

Los 12 principios del manifiesto ágil

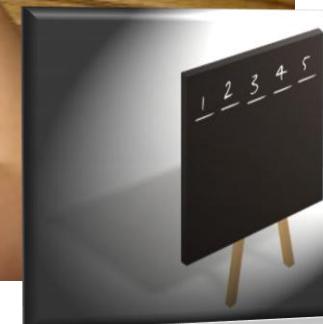


11.- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.



12.- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Práctica



cc-by: [LizMarie](#)

Práctica



15.- Minutos



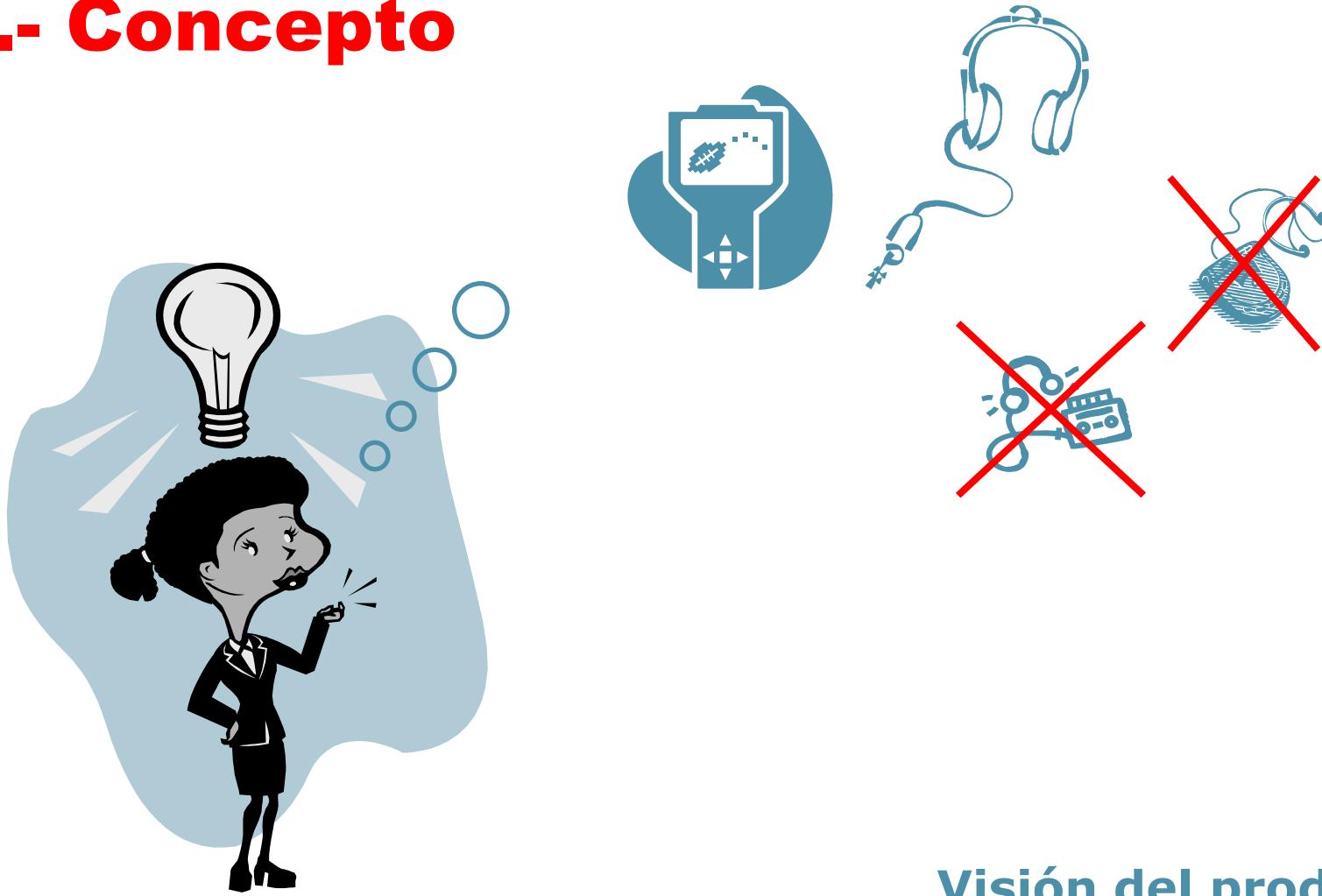
En grupos



Palabra o palabras (Máx. 3)
de síntesis de cada principio

Ciclo de desarrollo ágil

1.- Concepto



Visión del producto

Ciclo de desarrollo ágil

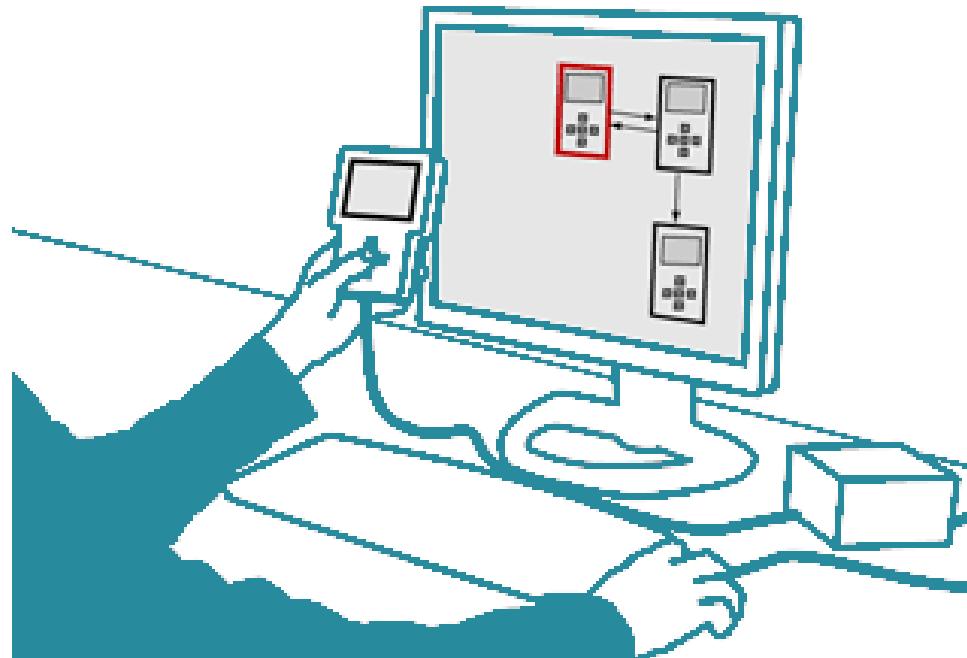
2.- Especulación



Presentación y discusión con usuarios

Ciclo de desarrollo ágil

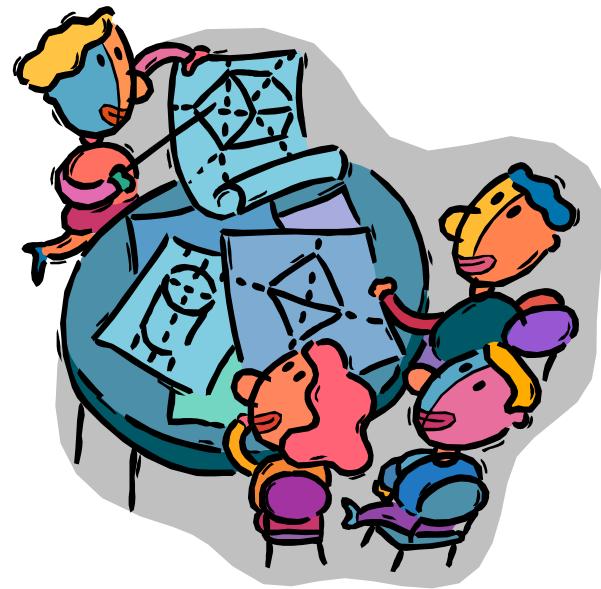
3.- Exploración



Desarrollo de un incremento

Ciclo de desarrollo ágil

4.- Revisión



Entrega y retro-información

Ciclo de desarrollo ágil

5.- Cierre



Producto final

Ciclo de desarrollo ágil



Concepto



Cierre

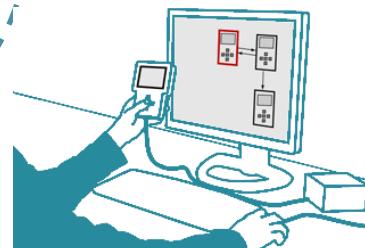


Revisión

Especulación

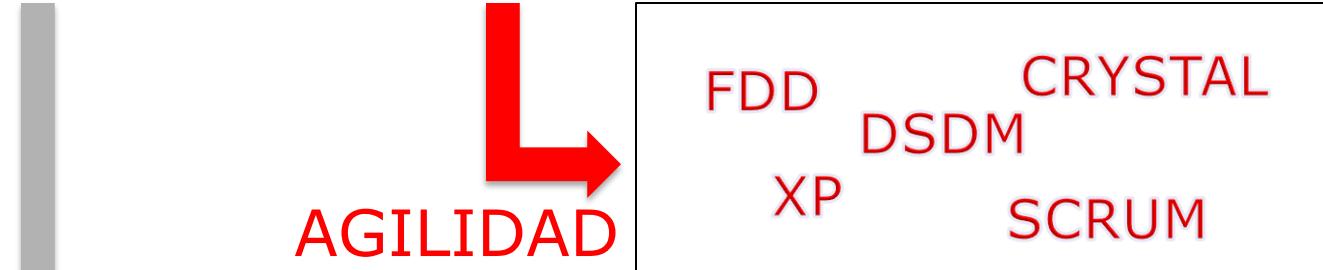
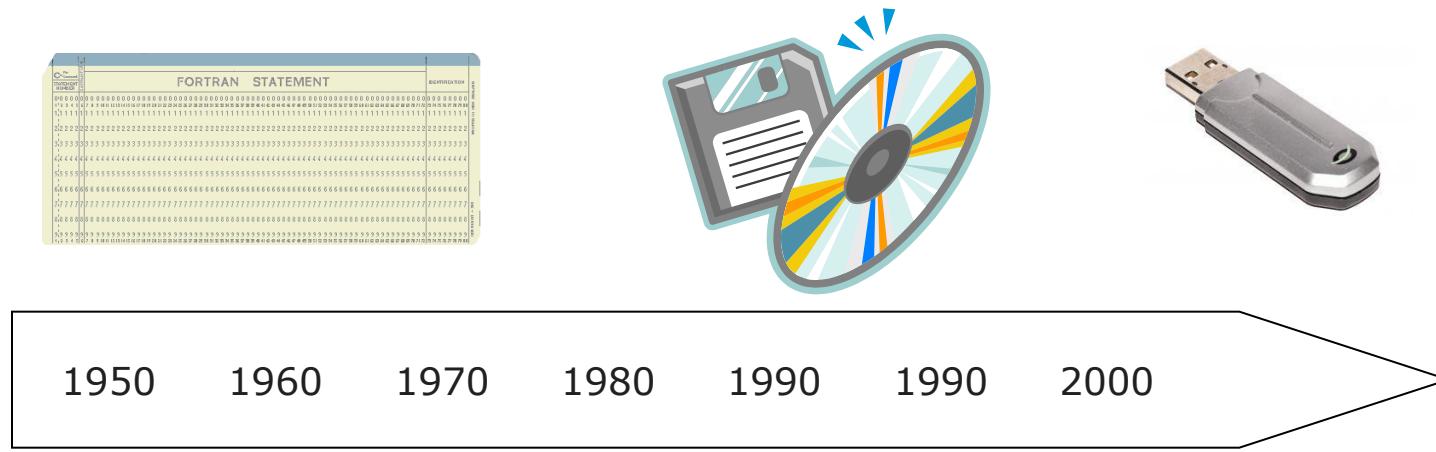


Iteraciones

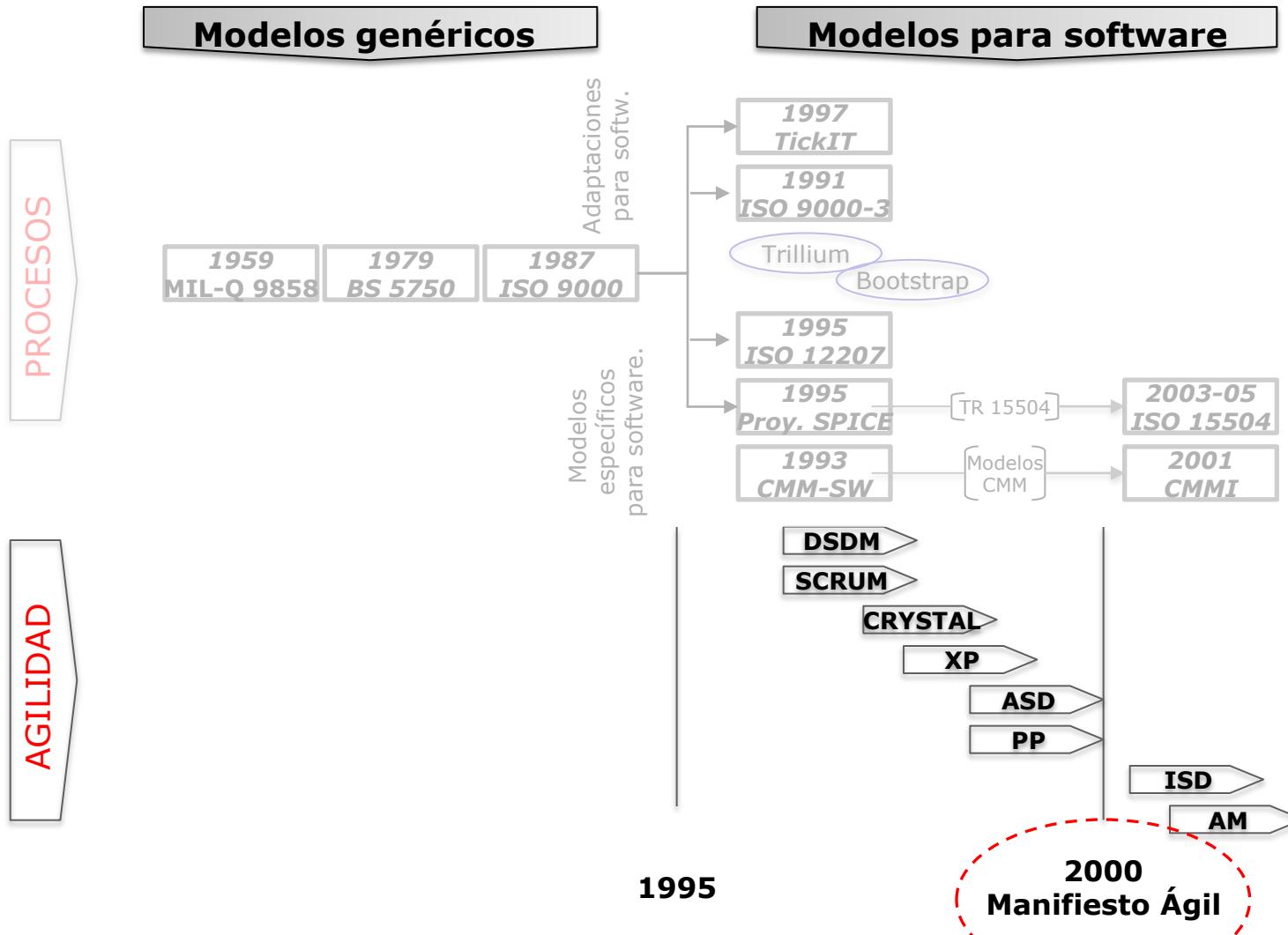


Exploración

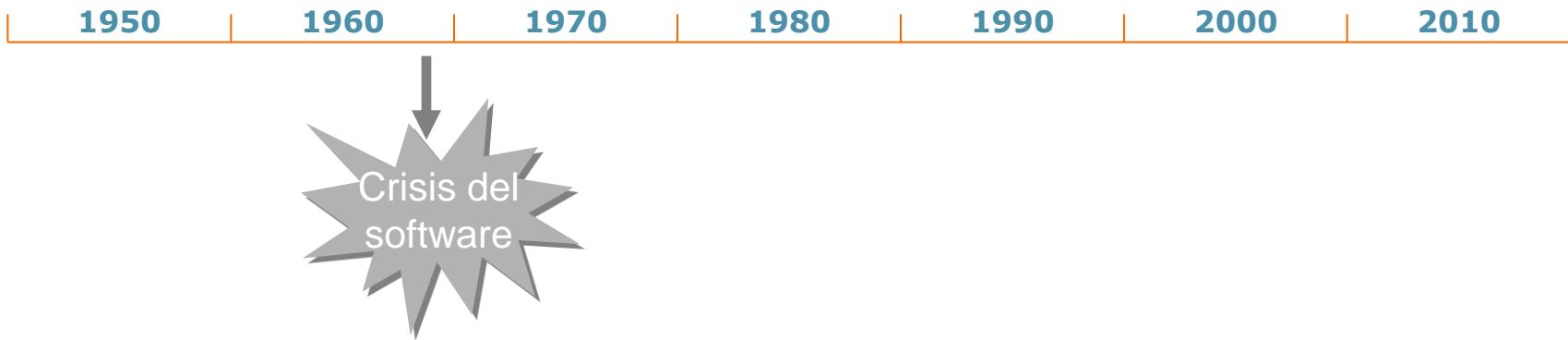
MANIFIESTO ÁGIL



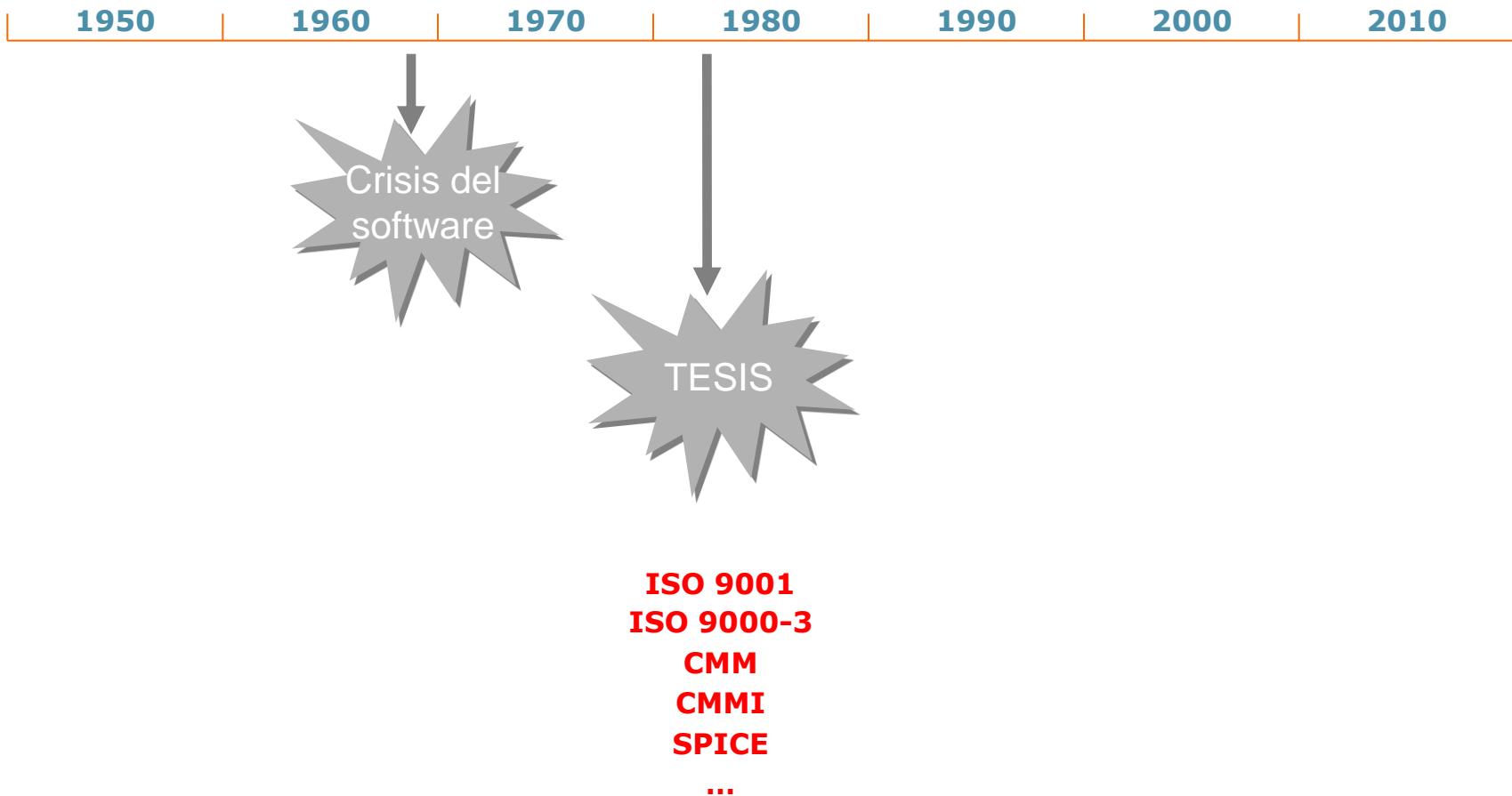
ANTÍTESIS A LOS MODELOS DE PROCESOS: AGILIDAD



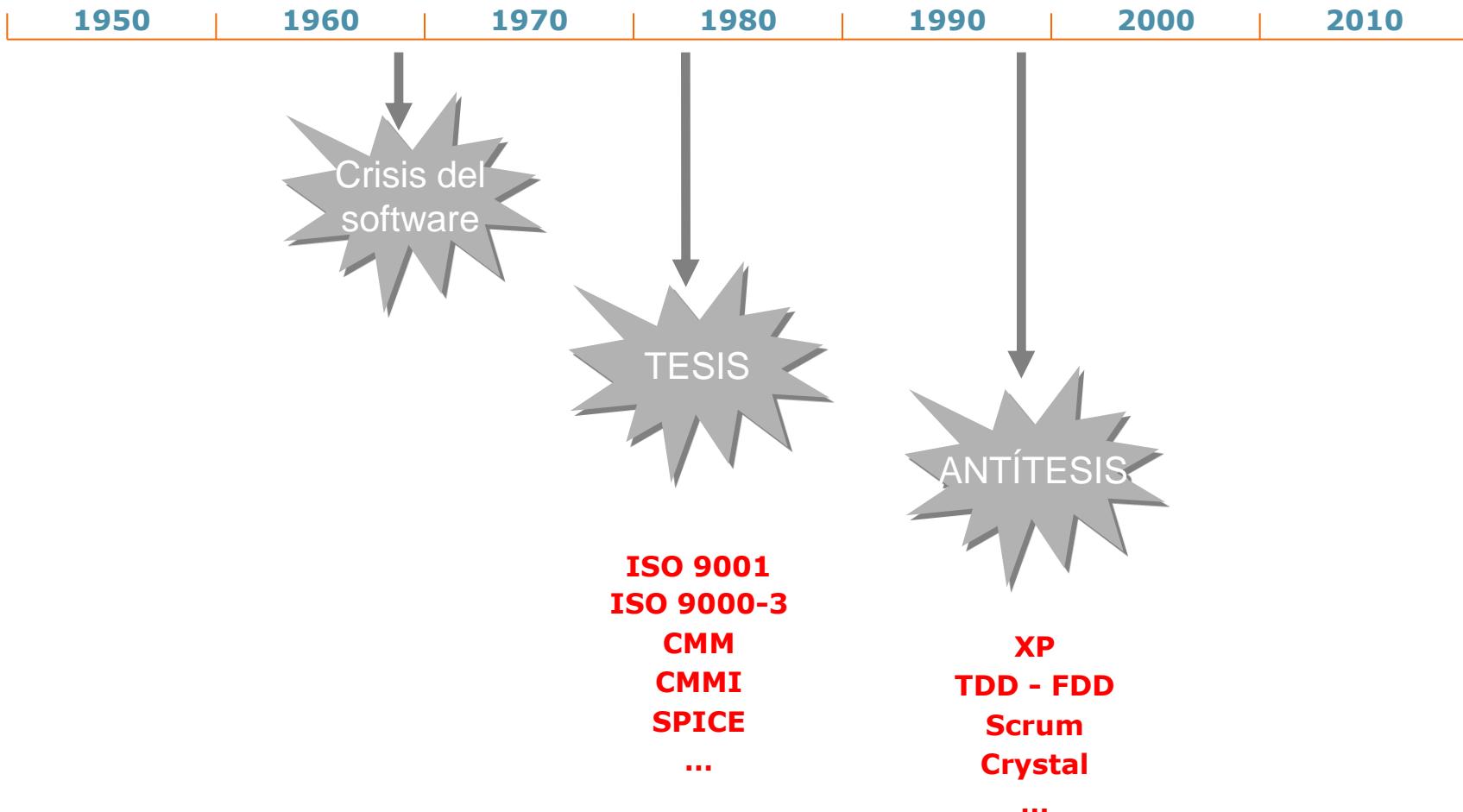
EVOLUCIÓN DEL CONOCIMIENTO



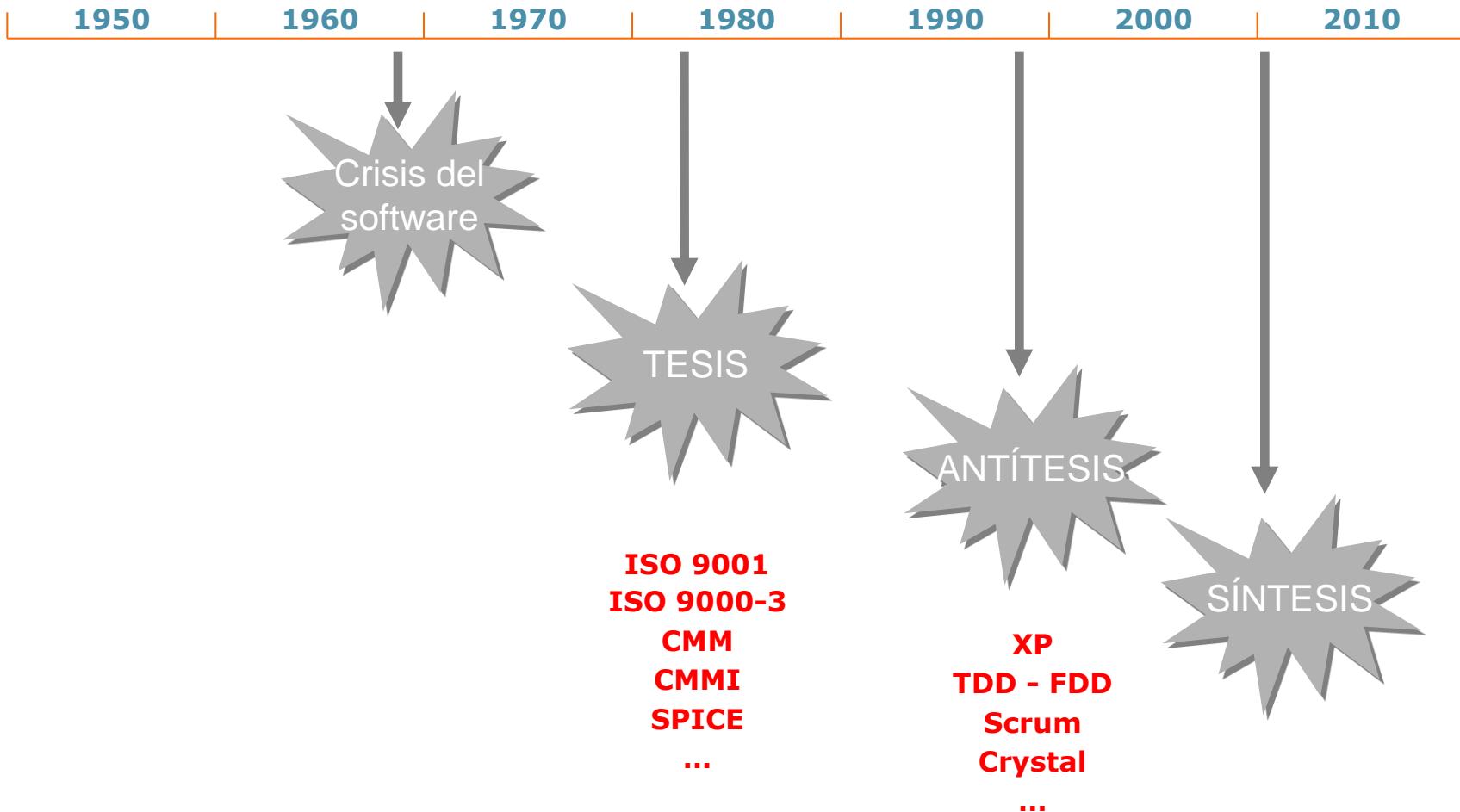
EVOLUCIÓN DEL CONOCIMIENTO



EVOLUCIÓN DEL CONOCIMIENTO



EVOLUCIÓN DEL CONOCIMIENTO

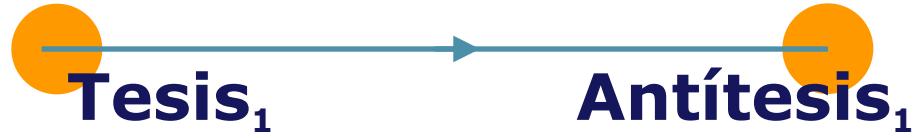


EVOLUCIÓN DEL CONOCIMIENTO

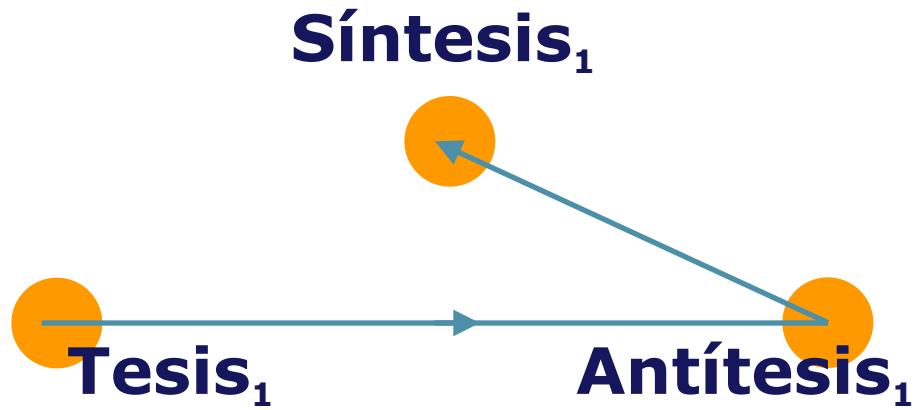


Tesis₁

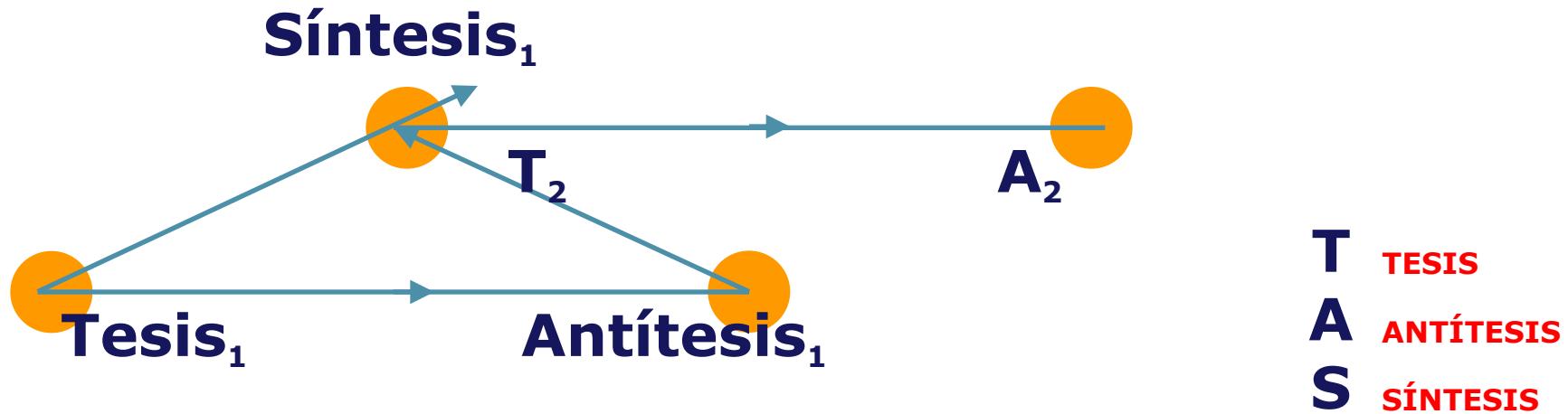
EVOLUCIÓN DEL CONOCIMIENTO



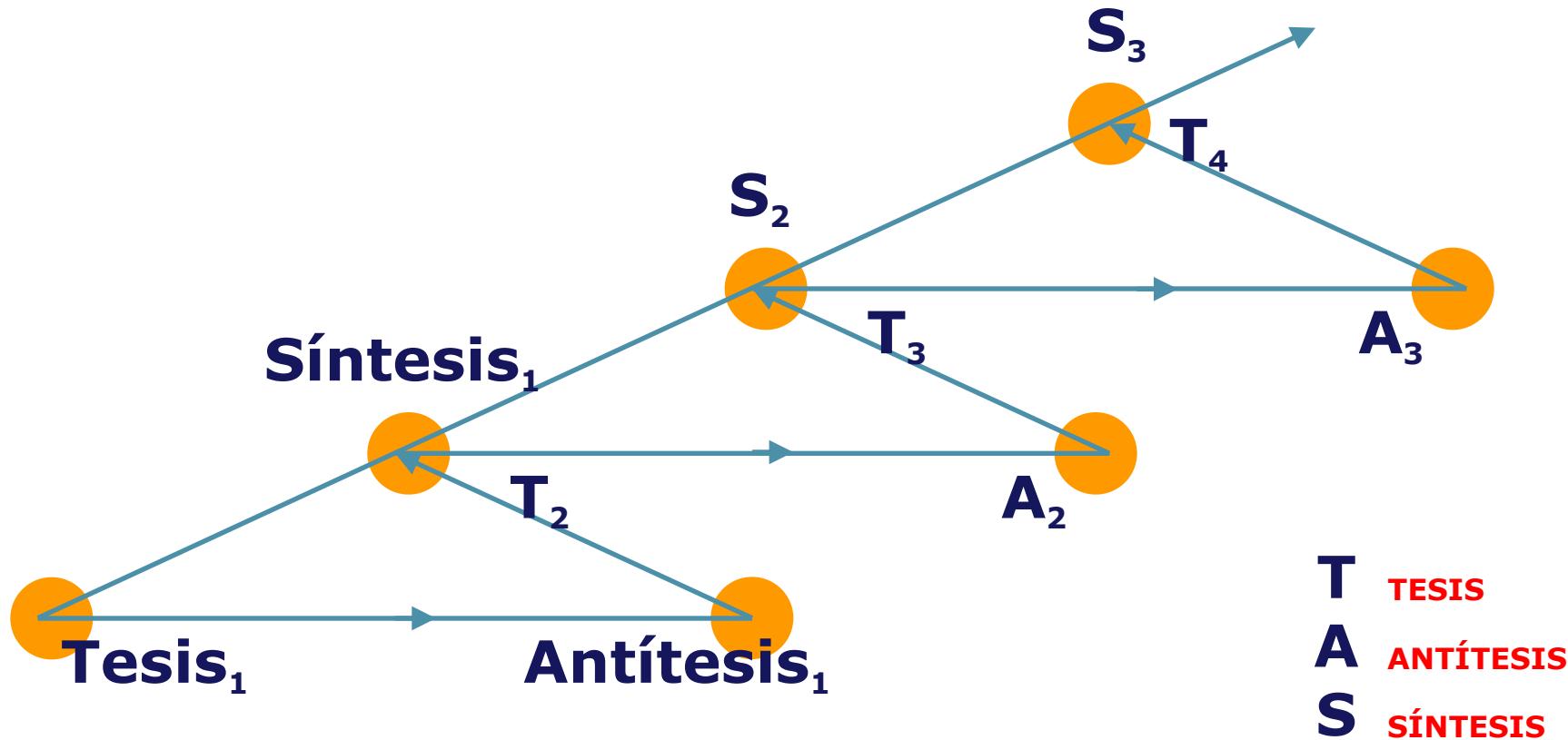
EVOLUCIÓN DEL CONOCIMIENTO



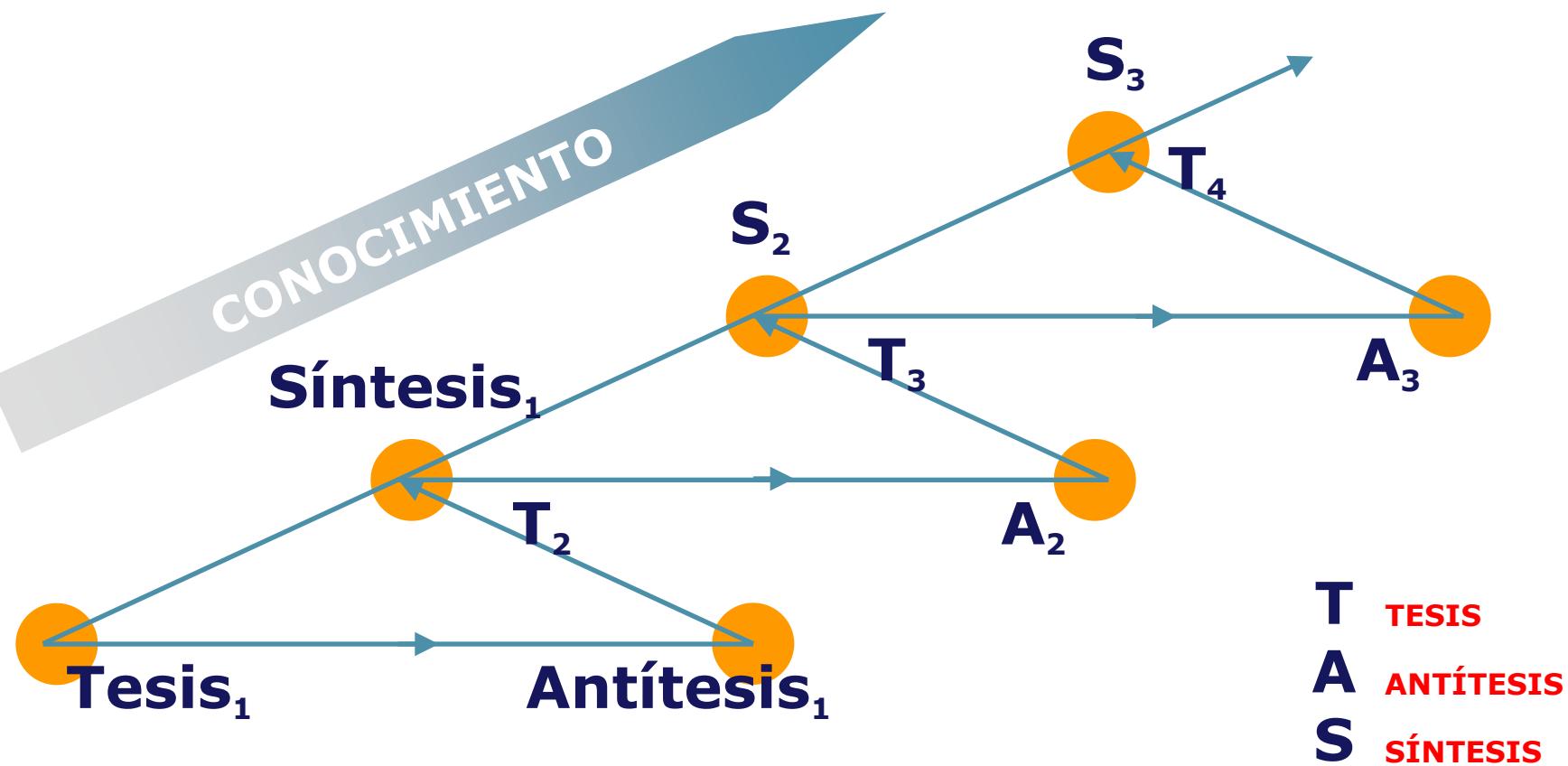
EVOLUCIÓN DEL CONOCIMIENTO



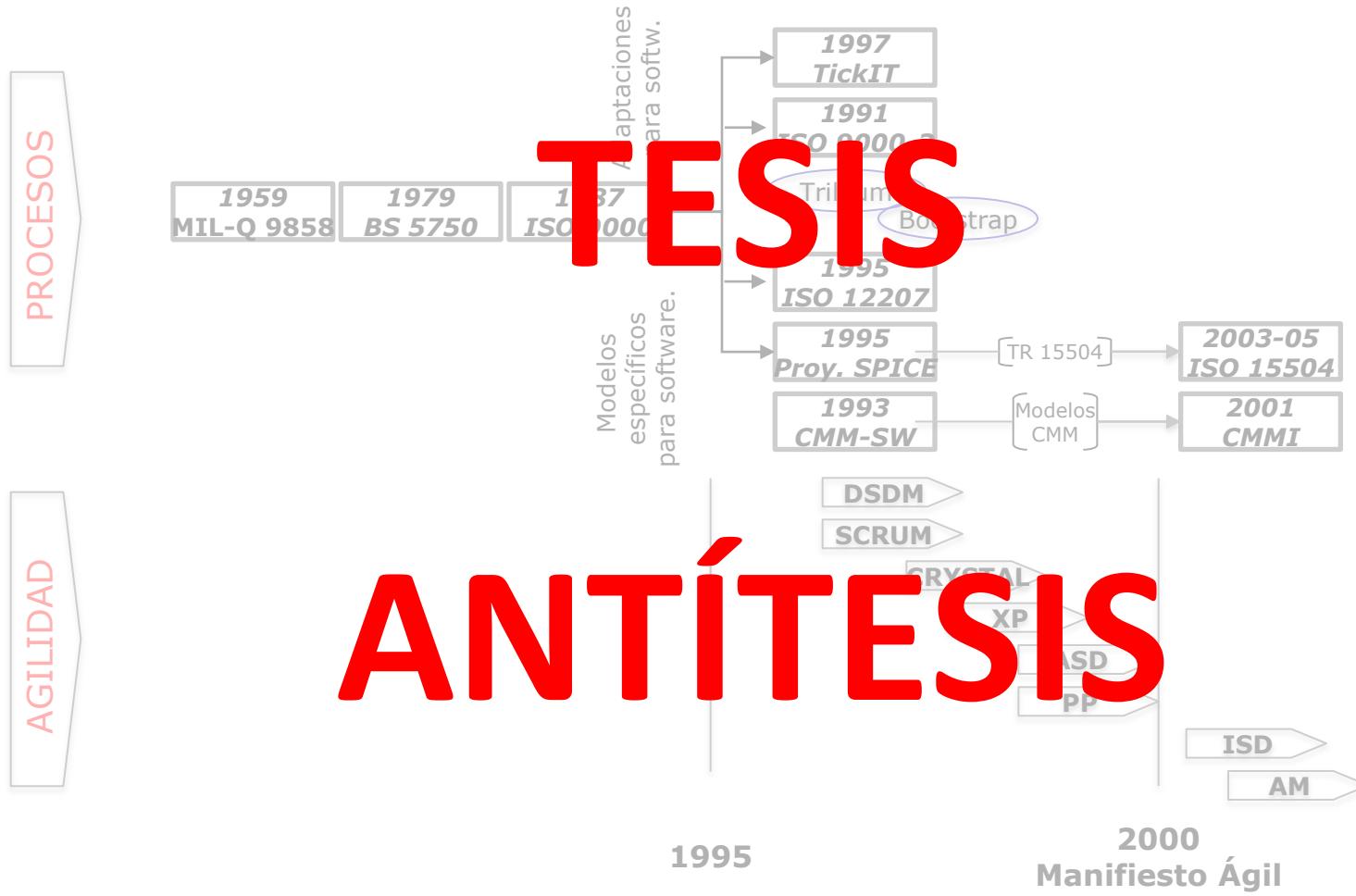
EVOLUCIÓN DEL CONOCIMIENTO



EVOLUCIÓN DEL CONOCIMIENTO



ANTÍTESIS A LOS MODELOS DE PROCESOS: AGILIDAD



(Conocimiento)



¿Procesos y previsión / Agilidad y cambio continuo?



Un paréntesis... sobre el conocimiento



Conocimiento para realizar un trabajo:

Tácito



Explícito



Los elementos de producción

PERSONAS



Tácito



PROCESOS



Explícito

TECNOLOGÍA

TESIS: Responsabilidad del resultado

PERSONAS



PROCESOS

TECNOLOGÍA

Antítesis: responsabilidad del resultado

PERSONAS



PROCESOS



TECNOLOGÍA



Visión de síntesis: Procesos = PROCEDIMIENTOS



PERSONAS

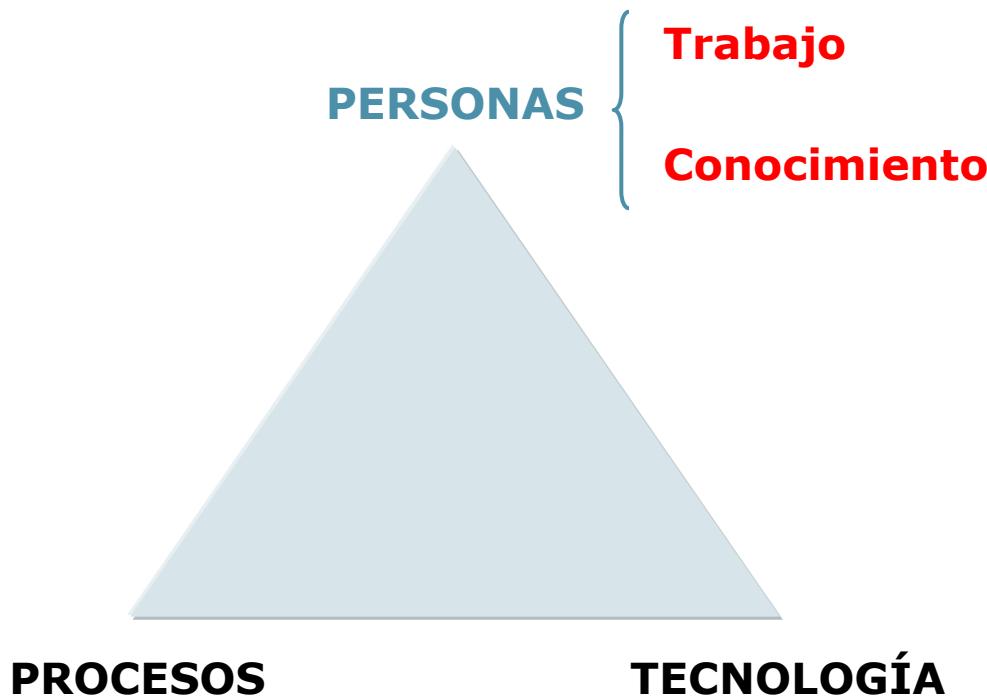
PROCEDIMIENTOS

TECNOLOGÍA

Procesos
Rutinas



Visión de síntesis: PERSONAS



Práctica: conocimiento tácito y explícito

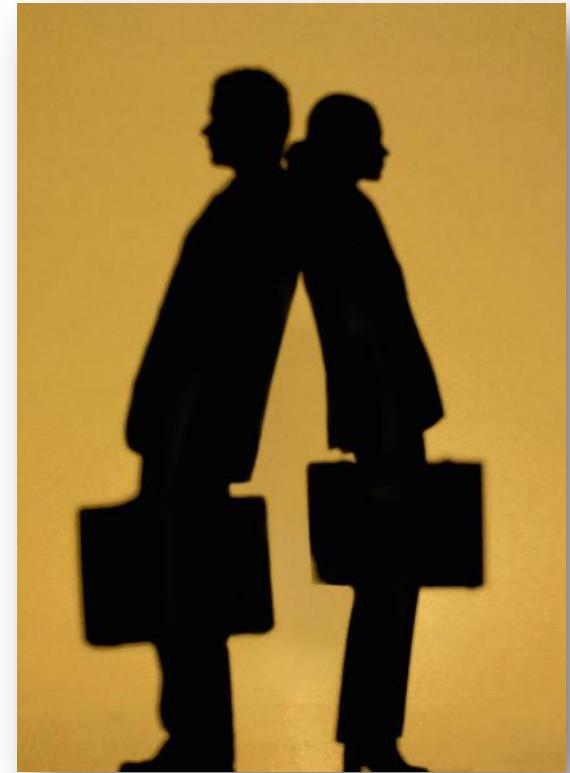


cc-by: [LizMarie](#)

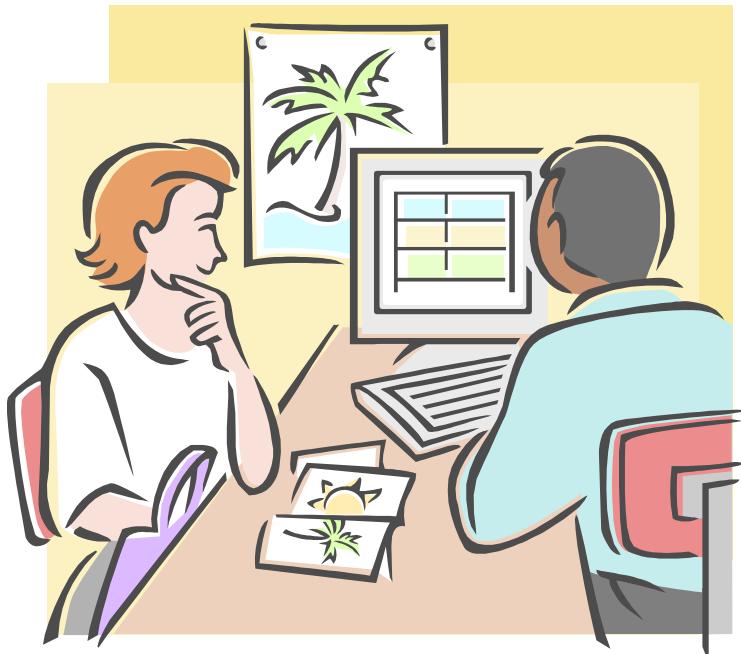
... sobre el conocimiento)



Síntesis entre procesos y
agilidad



Hipótesis inicial

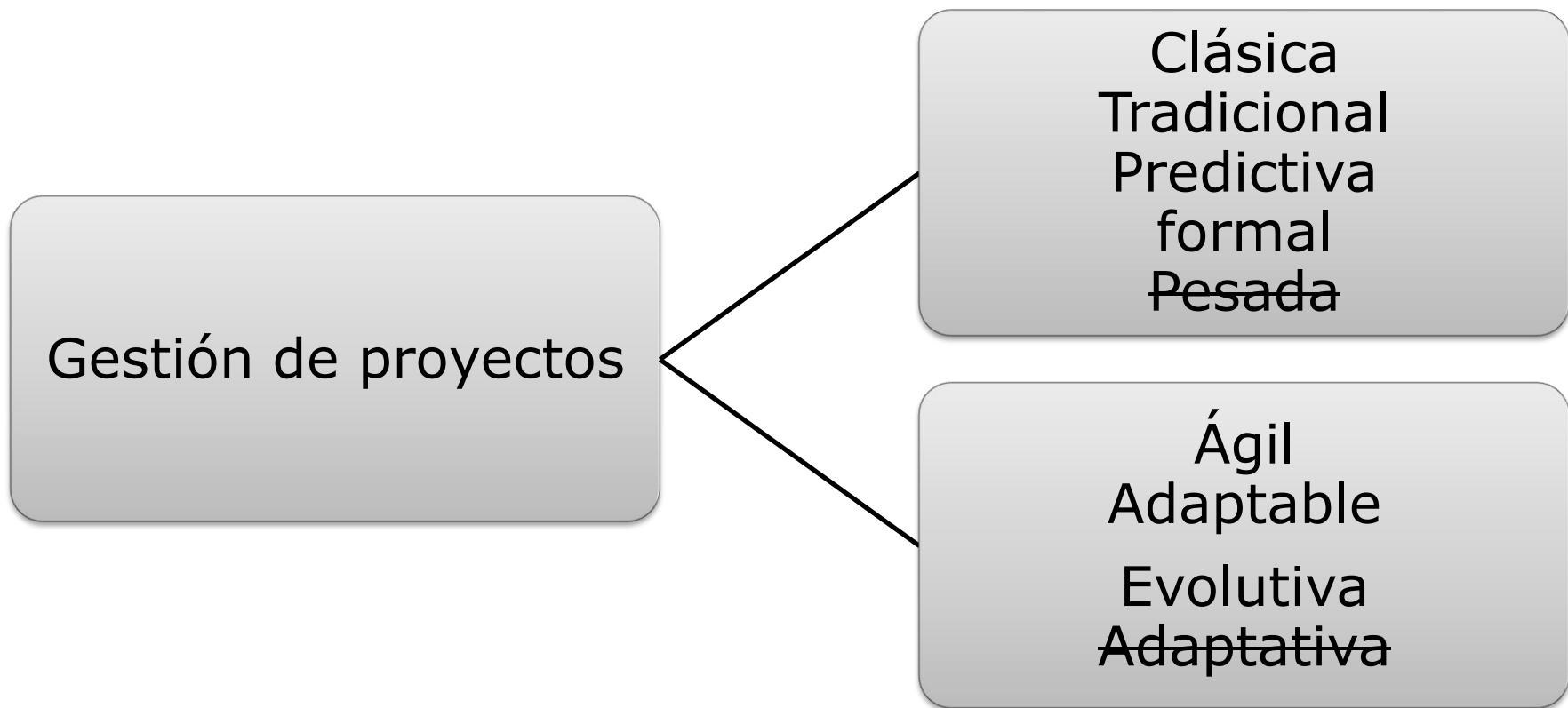


PLANIFICACIÓN

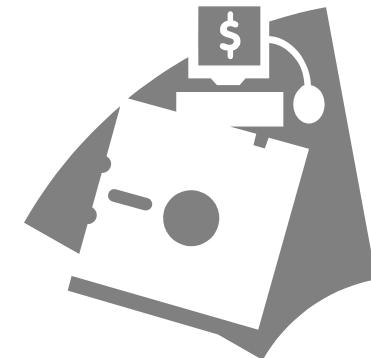
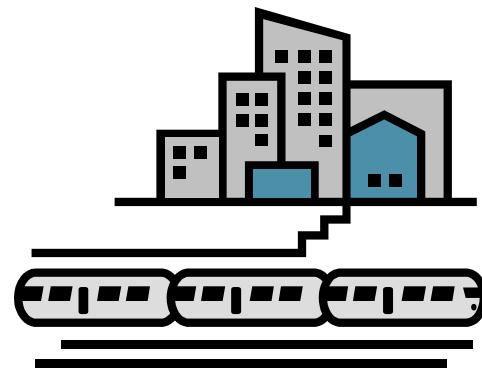
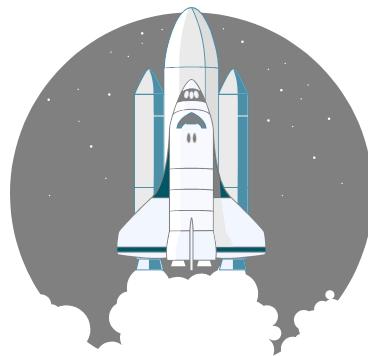


ADAPTACIÓN

¿Ágil, clásica, predictiva...?



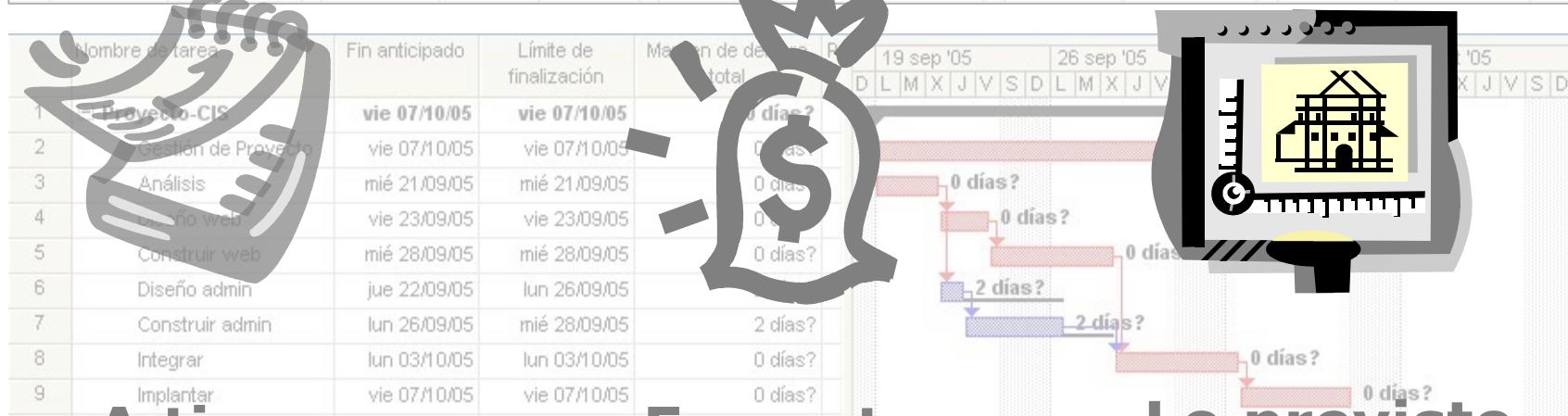
Premisas de la gestión de proyectos predictiva



Características y comportamientos regulares

¿Ágil, clásica, predictiva...?

OBJETIVO



A tiempo

En costes

Lo previsto

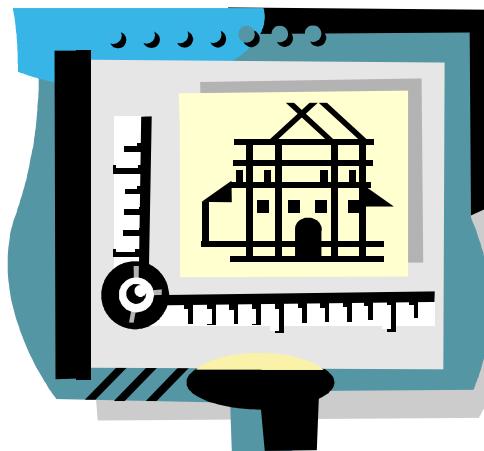
Características de la gestión de proyectos predictiva

UNIVERSALIDAD



Gestión predictiva – gestión evolutiva

PREDICTIVA



Definición

Diseño

Construcción

EVOLUTIVA



Visión

Exploración

Adaptación

Gestión predictiva – gestión evolutiva

PREDICTIVA

EVOLUTIVA

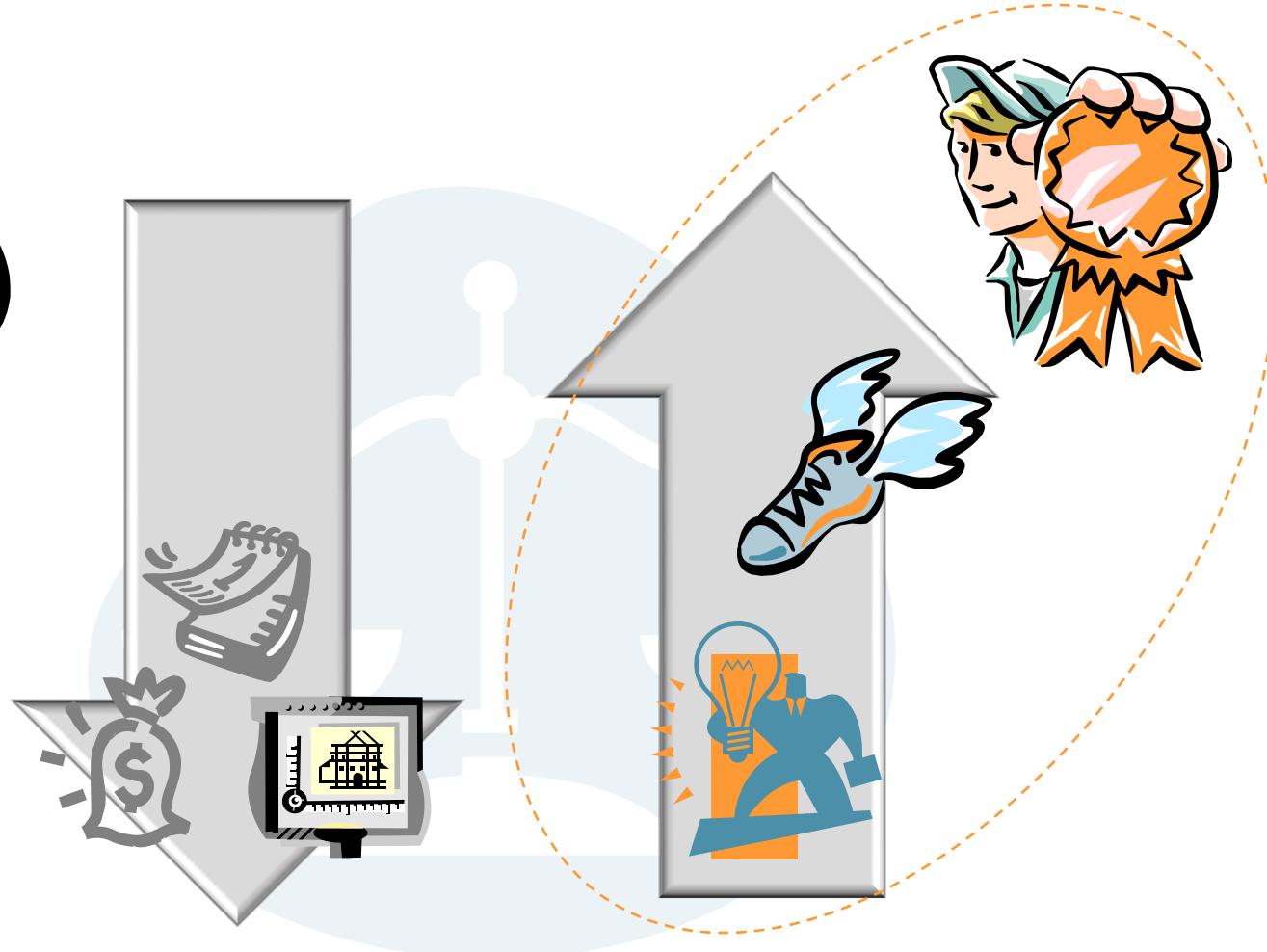
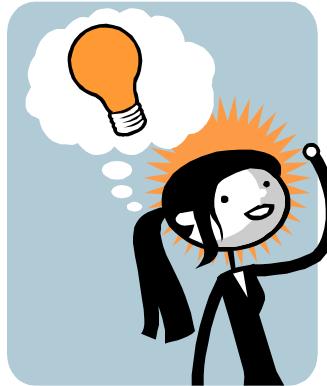


Gestión predictiva – gestión evolutiva

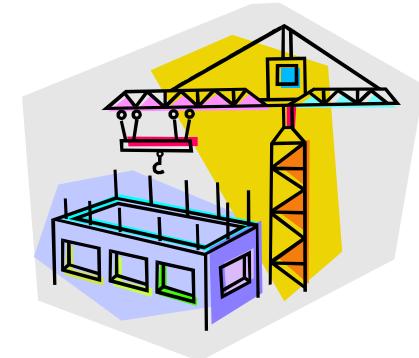
CRITERIOS DE IDONEIDAD

	EVOLUTIVA	PREDICTIVA
PRIORIDAD DE NEGOCIO	Valor	Cumplimiento
ESTABILIDAD DE REQUISITOS	Entorno inestable	Entorno estable
RIGIDEZ DEL PRODUCTO	Modifiable	Difícil de modificar
COSTE DE PROTOTIPADO	Bajo	Alto
CRITICIDAD DEL SISTEMA	Baja	Alta
TAMAÑO DEL EQUIPO	Pequeño	Grande

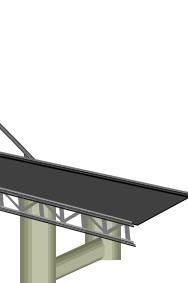
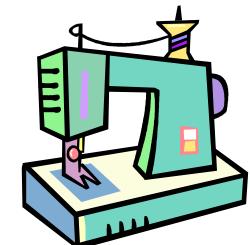
¿Producto, fecha y costes planificados?



¿Mismo patrón de gestión para todos?



Google™



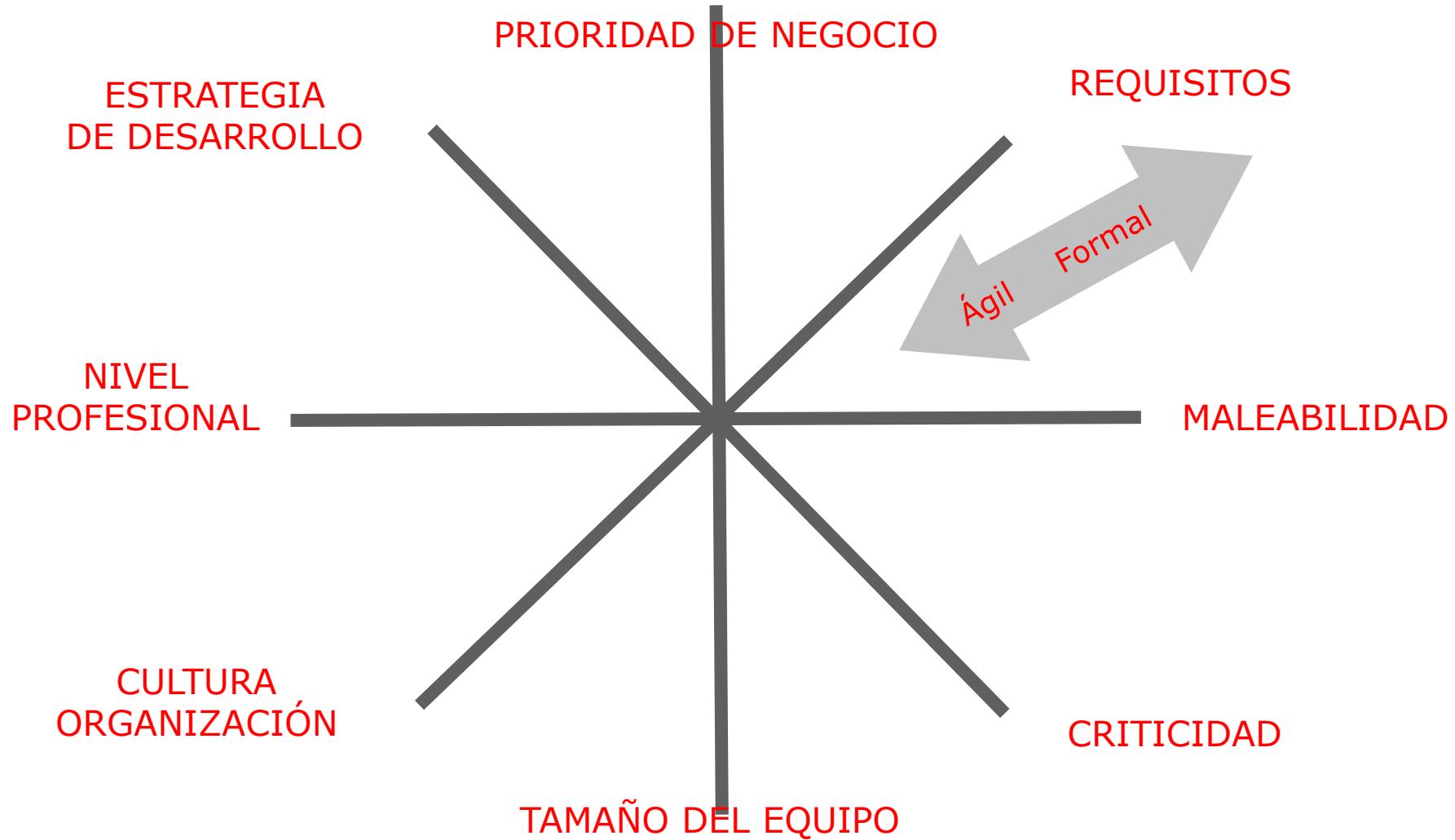
flickr™
BETA



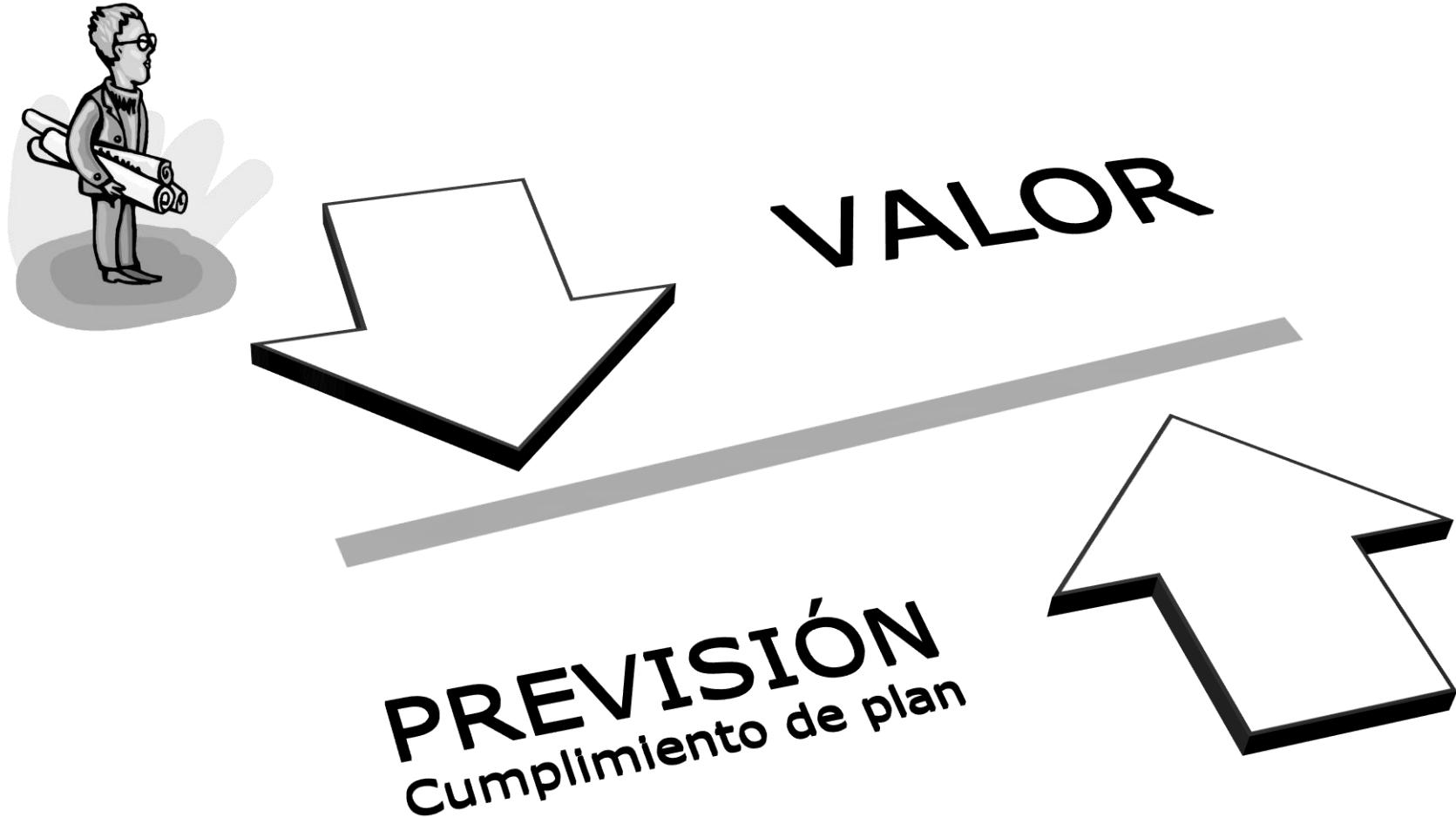
Criterios dependientes de:



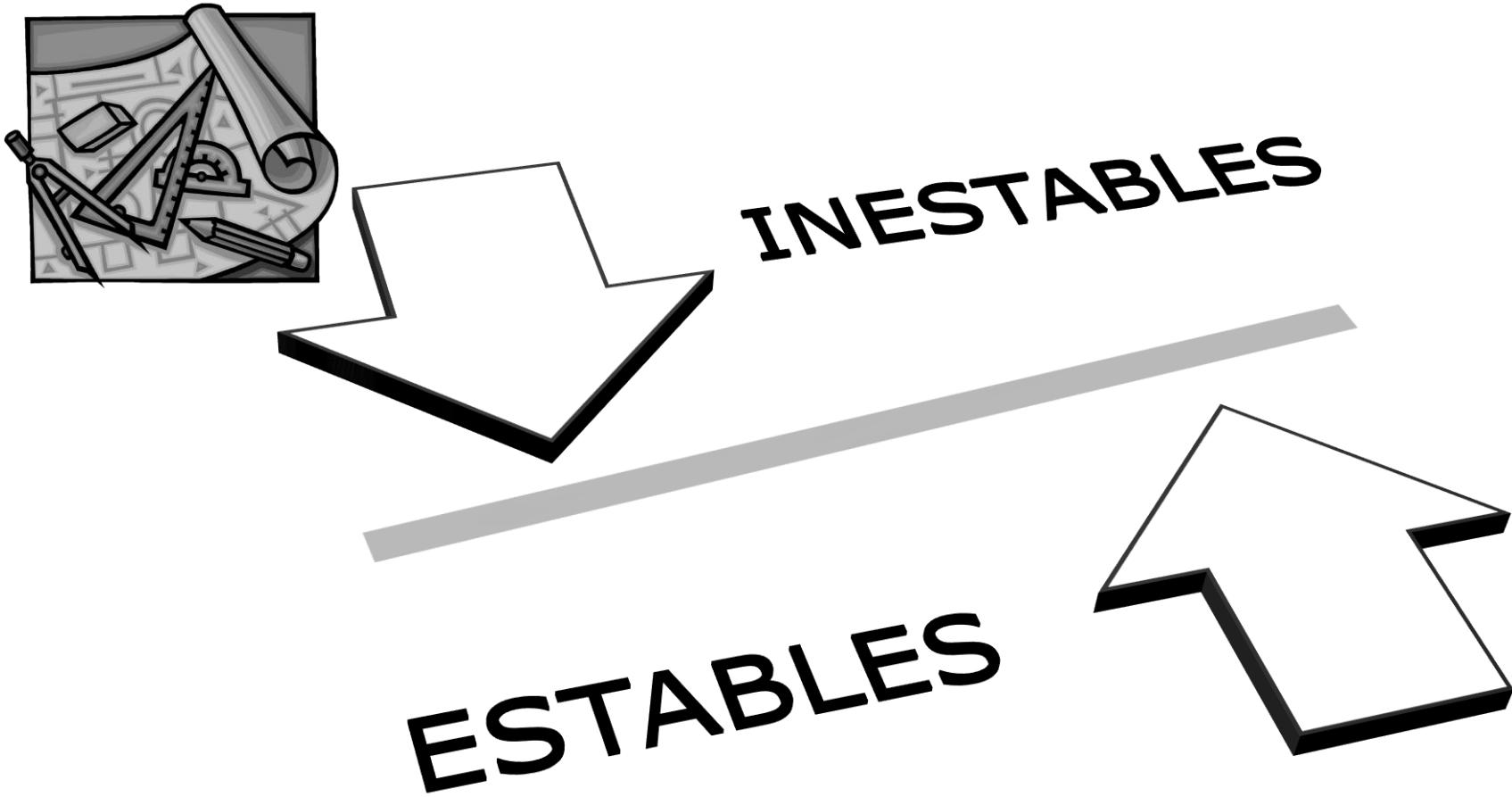
Criterios



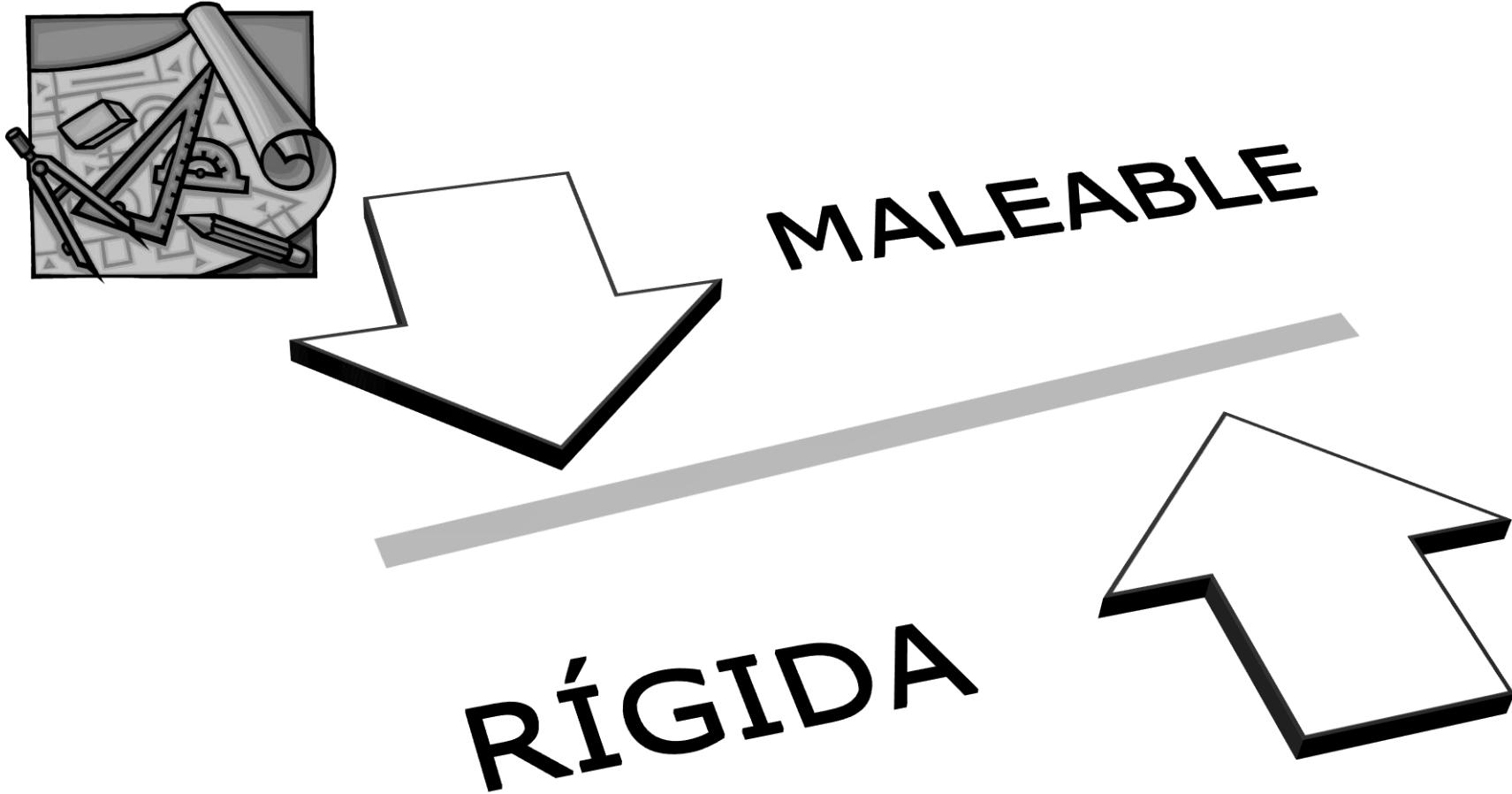
Prioridad de negocio



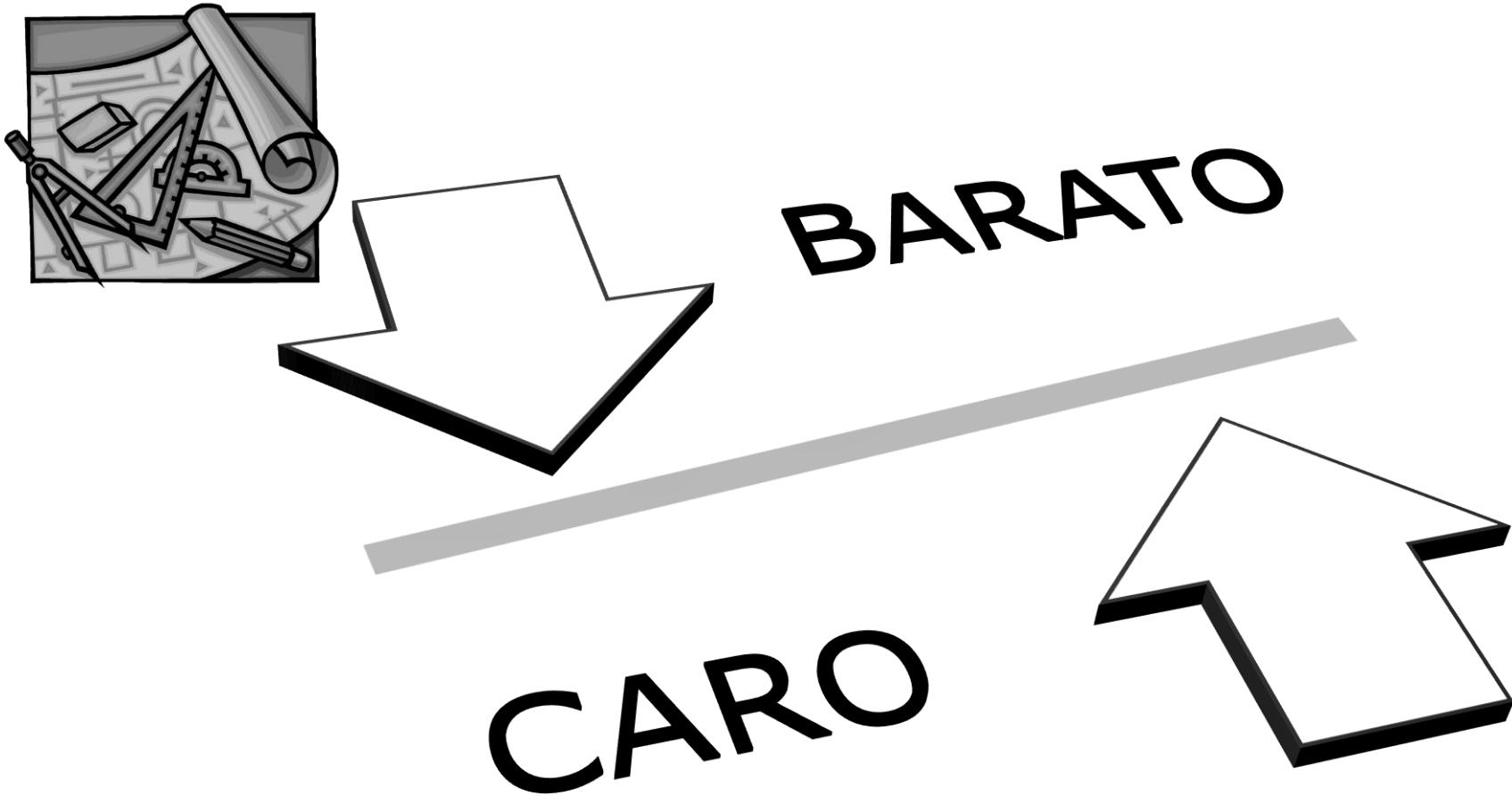
Requisitos



Maleabilidad (Materia prima)



Coste de prototipado



Coste de prototipado

COSTE / BENEFICIO

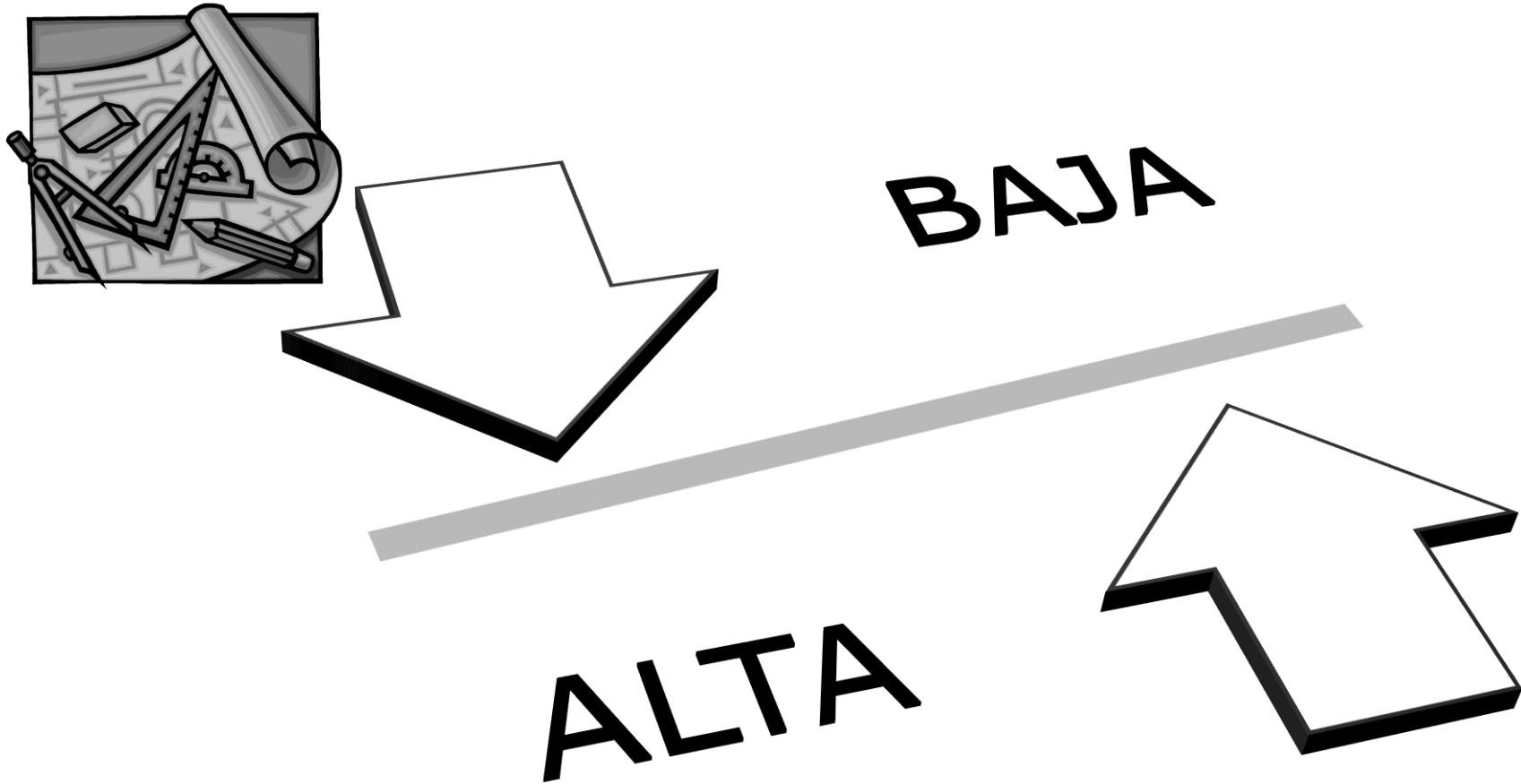


PROTOTIPADO

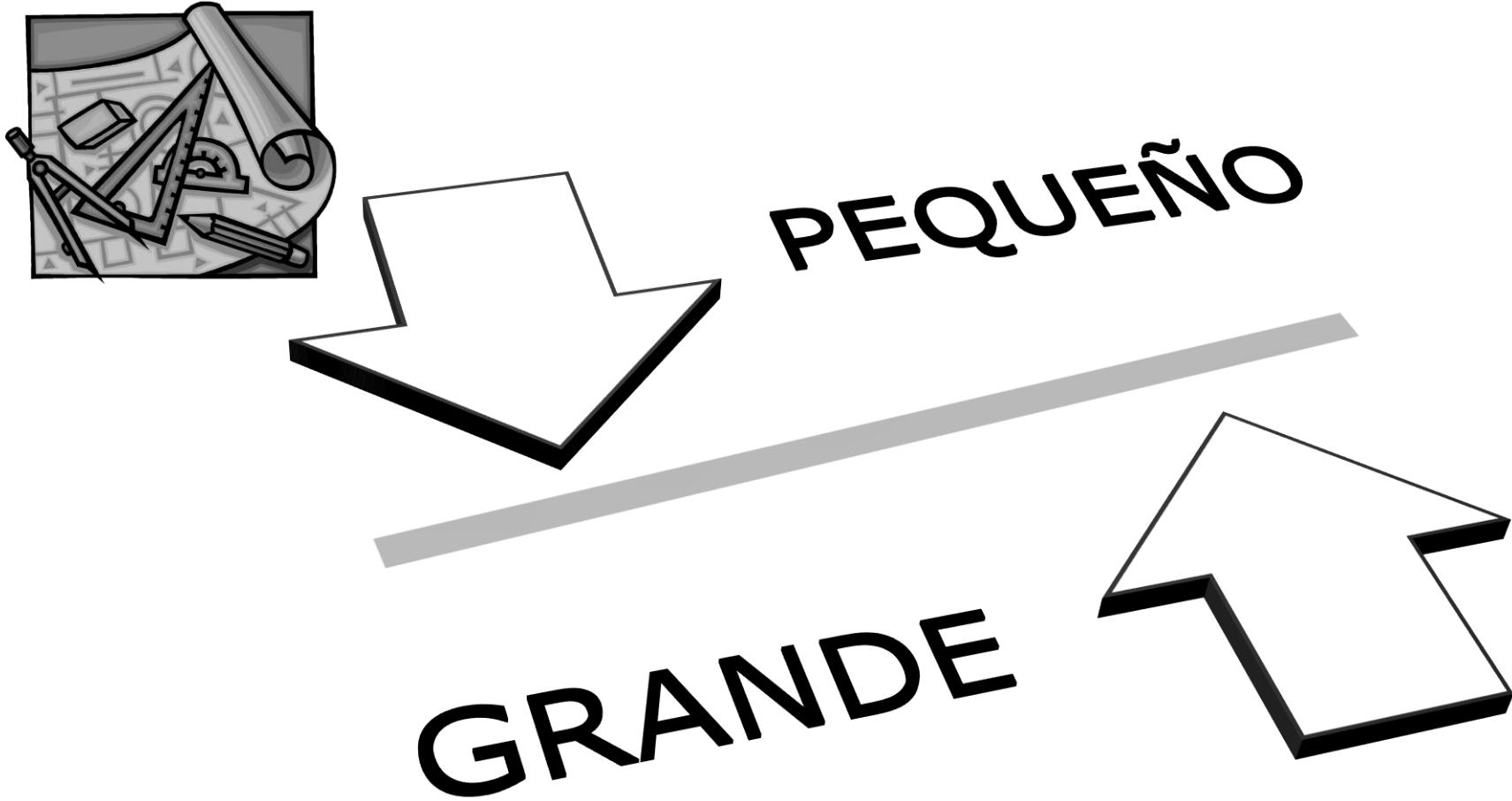
NO REHACER - PLANIFICAR

PROBAR Y EXPLORAR

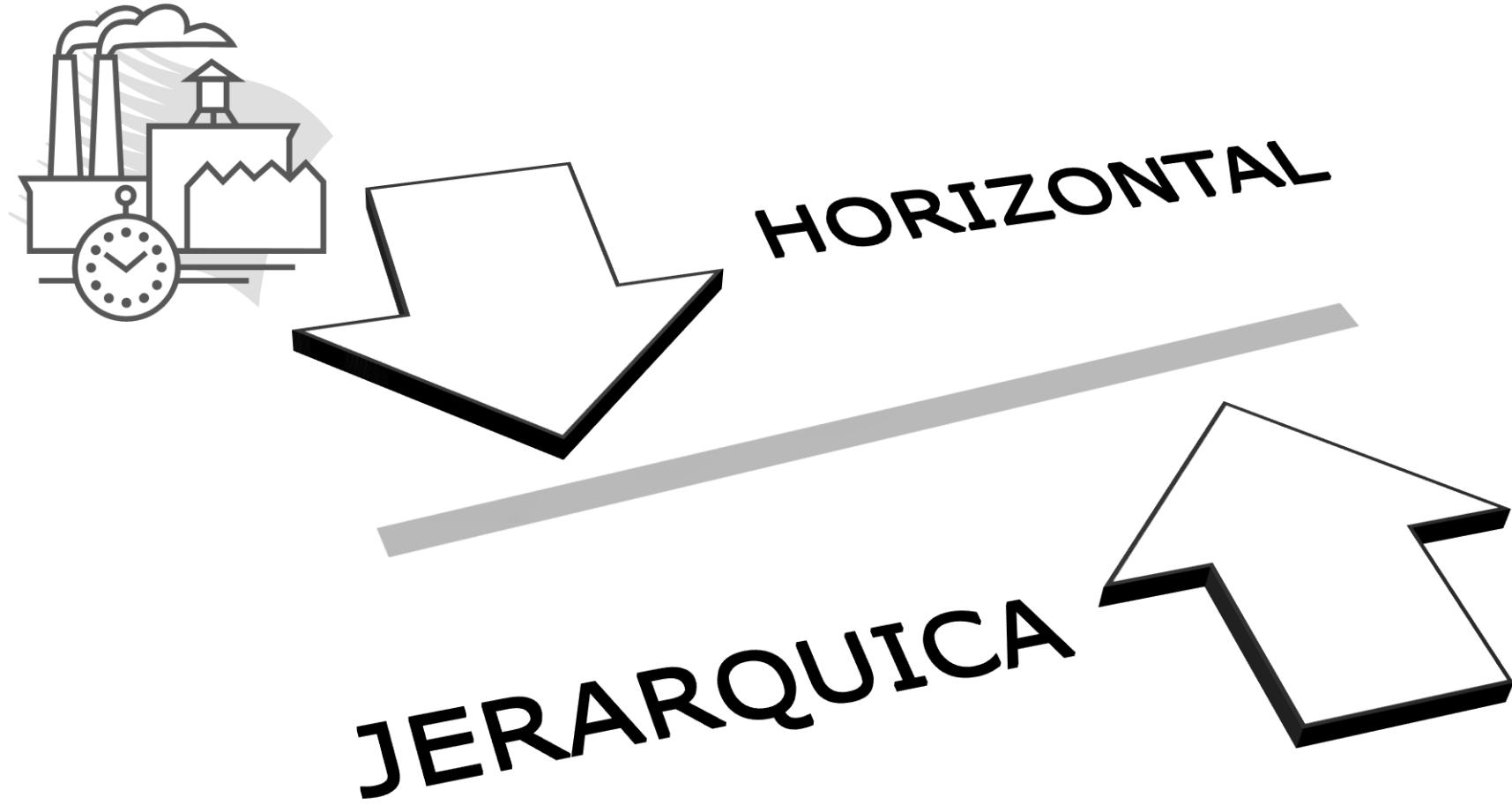
Criticidad del sistema



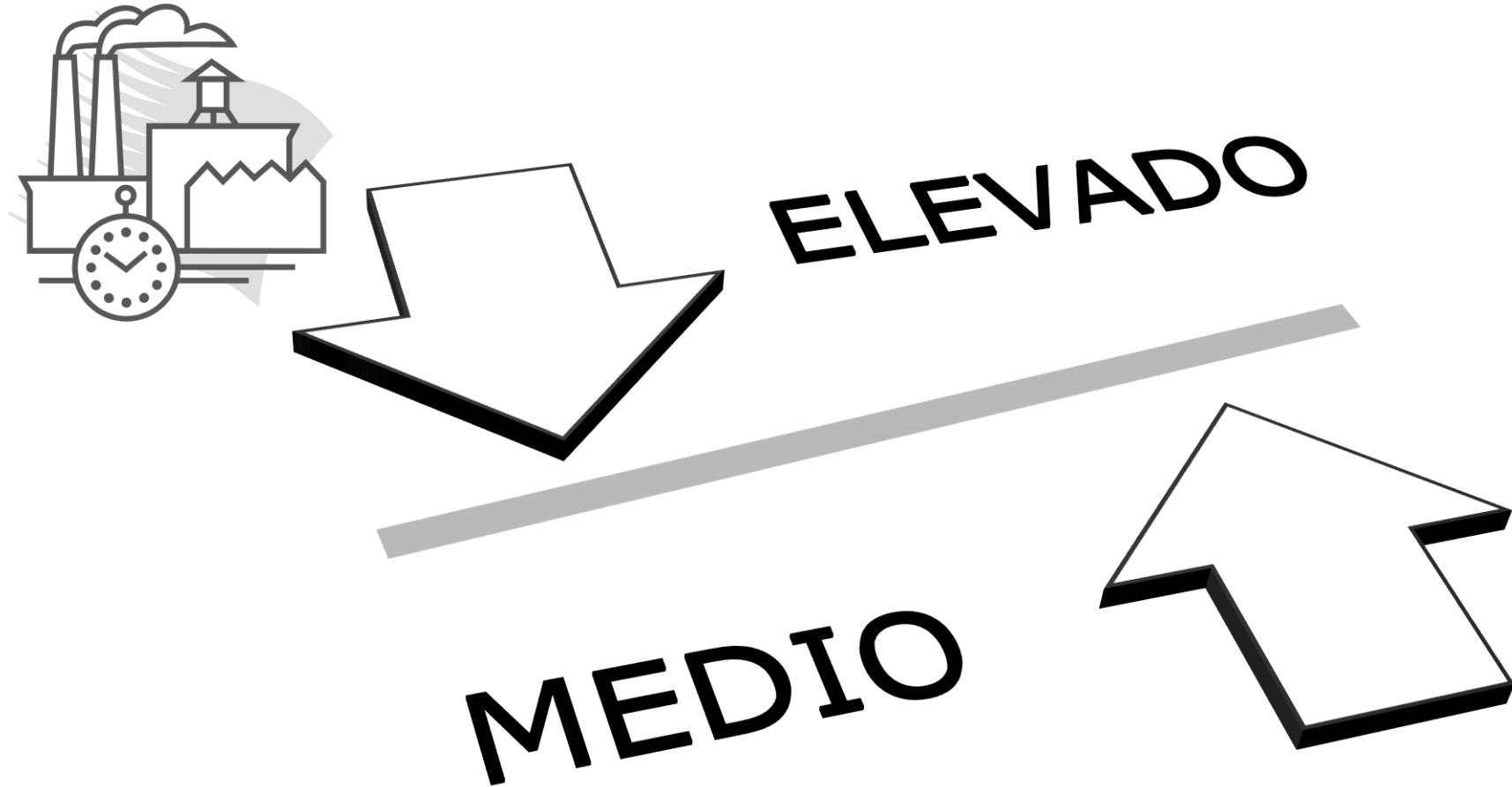
Tamaño del equipo



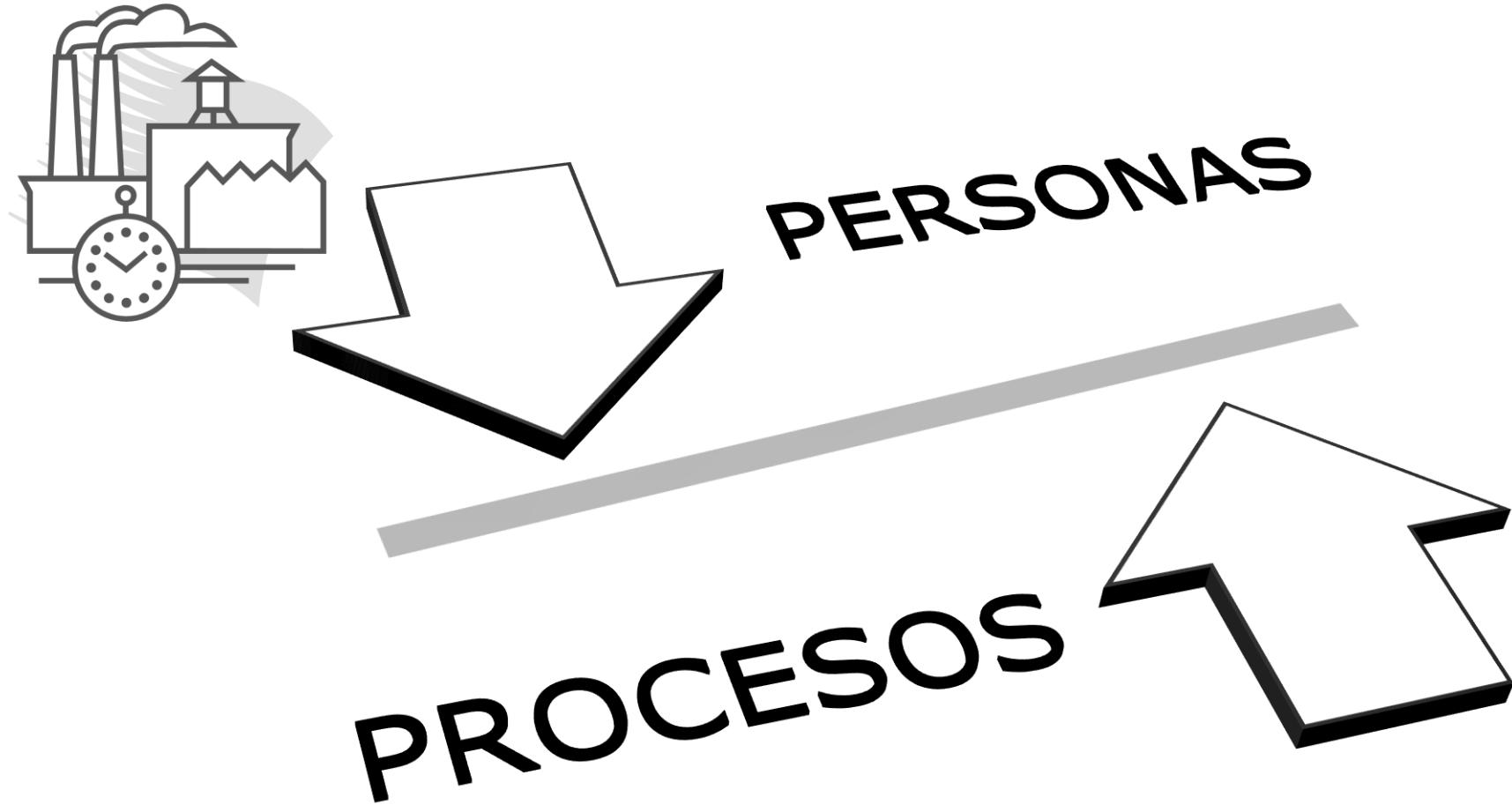
Cultura de la organización



Nivel profesional



Modelo de desarrollo



Gestión predictiva – gestión evolutiva

CARACTERÍSTICAS DE LA ORGANIZACIÓN



	EVOLUTIVA	PREDICTIVA
NIVEL PROFESIONAL	Senior	Junior
CULTURA ORGANIZATIVA	Horizontal, flexible	Vertical, rígida
MODELO DE DESARROLLO	Personas	Procesos

Gestión predictiva – gestión evolutiva

CARACTERÍSTICAS DEL PROYECTO

	EVOLUTIVA	PREDICTIVA
PRIORIDAD DE NEGOCIO	Valor	Cumplimiento
ESTABILIDAD DE REQUISITOS	Entorno inestable	Entorno estable
RIGIDEZ DEL PRODUCTO	Modificable	Difícil de modificar
COSTE DE PROTOTIPADO	Bajo	Alto
CRITICIDAD DEL SISTEMA	Baja	Alta
TAMAÑO DEL EQUIPO	Pequeño	Grande

Scrum



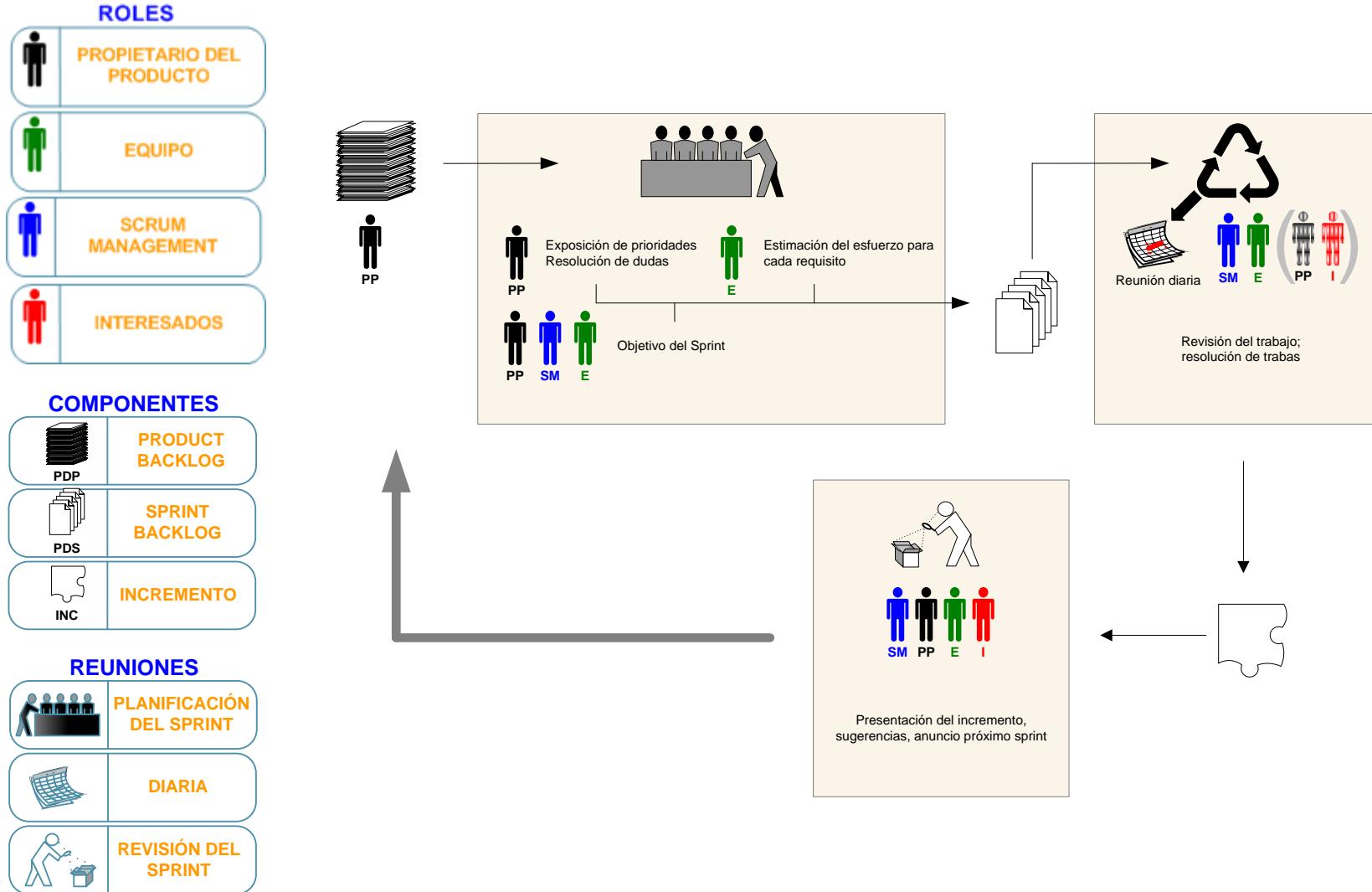
El origen de Scrum



"The New New Product Development Game"

Nonaka Takeuchy & Ikujiro Nonaka

Roles, componentes y reuniones

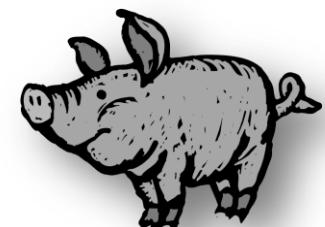


Roles: comprometidos e implicados



Una gallina y un cerdo paseaban por la carretera. La gallina dijo al cerdo: “Quieres abrir un restaurante conmigo”. El cerdo consideró la propuesta y respondió: “Sí, me gustaría. ¿Y cómo lo llamaríamos?”. La gallina respondió: “Huevos con beicon”.

El cerdo se detuvo, hizo una pausa y contestó: “Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada”.



Roles: comprometidos e implicados



IMPLICADOS

Usuarios, Marketing,
Comercial, Cliente,
Dirección...

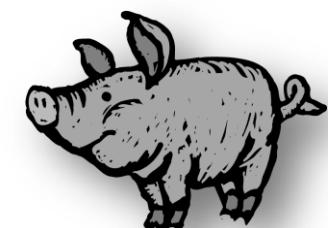


COMPROMETIDOS

Propietario de
producto



Equipo



Roles: comprometidos e implicados

● Propietario producto



Visión del producto

● Equipo



Auto-organizado
Multifuncional

● Interesados



Retro-information
Asesoría

● Scrum Manager



Garantía de funcionamiento
del modelo

Implantaciones flexibles

**LA ASIGNACIÓN DE
RESPONSABILIDADES
SE ADAPTA A LA
ORGANIZACIÓN**

Dirección

Calidad

RR.HH.

Oficina proyectos

Comercial

Programación

Roles



Propietario del producto

Representa a todos los interesados en el producto final.
Sus áreas de responsabilidad son:

- **Financiación del proyecto**
- **Funcionalidad del sistema**
- **Retorno de la inversión del proyecto**
- **Lanzamiento del proyecto**

Roles



Equipo

Responsable de transformar la pila de la iteración en un incremento de la funcionalidad del software

- **Auto-gestionado**
- **Auto-organizado**
- **Multifuncional**

Roles



Interesados

Cliente, usuarios, marketing,
comercial, dirección...

- **Retro-information**
- **Asesoría, sugerencias**
- **Colaboración**

Roles



Scrum Master

Como rol

Marco de scrum estándar o académico

Recomendable en organizaciones y equipos que comienzan a trabajar con scrum.

Como responsabilidad

En implantaciones flexibles o avanzadas de Scrum

Recomendable en organizaciones y equipos conocedores y experimentados con scrum

Roles



Scrum Master

- **Formador y entrenamiento del proceso**
- **Introducción de Scrum en la cultura de la empresa**
- **Garantía de cumplimiento de roles y formas del modelo**
- **Coaching de las personas**

Objetivo: EL EQUIPO

Roles



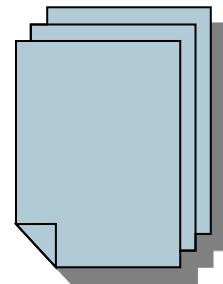
Rol Scrum Master

- **En primeras implantaciones**
- **En equipos con rotación de personas**

- **Asesoría para product backlog al propietario del producto**
- **Moderación de reuniones, asesoría estimación, velocidad...**
- **Gestión de personas para trabajo colaborativo**
- **Resolución de impedimentos**
- **Vigilar las formas scrum: objetivos de las reuniones, revisiones diarias, compartir la visión, etc.**
- **Asesoría y formación sobre el modelo y forma de trabajo**
- **Agendas de sprints, y respeto a las pautas del modelo...**

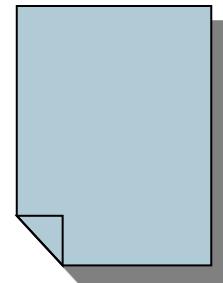
Componentes

■ Product Backlog



Lista de funcionalidades del sistema

■ Sprint Backlog



Tareas que se van a realizar en el sprint

■ Incremento



Parte del producto desarrollada en un sprint

Reuniones



Planificación del sprint



¿Cuánto va a durar el sprint y qué vamos a hacer?



Diaria



Seguimiento diario del avance



Revisión del sprint



Revisión del incremento realizado en el sprint

Práctica: Simulación de un ciclo scrum



cc-by: [LizMarie](#)

Práctica

 Product Backlog

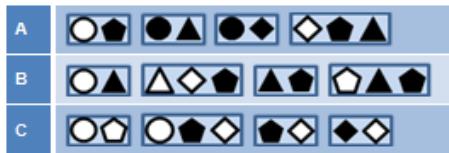
Funcionalidades priorizadas

Nº de pentágonos	Valor de negocio
1	*
2	**
3	***
4	****



ID	Funcionalidades	Nº	Valor.
A		2	**
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			

Práctica



Premisas para desarrollo ágil

- El “sistema” que necesita el cliente se puede descomponer en funcionalidades
- Tiene sentido y valor para el negocio del cliente poder disponer de ellas de forma parcial y cuanto antes

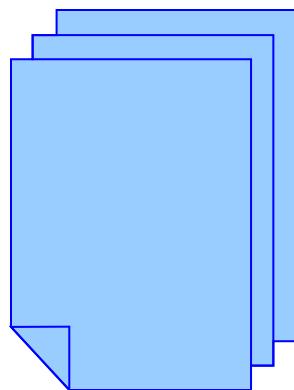
Primera acción del cliente

- Definir la visión de lo que quiere conseguir
- Identificar las primeras necesidades (Product Backlog)
- Priorizarlas según el valor para su entorno de negocio

Pila de producto (product backlog)

Los requisitos del sistema en un desarrollo ágil están en continua revisión.

El backlog las lista como historias de usuario priorizadas por las necesidades de negocio del cliente



- Lista de funcionalidades¹ del sistema
- Priorizadas
- Documento “vivo”
- Accesible a todos los roles
- Todos pueden contribuir y aportar elementos
- Es propiedad del responsable del producto (product manager, product owner, cliente, marketing, etc.)

(1) Idealmente. En realidad, todo lo que represente trabajo para desarrollar el producto

Requisitos con gestión predictiva y evolutiva



Predictivo



Evolutivo

SISTEMA

Requisitos del sistema

ISO/IEC 12207

ConOps

IEEE 1362



SOFTWARE

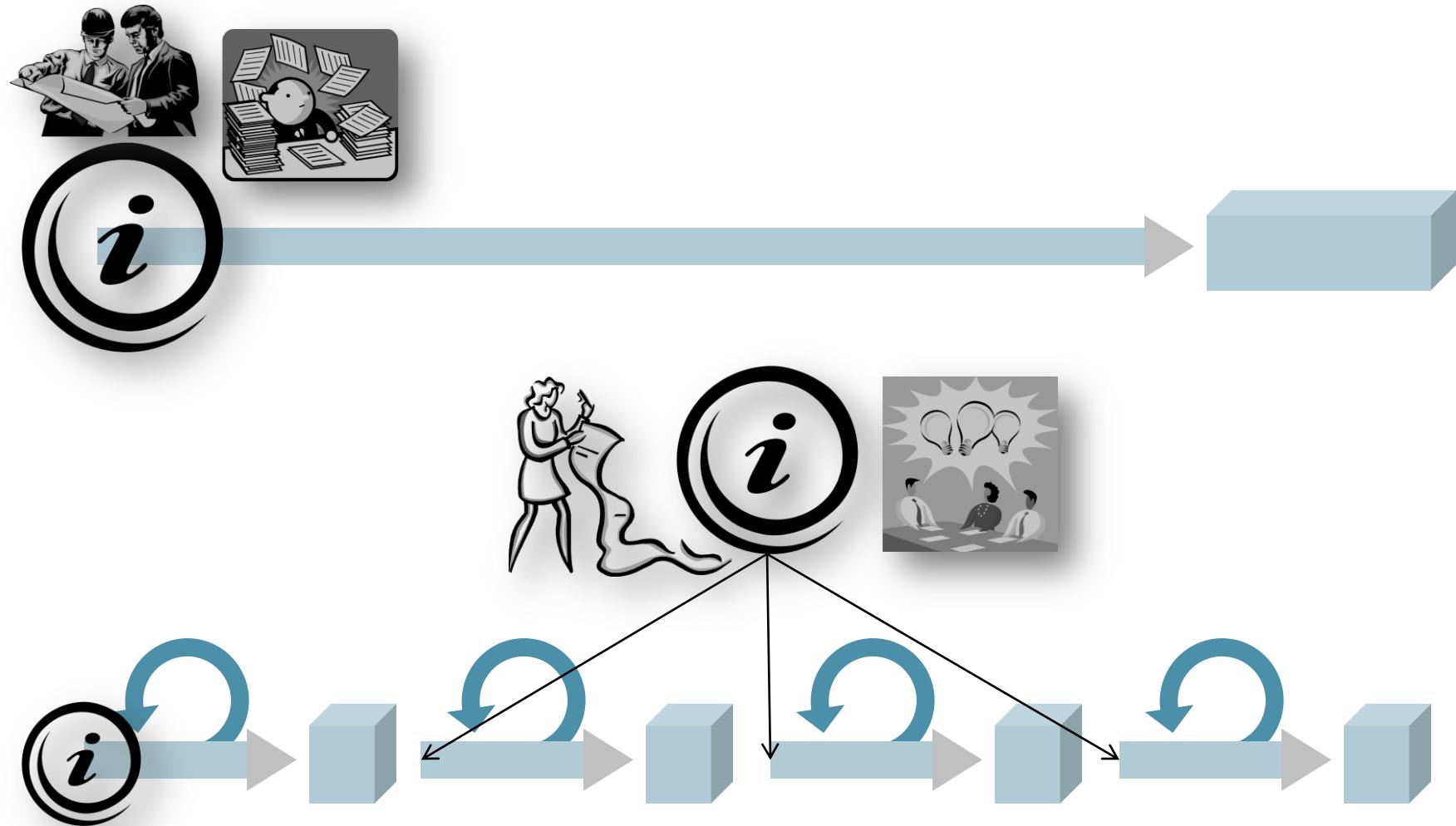
SRS

IEEE 830

Visión del producto
Product Backlog
Historias de usuario

Objetivo del sprint
Historias de usuario
Sprint Backlog

Requisitos con gestión predictiva y evolutiva



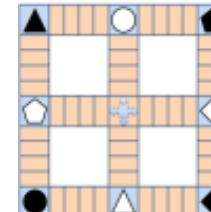
Ejercicio



Product Backlog

Funcionalidades priorizadas

Funcionalidades estimadas



ID	Funcionalidades	Esfuerzo	Valor.
A	○●○△○●○◆○◆○●○△○◆		
B	○●○△○◆○●○△○●○△○●○△		
C	○●○△○●○◆○●○◆○●○◆		
D	○●○△○●○◆○●○◆○●○△○●○△		
E	○●○△○●○◆○●○◆○●○◆○●○△○●○△○●○△		
F	○●○●○△○●○◆○●○△○●○●○△○●○△	90	
G	○●○●○△○●○◆○●○△○●○◆○●○●○◆	85	
H	○●○●○△○●○◆○●○◆○●○●○◆○●○●○◆	85	
I	○●○●○△○●○◆○●○●○◆○●○●○◆	80	
J	○●○●○△○●○◆○●○●○△○●○●○△○●○●○△	80	
K	○●○●○△○●○◆○●○●○◆○●○●○△○●○●○△	105	

Ejercicio



Product Backlog



Características del product backlog

- **Funcionalidades priorizadas**
- **Estimadas**
- **Documento “vivo”**

Se trata de determinar

- **Cómo de valiosa o urgente es cada funcionalidad**
- **Cómo de costosa es cada funcionalidad**



Y tener los criterios de decisión para...

La reunión de planificación de sprint

Ejercicio



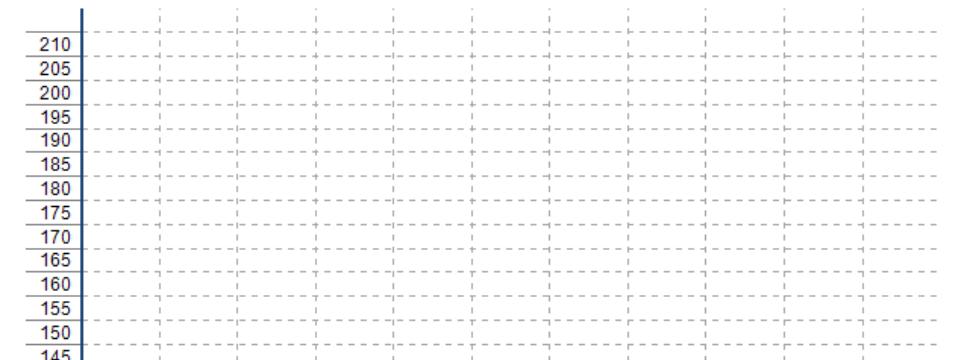
Planificación del sprint



- 1.- Calcular la velocidad del equipo
 - 2.- Funcionalidades prioritarias del cliente
 - 3.- Descomposición en tareas, estimación y asignación
 - 4.- Plan del sprint y Sprint Backlog

4.- Seguimiento del Sprint Backlog

5.- Gráfico Burn Down



Ejercicio



Planificación del sprint



AUTOGESTIÓN DEL EQUIPO

- **Cliente y programadores determinan cuánto tiempo va a durar el Sprint, y qué van a hacer**
- **Descomponen, estiman y asignan el trabajo (no un gestor de proyectos)**

Ejercicio



- 1.- Revisión individual de la(s) tareas con las que está cada uno
- 2.- Estimación del trabajo pendiente
- 3.- Dibujo del gráfico de avance (Burn Down)

4.- Seguimiento del Sprint Backlog

Nombre	Tarea nº	Pendiente (tiradas)									
		0	1	2	3	4	5	6	7	8	9

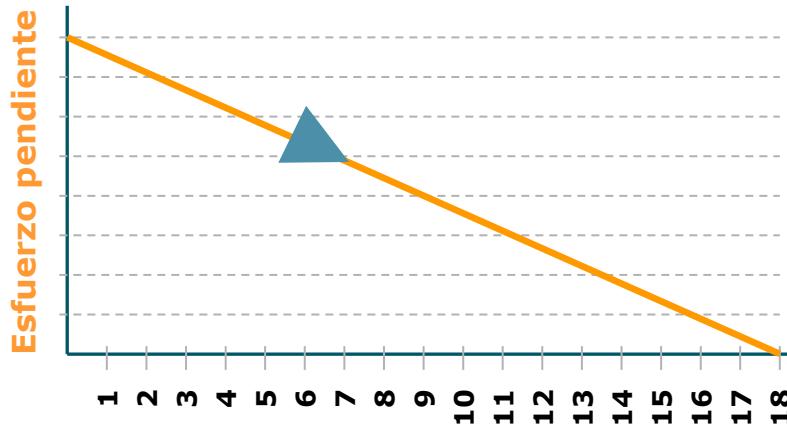
Plan del el Sprint (Sprint Backlog)

Nº	TAREA DEL SPRINT TAREA	ESF	ID P. BACKLOG	ASIGNADA A	ESTIMADO (Tiradas)
1	○▲		B		3

Ejercicio

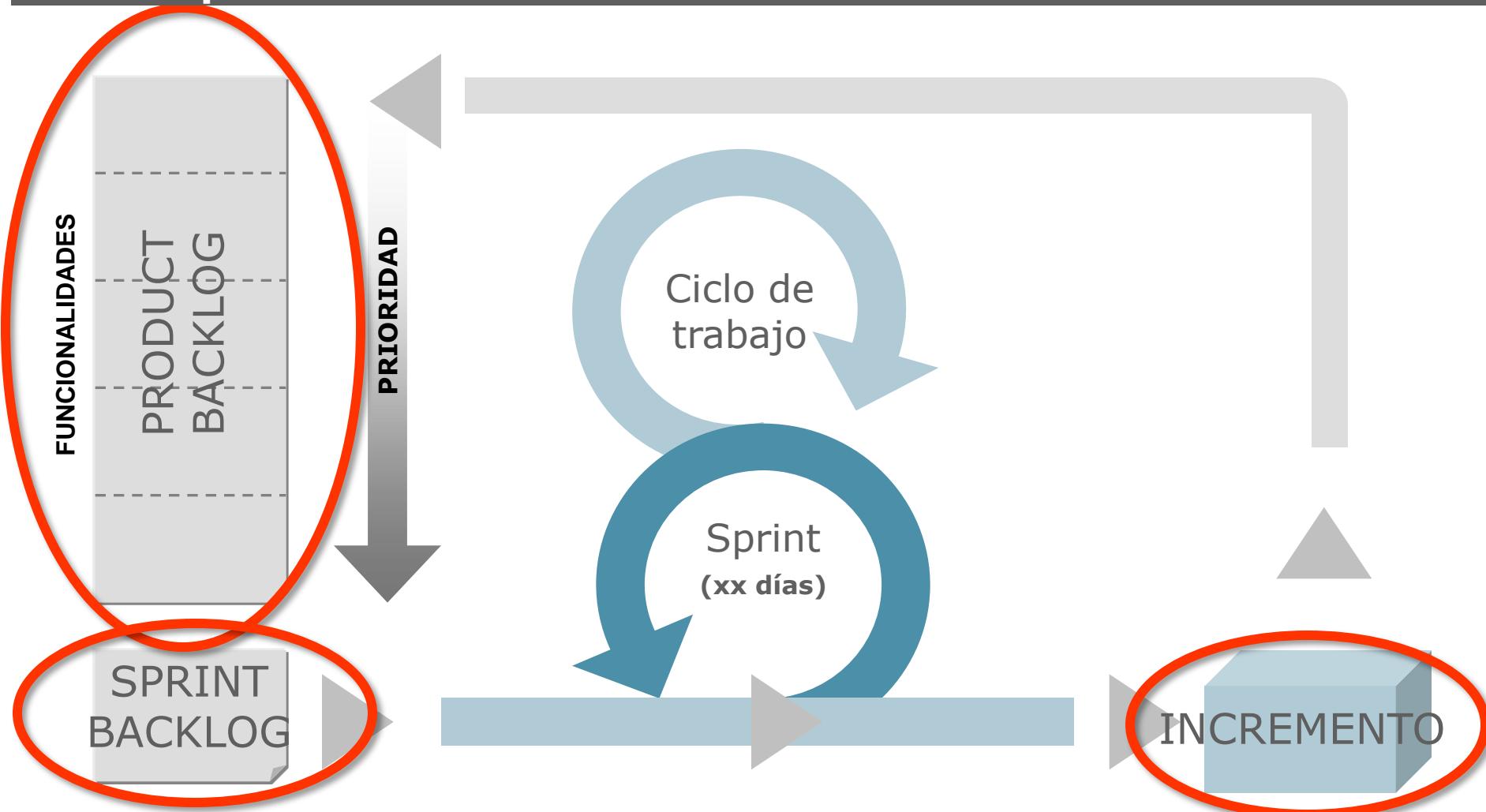


Sprint

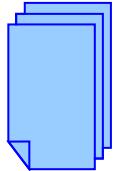


- Seguimiento diario
- Gráfica Burn Down como indicador de progreso
- El avance se mide por lo que falta, no por lo que se ha hecho

Los componentes de scrum



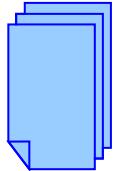
Pila de producto



Ejemplo de formato

Id	Prioridad	Descripción	Est.	Por
1	Muy alta	Plataforma tecnológica	30	AR
2	Muy alta	Interfaz usuario	40	LR
3	Muy alta	Un usuario se registra en el sistema	40	LR
4	Alta	El operador define el flujo y textos de un expediente	60	AR
5	Alta	Etc...	999	XX

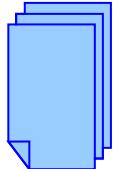
Pila de producto



Ejemplo de formato

Id	Prioridad	Módulo	Descripción	Est.	Por
1	Muy alta		Plataforma tecnológica	30	JM
2	Muy alta		Interfaz de usuario	40	LR
3	Muy alta		Un usuario se registra en el sistema	40	LR
4	Alta	Trastienda	El operador define el flujo y textos de un expediente	60	JM
5	Alta	Trastienda	Etc...	999.	XX

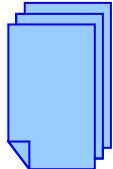
Pila de producto



Ejemplo de formato

Id	Orden	Est.	Descripción	Criterio validación	Obs.
1	10	30	Plataforma tecnológica	Se tiene el diagrama de la arquitectura, validado por xxx	La arquitectura debe permitir escalabilidad por clusterización de
2	20	40	Prototipos interfaz usuario	Todas las pantallas de interfaz están dibujadas y se puede recorrer toda la func	Debe estar interfaz para las funcionalidades de la pila a fecha
3	30	40	Diseño de datos	Diagrama BB.DD. Realizado, validado por xxx	
4	40	60	El operador define el flujo y textos de un expediente	Definir completamente un expediente con la funcionalidad programada	
5	50	999.	Etc...	Etc...	

Pila de producto



Ejemplo de formato

Id	Módulo	Descripción	Est.	Por
Crítico				
1		Plataforma tecnológica	30	AR
2	Cliente	Interfaz de usuario	40	LR
3	Cliente	Un usuario se registra en el sistema	40	LR
4	Trastienda	El operador define el flujo y textos de un expediente	60	AR
5	Trastienda	Etc...	999.	XX
Necesario				
6	Cliente	El usuario modifica su ficha personal	30	AR
7	Cliente	El usuario consulta los expedientes asignados	15	LR
8	Cliente	El usuario tramita un expediente	35	LR

Pila de producto

IMPRESCINDIBLE

SEGÚN PROYECTO

Id

Obs.

Orden / prioridad

Asignado

Descripción

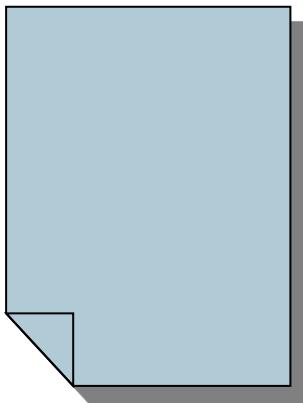
Nº Sprint

Criterio validación

Etc.

Estimación

Pila del sprint



Pila del sprint

- Funcionalidades que se van a realizar en el sprint
- Comprometidas por el equipo
- Asignadas
- Estimadas

Pila del sprint

SPRINT	INICIO	DURACIÓN	ESFUERZO												
1	1-mar-07	12	J	V	L	M	X	J	V	L	M	J	V	L	
	1-mar	2-mar	5-mar	6-mar	7-mar	8-mar	9-mar	12-mar	13-mar	15-mar	16-mar	19-mar			
	23	23	19	16	16	13	9	9	9	9	9	9	9	9	
	276	246	216	190	178	158	110	110	110	110	110	110	110	110	
SPRINT BACKLOG			ESFUERZO												
Tarea	Estado	Responsable	16	16	16	16	16	16	12	12	12	12	12	12	
Descripción de la tarea 1	Terminada	Luis	16	16	16	16	16	16	12	12	12	12	12	12	
Descripción de la tarea 2	Terminada	Luis	12	8					8	8	8	8	8	8	
Descripción de la tarea 3	Terminada	Luis	4	4	4	4	4	4							
Descripción de la tarea 4	Terminada	Elena	8	4											
Descripción de la tarea 5	Terminada	Elena	16	16	4										
Descripción de la tarea 6	Terminada	Elena	6	6	2										
Descripción de la tarea 7	Terminada	Antonio	16	4											
Descripción de la tarea 8	Terminada	Antonio	16	16	20	12	4								
Descripción de la tarea 9	Terminada	Antonio	12	2											
Descripción de la tarea 10	En curso	Luis	12	12	12	12	12	12	12	12	12	12	12	12	
Descripción de la tarea 11	Pendiente	Luis	8	8	8	8	8	8	8	8	8	8	8	8	
Descripción de la tarea 12	Terminada	Luis	14	14	14	14	14	14	14						
Descripción de la tarea 13	En curso	Antonio	8	8	8	8	8	8	6						
Descripción de la tarea 14	Pendiente	Antonio	16	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 15	Pendiente	Antonio	16	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 16	Terminada	Elena	8	8	8										
Descripción de la tarea 17	Terminada	Elena	12	12	12	8	4								
Descripción de la tarea 18	En curso	Elena	16	16	16	16	16	10	10	10	10	10	10	10	
Descripción de la tarea 19	Terminada	Elena	12	12	12	12	12	12							
Descripción de la tarea 20	Pendiente	Elena	16	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 21	Pendiente	Elena	12	12	12	12	12	12	12	12	12	12	12	12	
Descripción de la tarea 22	Pendiente	Antonio	8	8	8	8	8	8	8	8	8	8	8	8	
Descripción de la tarea 23	Pendiente	Antonio	12	12	12	12	12	12	12	12	12	12	12	12	

Incremento



Entregable

iTerminada!

Implementada

Probada

Documentada

Pila del sprint

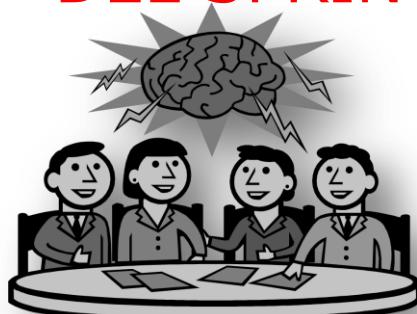


Incremento

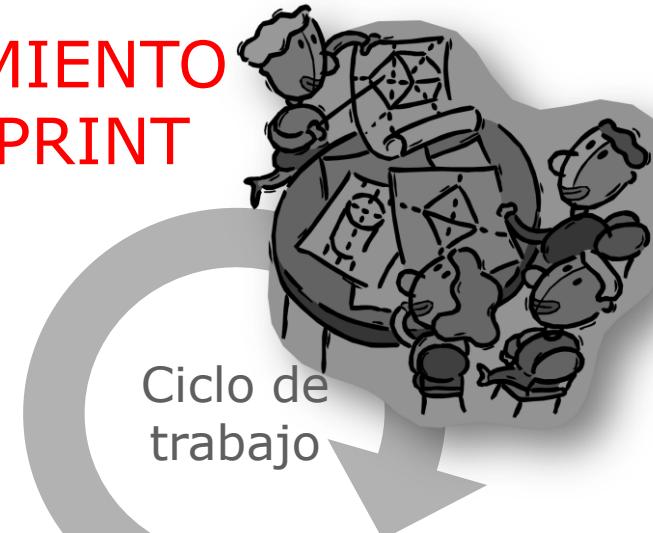
- Parte del producto desarrollada en un sprint
- En condiciones de ser usada
- Es una funcionalidad

Las reuniones

PLANIFICACIÓN
DEL SPRINT



SEGUIMIENTO
DEL SPRINT



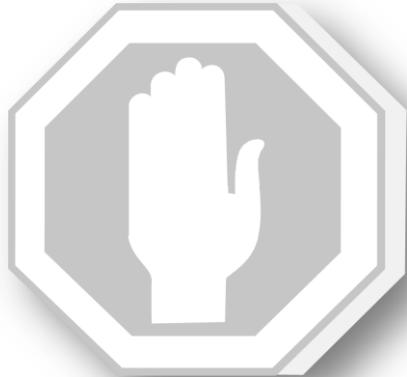
Sprint
(15 – 30 días)

REVISIÓN
DEL SPRINT



Incremento

Planificación del sprint



ANTES DE EMPEZAR

- Están determinados los recursos posibles
- Hay un product backlog (lista de producto) preparado y válido
- Propietario del producto y equipo trabajan juntos
- El equipo conoce las tecnologías empleadas y el negocio del producto

Planificación del sprint

PRODUCT BACKLOG

PRODUCTO DESARROLLADO

INFORMACIÓN ENTORNO



Planificación del sprint



Máximo: 1 día



Exposición
Product backlog



Resolución
Sprint backlog

Seguimiento del sprint (stand up)

Reunión del equipo con duración máxima de 15 minutos.

- Todos los días en el mismo sitio y a la misma hora.
- Se recomienda que sea la primera actividad del día.
- Deben acudir todos los miembros del equipo.
- Se recomienda estar de pie.
- Moderada por el Scrum Manager o Team Leader.

Las tres preguntas



- ¿Qué trabajo has realizado desde la última reunión?
- ¿Qué tienes previsto para hoy?
- ¿Qué necesitas?

Revisión del sprint

PRESENTACIÓN DEL EQUIPO AL PROPIETARIO DEL PRODUCTO Y TODOS LOS IMPLICADOS (GALLINAS) DEL INCREMENTO REALIZADO EN EL SPRINT

- Duración máxima: 4 horas.
- Se muestra el producto realizado: NO POWERPOINTS
- No es un acontecimiento social. NO DEBE REQUERIR NINGUNA PREPARACIÓN ESPECIAL
- Objetivos:
- Comprobar el trabajo realizado
- Retro-information para la evolución del Product Backlog
- No es una reunión para valoraciones y críticas.

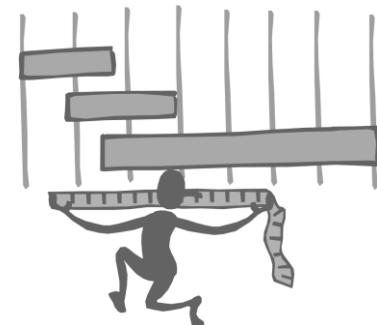
Métricas: fondos de escala

ORGANIZACIÓN



- Rendimiento
- Satisfacción clientes
- Eficiencia
- Desempeño
- ...

PROYECTO



- Esfuerzo
- Coste
- Tamaño
- ...

DESARROLLO



- Densidad de fallos
- Complejidad de diseño
- Disponibilidad

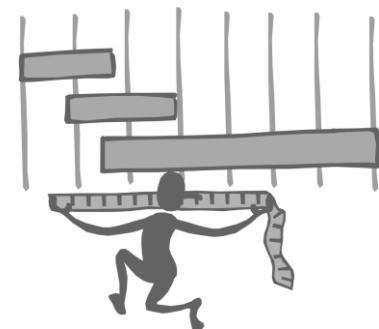
Métricas: fondos de escala

ORGANIZACIÓN



- Rendimiento
- Satisfacción clientes
- Eficiencia
- Desempeño
- ...

PROYECTO



- Esfuerzo
- Coste
- Tamaño
- ...

DESARROLLO



- Densidad de fallos
- Complejidad de diseño
- Disponibilidad

¿ Métrica ?



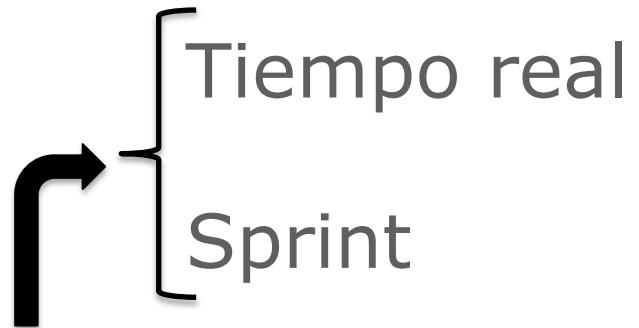
VALOR APORTADO

VALOR POR OMITIRLA

Magnitudes: velocidad – trabajo - tiempo

Velocidad = Trabajo / Tiempo

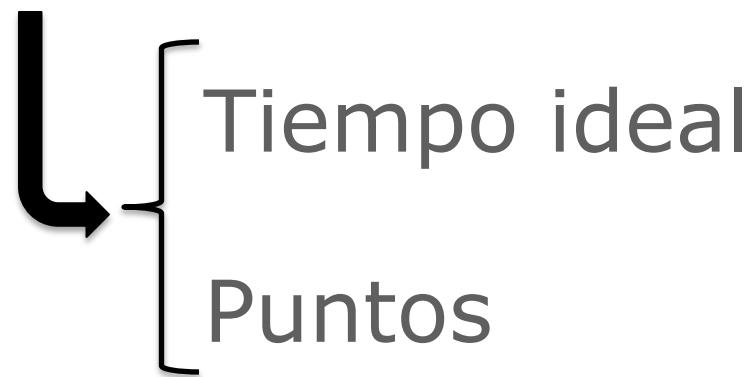
Unidades de tiempo



Velocidad = Trabajo / Tiempo

Unidades de trabajo

Velocidad = Trabajo / Tiempo



Unidades de trabajo: tiempo ideal

Tiempo ideal

¿Cuánto cuesta programar el registro de un nuevo usuario?

¿En tiempo ideal, o en tiempo real?



?



Unidades de trabajo: tiempo ideal

¿Cuánto dura un partido de Baloncesto?



Tiempo ideal: 40 minutos

Tiempo real: > 2 horas

Fotografía cc-by: SD
Dirk

Unidades de trabajo: tiempo ideal

Tiempo ideal

Sin interrupciones

Está disponible todo lo que se necesita

Estado mental óptimo

Unidades de trabajo: tiempo ideal

Puntos

“Cantidad” de trabajo considerando:

Dificultad

Tamaño

Unidad relativa

Unidad propia del equipo / organización

Unidades de trabajo: tiempo ideal



cc-by: [LizMarie](#)

Estimación de póquer: serie natural

0

0

0

 $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

1

1

1

2

2

2

3

3

3

5

5

5

6

6

9

7

7

L

 ∞ ∞ ∞

P

?

d

 $\frac{1}{2}$  $\frac{1}{2}$

Estimación de póquer: serie Fibonacci

0

0

0

 $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

1

1

1

2

2

2

3

3

3

5

5

5

8

8

8

13

13

13

21

21

21

 ∞ ∞ ∞

?

?

?

 $\frac{1}{2}$  $\frac{1}{2}$

Práctica de estimación de póquer



cc-by: [LizMarie](#)

Medición del trabajo



La gestión ágil NO determina el grado de avance del proyecto midiendo el trabajo realizado, sino el pendiente de realizar



Medición del trabajo

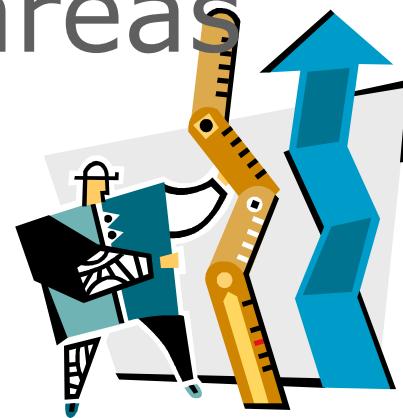
SCRUM MIDE EL TRABAJO PENDIENTE

A

B

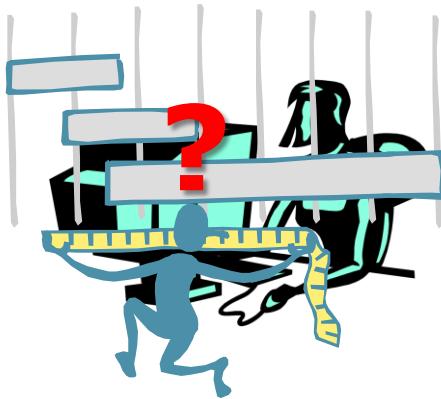


Para estimar esfuerzo y duración de las tareas



	1-feb	2-feb	3-feb	6-feb	7-feb	8-feb	9-feb
Tareas pendientes	12	12	11	9	8	6	5
Horas de trabajo pendientes	12	12	11	9	8	6	5
Estado	Terminada						
Responsable	Luis	Luis	Luis	Luis	Elena	Elena	Elena
24	24	20	14	8	8	5	5
16	16	16	16	10	10	5	5
8	8	8	8	5	5	5	5

¿Se puede estimar con precisión?



- Los requisitos no son realidades de solución única
- La capacidad y calidad de cada persona es diferente
- Los entornos de trabajo influyen en la productividad

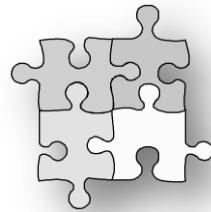
Estrategia de la estimación ágil



PRECISIÓN RELATIVA



JUICIO DE EXPERTOS



TAREAS DE TAMAÑO PEQUEÑO

Unidades ágiles

Velocidad



Trabajo
Por
Sprint

(Scrum académico)

Fotografía cc-by: fdecomite

Unidades ágiles

Velocidad



Fotografía cc-by: Luis Argerich

**Trabajo por
unidad de
tiempo**

(Scrum flexible)

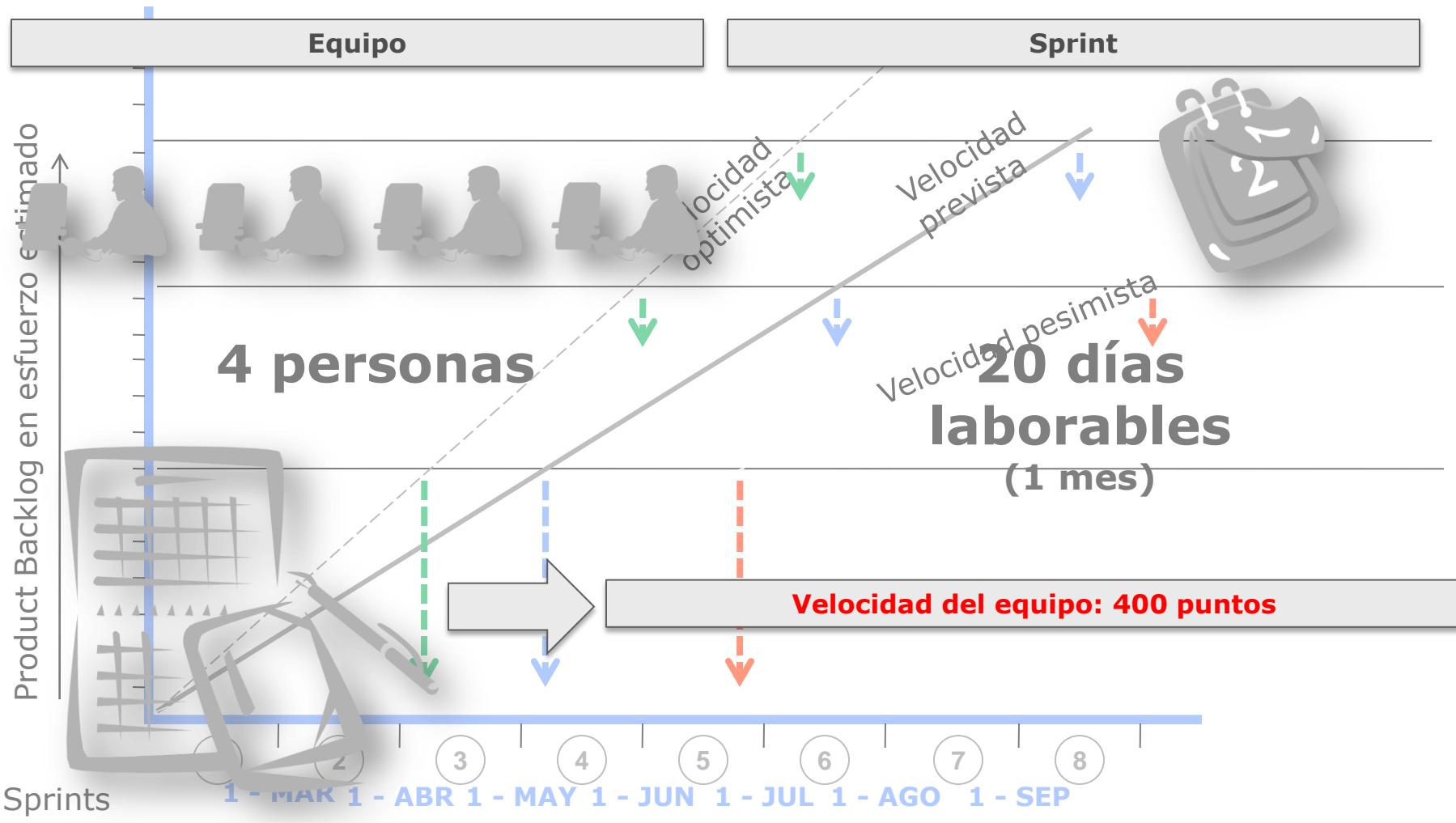
Product Backlog

Id	Historias	Trabajo	Criterio de validación
1	Historia A 1.0	150	Lorem ipsum dolor sit amet
2	Historia B 1.0	250	consectetuer adipiscing elit
3	Historia C 1.0	250	Aliquam vehicula accumsan tortor
4	Historia D 1.0	300	Pellentesque turpis
5	Historia A 1.1	250	Phasellus purus orci
6	Historia D 1.1	350	penatibus et magnis dis parturient
7	Historia E 1.0	150	Quisque volutpat ante sit amet velit
8	Historia B 1.1	500	Cras iaculis pede eu tellus
9	Historia C 1.1	150	Vestibulum vel diam sed pede blandit
10	Historia E 1.1	200	Suspendisse aliquam felis et turpis
11	Historia F 1.0	TBD	Nullam imperdiet lorem vitae justo
12	Historia A.1.2	TBD	Suspendisse potenti. In nec nunc
13	Historia B 1.2	TBD	Nam eros tellus, facilisis sed, pretium
14	Historia F 1.1	TBD	Morbi arcu tellus, condimentum

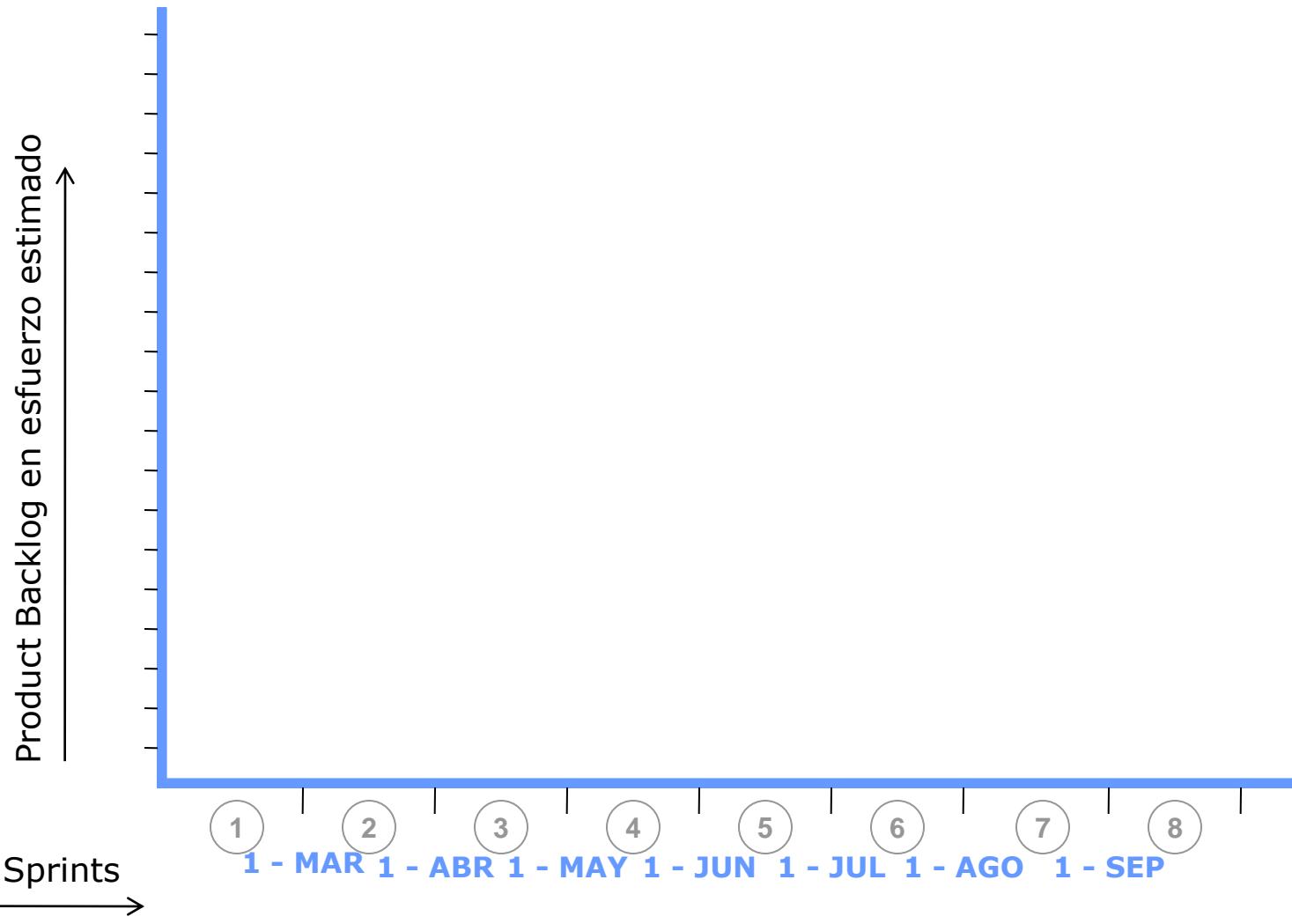
Plan de producto: Gráfico burn-up

Id	Historias	Trabajo	Criterio de validación	
1	Historia A 1.0	150	Lorem ipsum dolor sit amet consectetuer adipiscing elit	Estimación: 950 PUNTOS
2	Historia B 1.0	250		
3	Historia C 1.0	250	Aliquam vehicula accumsan tortor	
4	Historia D 1.0	300	Pellentesque turpis	Versión 1.0 →
5	Historia A 1.1	250	Phasellus purus orci	
6	Historia D 1.1	350	penatibus et magnis dis parturi	1.700 PUNTOS
7	Historia E 1.0	150	Quisque volutpat ante sit amet velit	Versión 1.1 →
8	Historia B 1.1	500	Cras iaculis pede eu tellus	
9	Historia C 1.1	150	Vestibulum vel diam sed pede b	2.550 PUNTOS
10	Historia E 1.1	200	Suspendisse aliquam felis et turpis	Versión 1.2 →
11	Historia F 1.0	TBD	Nullam imperdiet lorem vitae justo	
12	Historia A.1.2	TBD	Suspendisse potenti. In nec nunc	
13	Historia B 1.2	TBD	Nam eros tellus, facilisis sed, pretium	
14	Historia F 1.1	TBD	Morbi arcu tellus, condimentum	

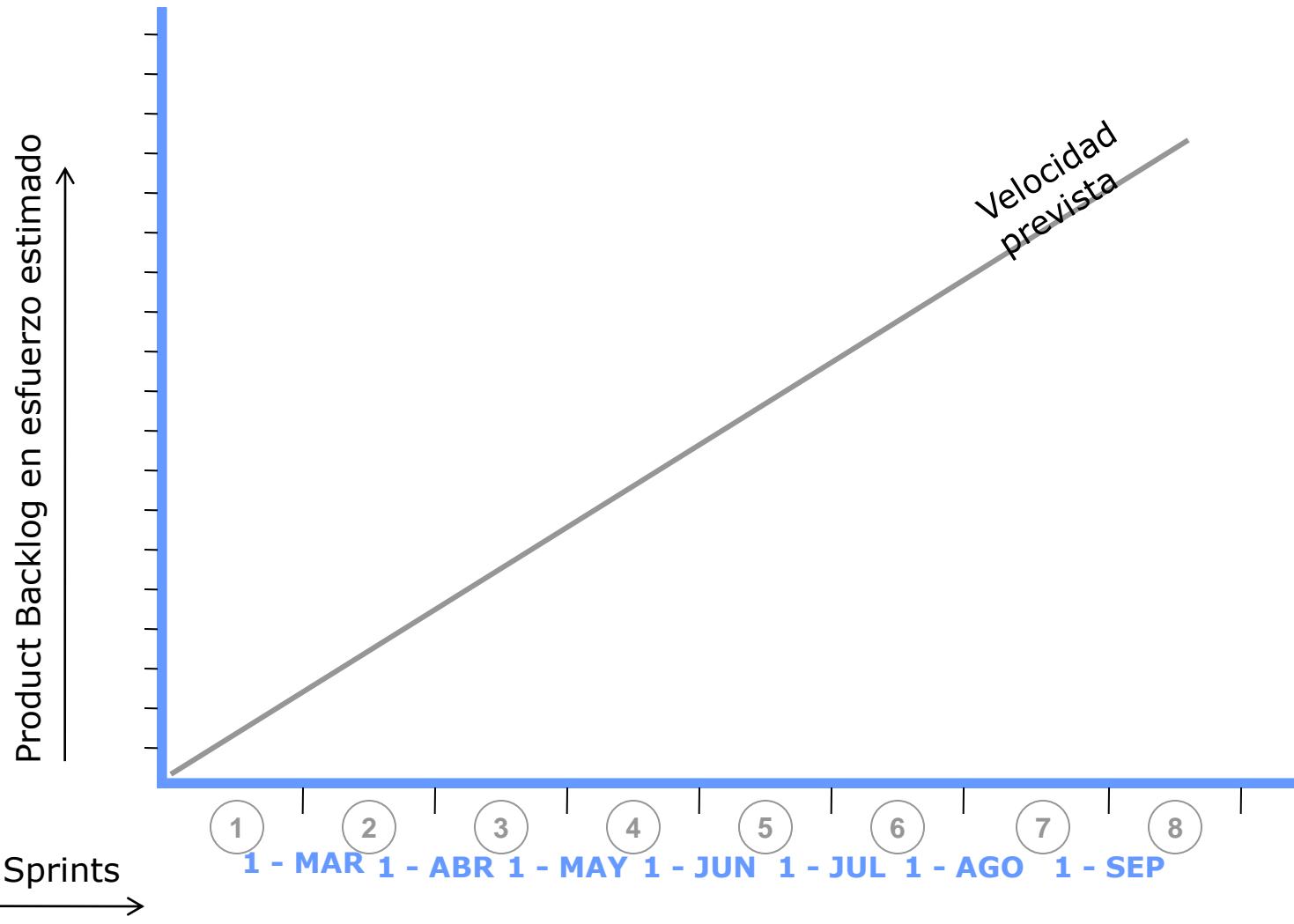
Plan de producto: Gráfico burn-up



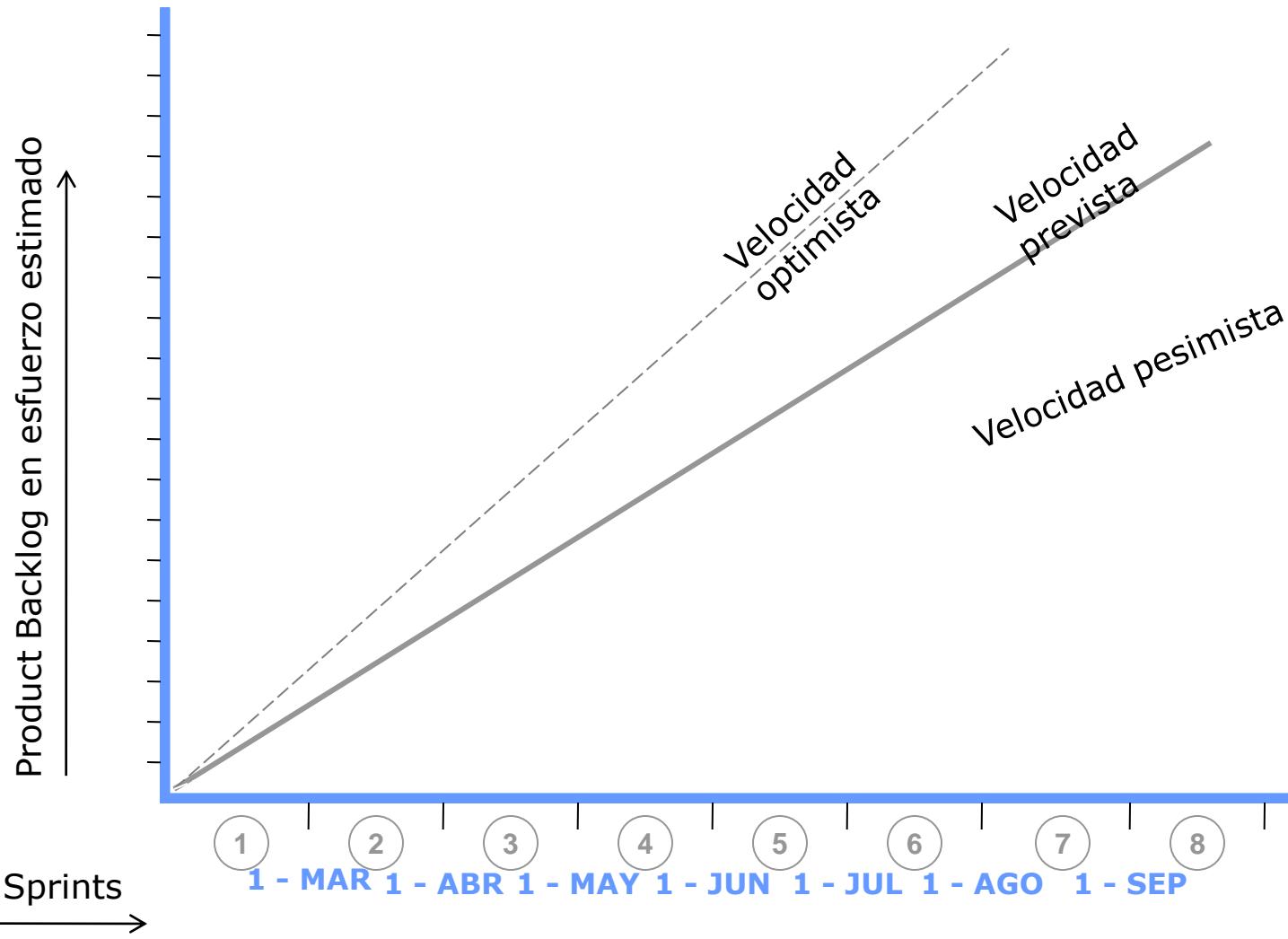
Plan de producto: Gráfico burn-up



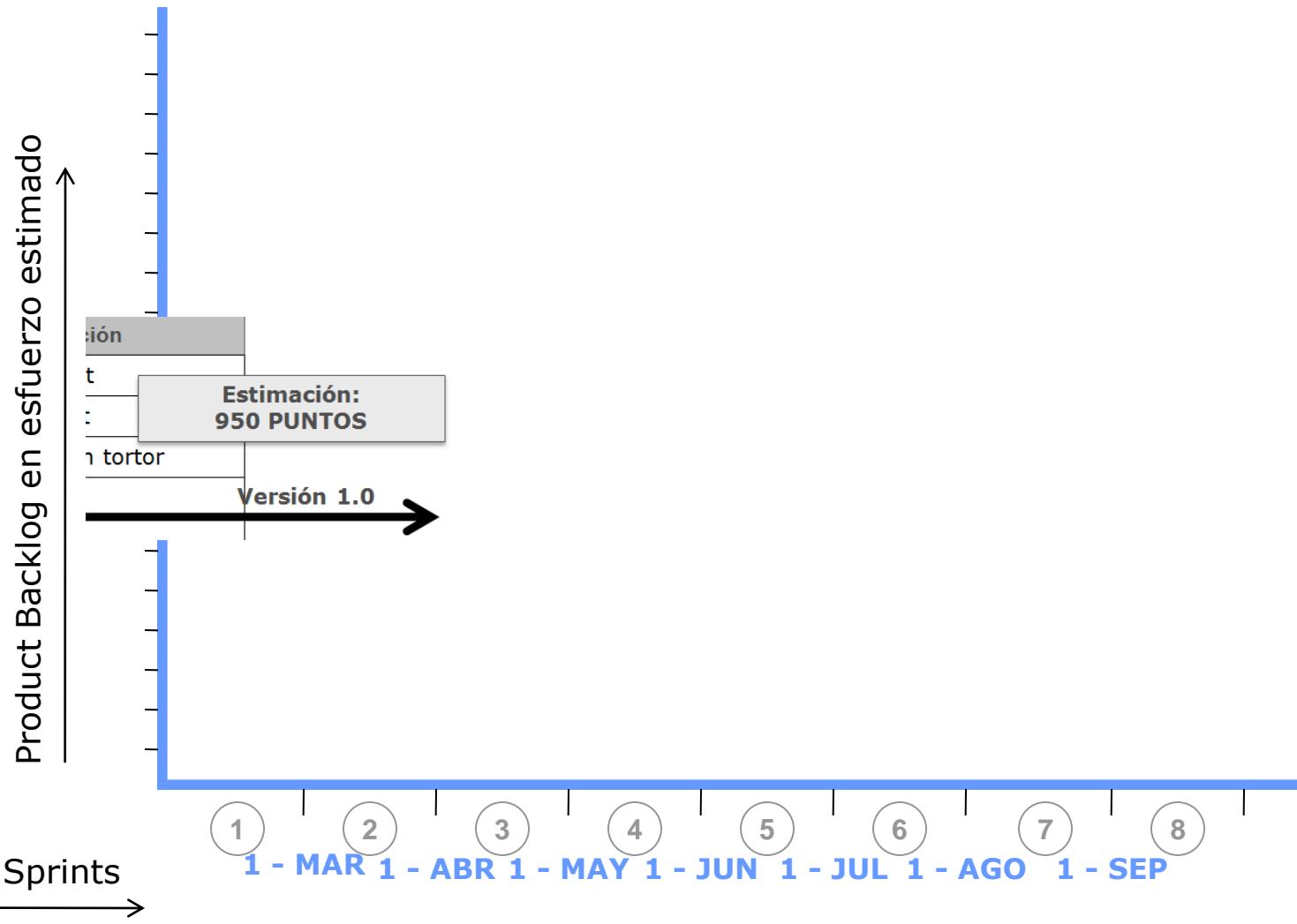
Plan de producto: Gráfico burn-up



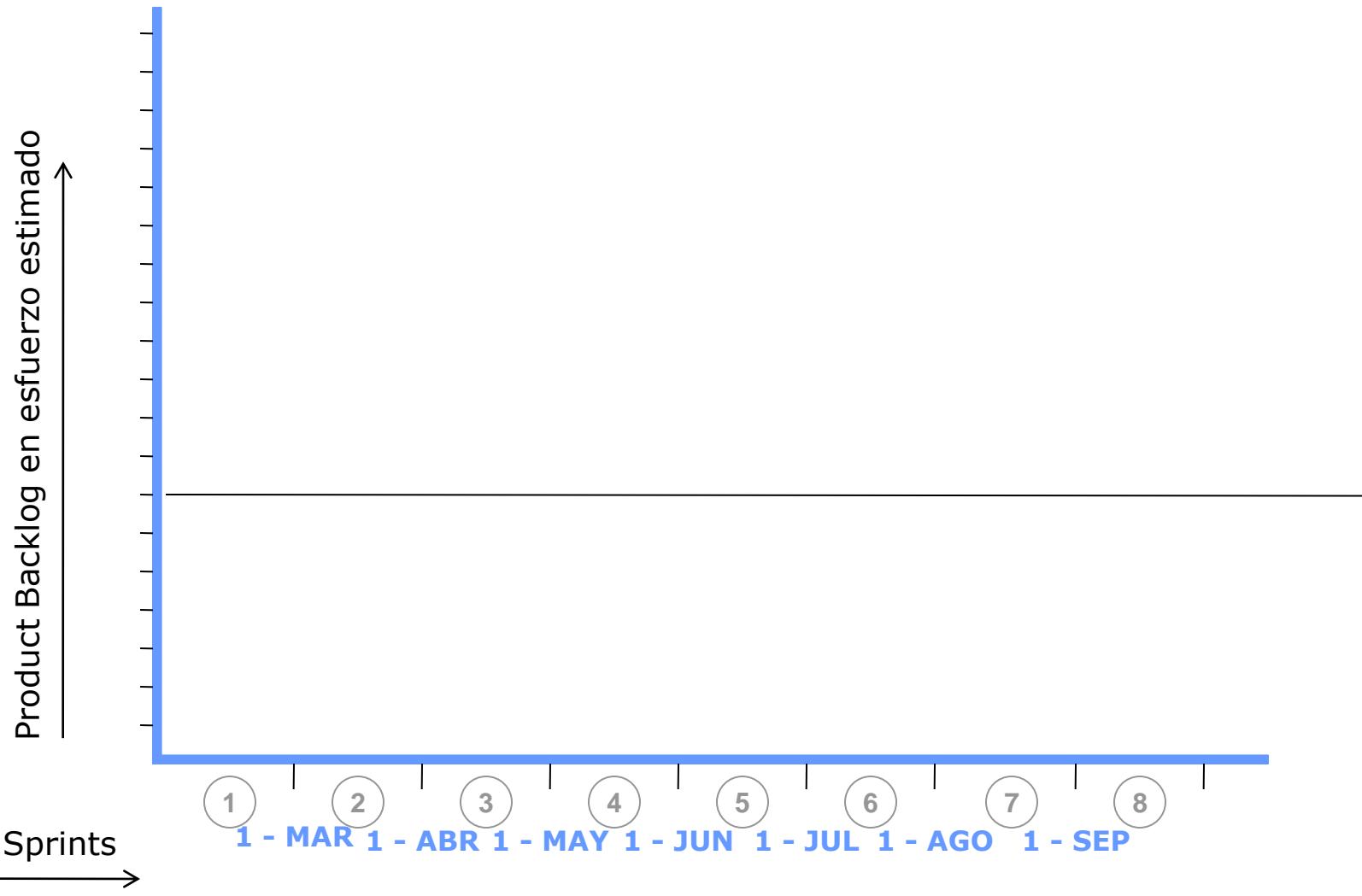
Plan de producto: Gráfico burn-up



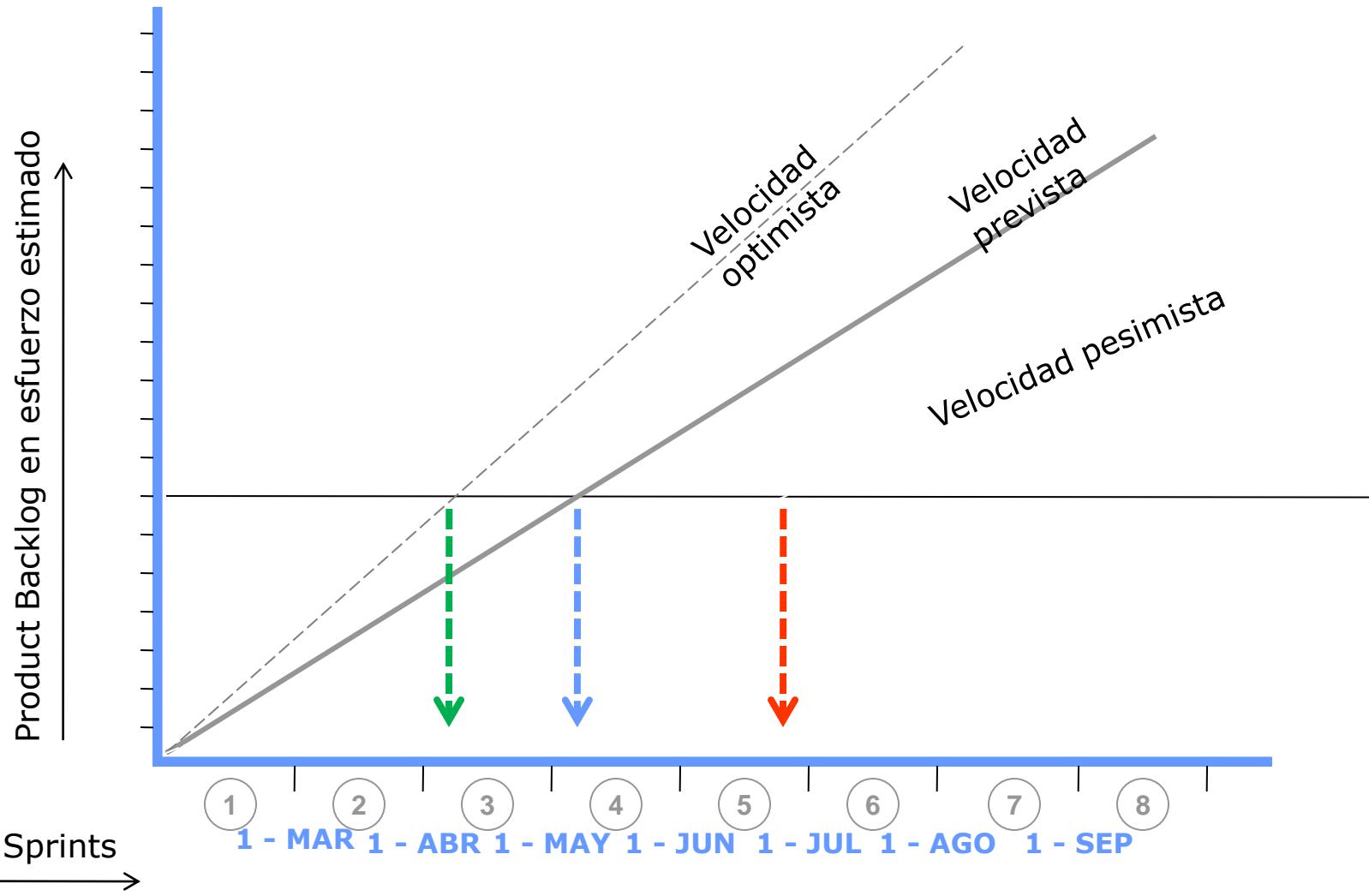
Plan de producto: Gráfico burn-up



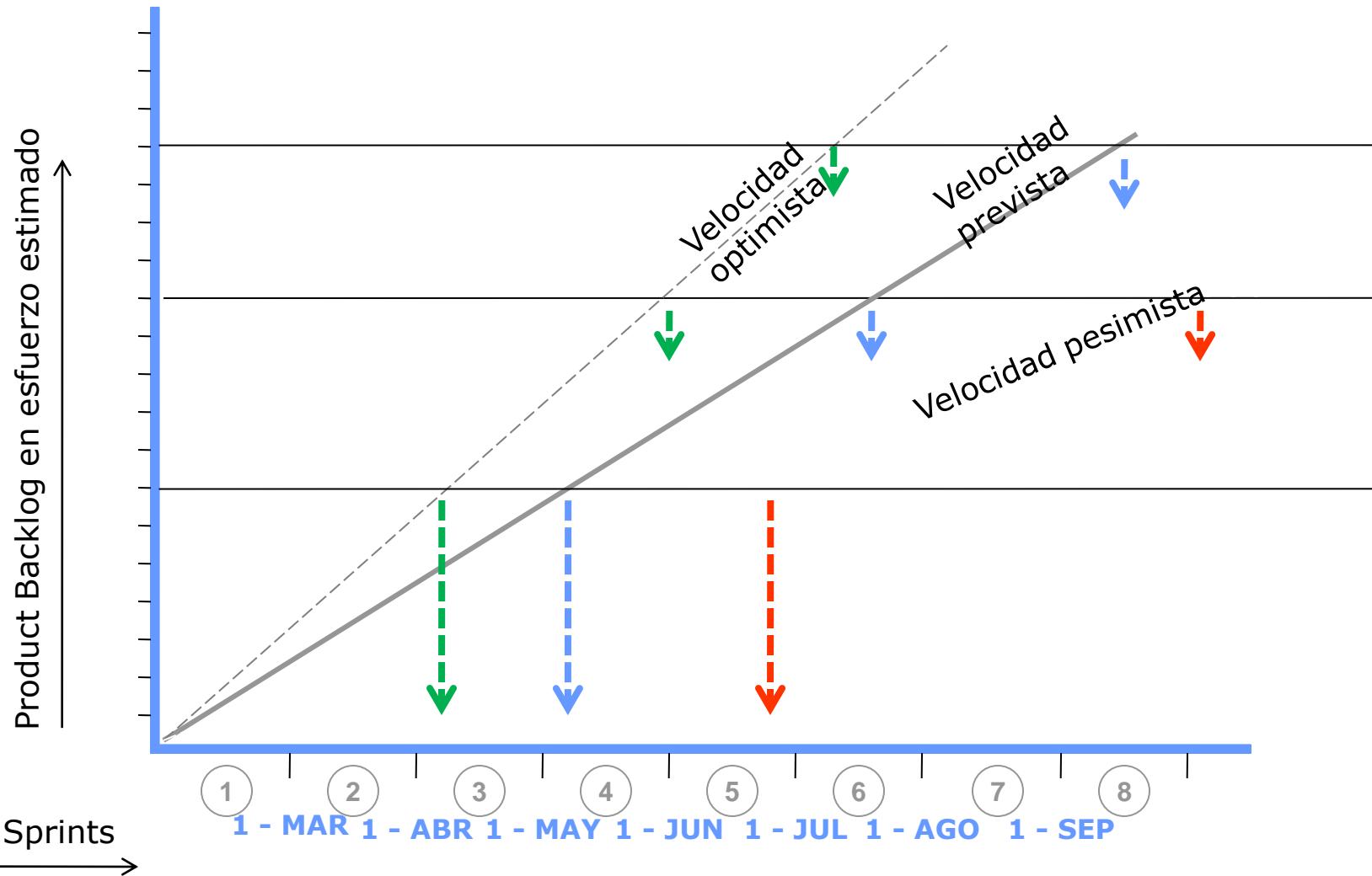
Plan de producto: Gráfico burn-up



Plan de producto: Gráfico burn-up



Plan de producto: Gráfico burn-up



Evolución del esfuerzo pendiente en la pila del sprint

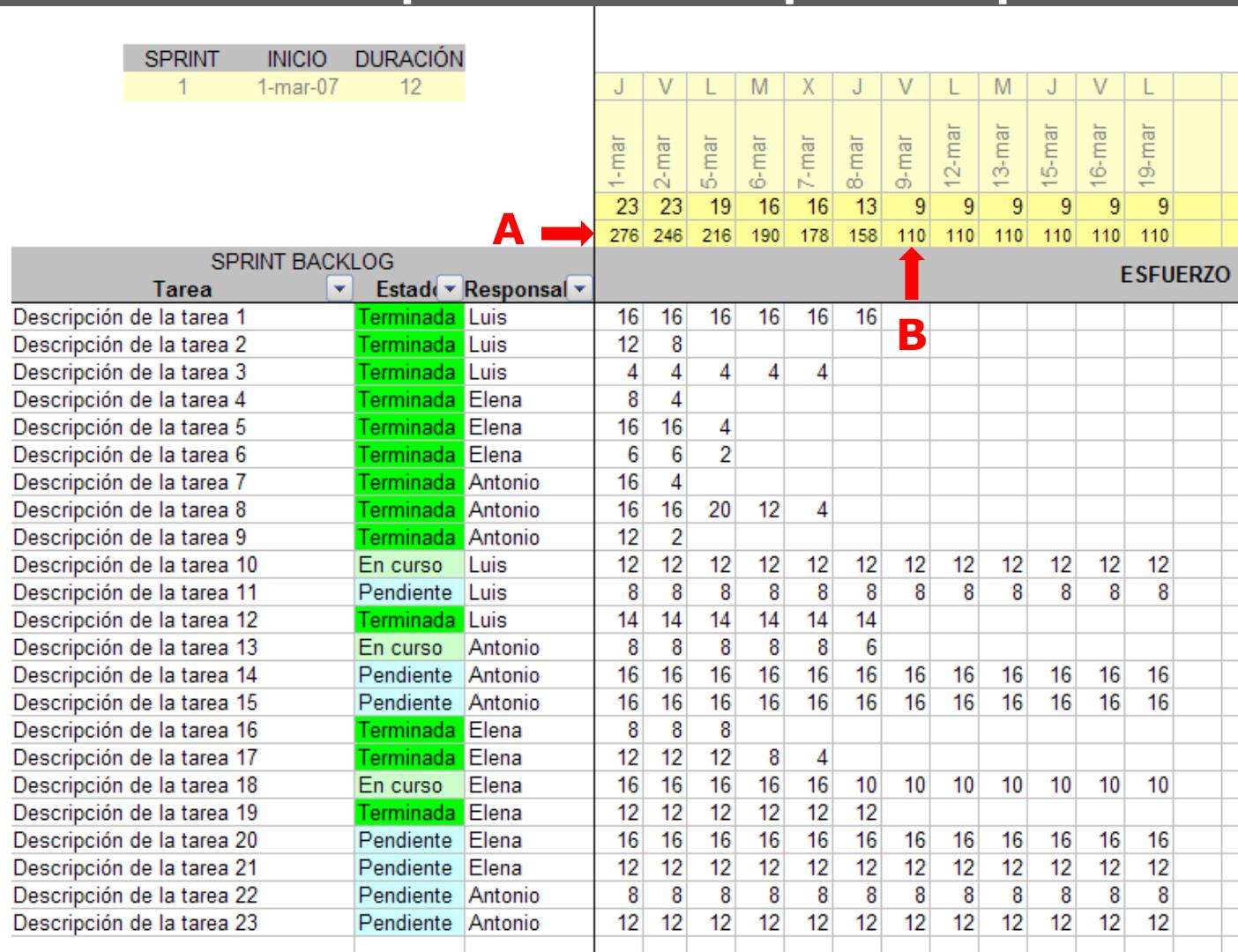


Gráfico de avance (burn down)

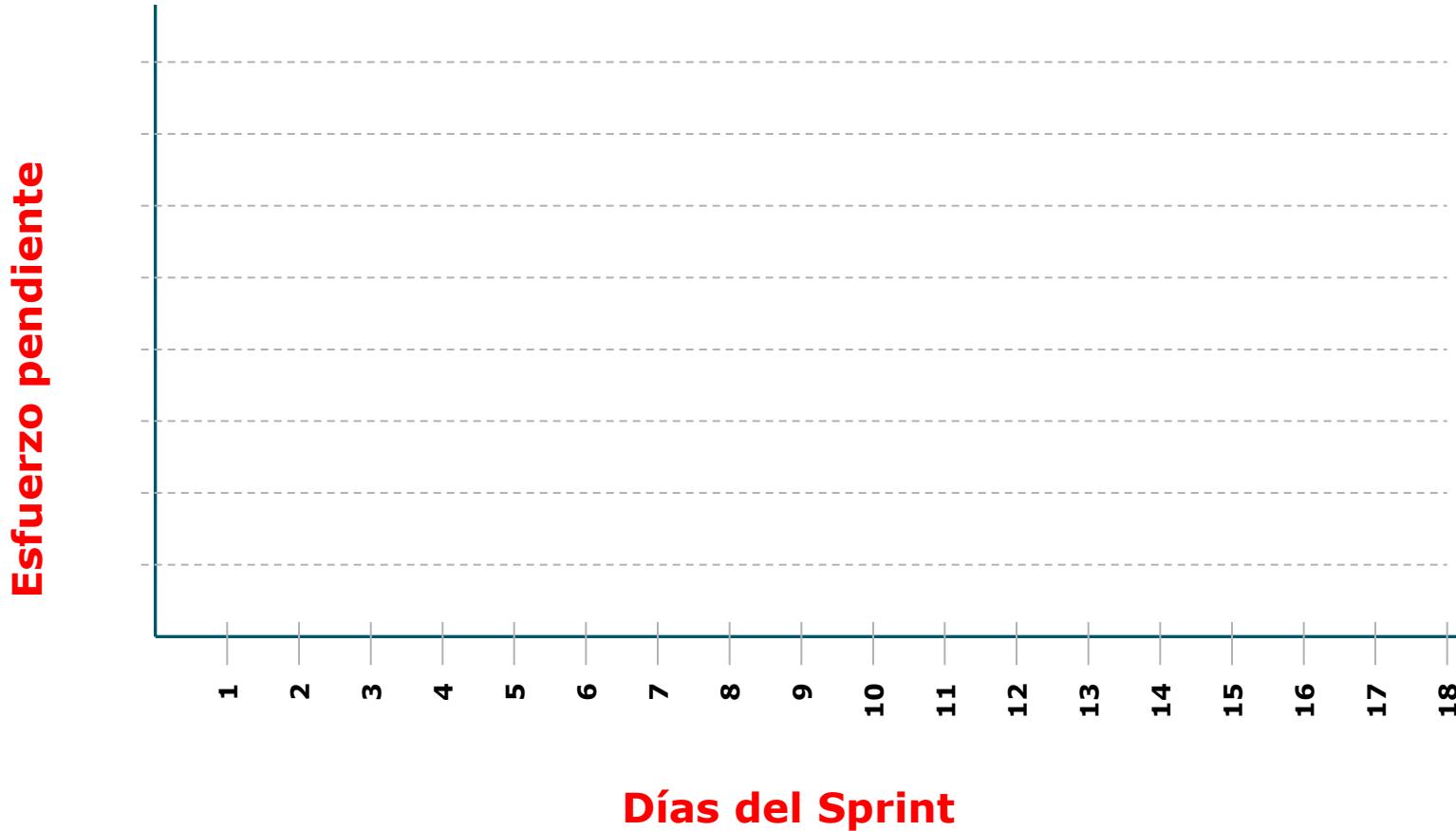


Gráfico de avance (burn down)

Se cumplimenta día a día

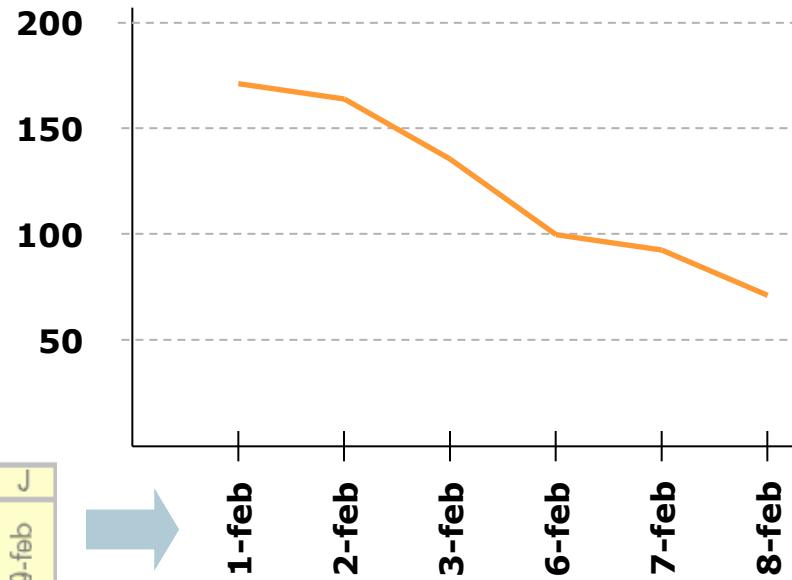


X	J	V	L	M	X	J
1-feb	2-feb	3-feb	6-feb	7-feb	8-feb	9-feb
12	12	11	9	8	6	5

Tareas pendientes

X	J	V	L	M	X	J
1-feb	2-feb	3-feb	6-feb	7-feb	8-feb	9-feb
176	164	144	111	93	76	50

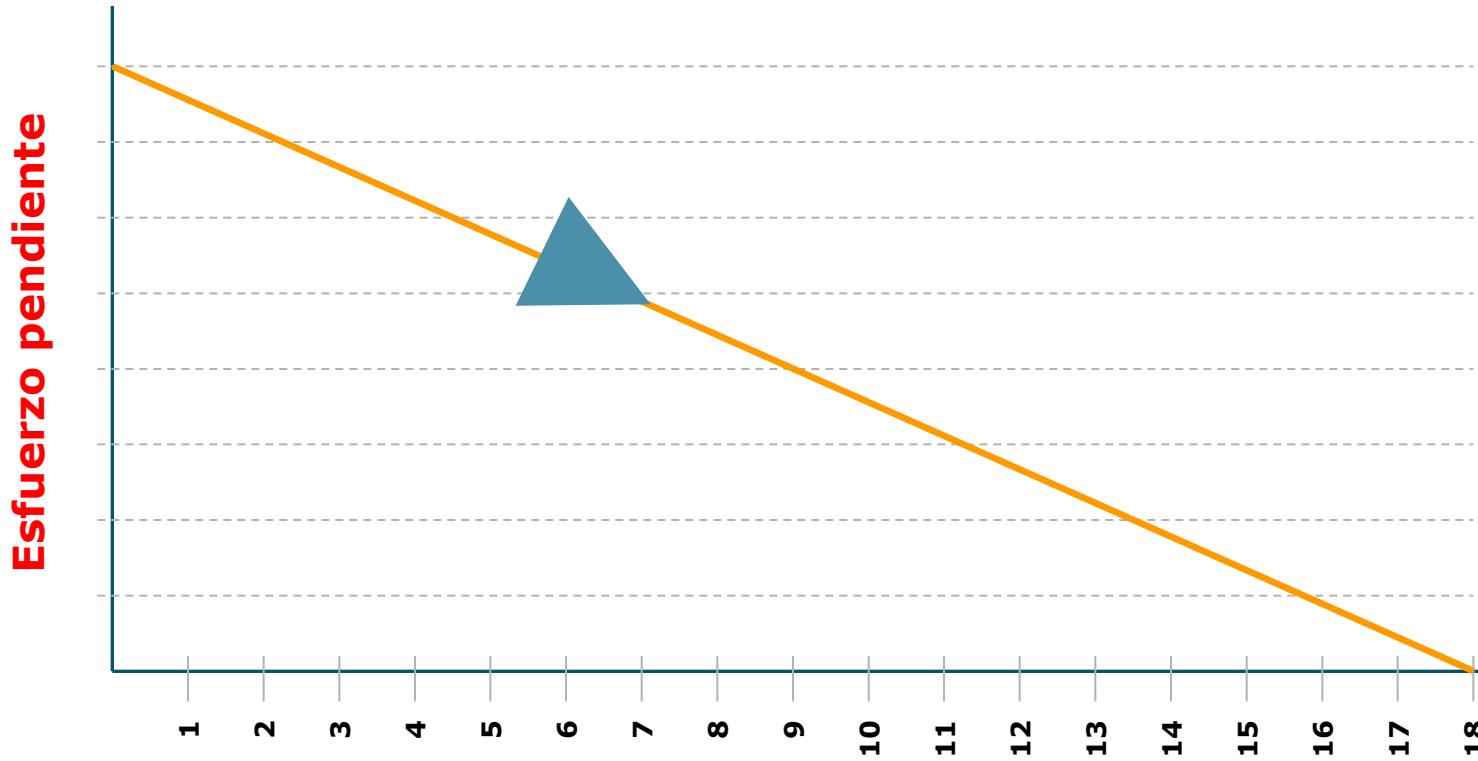
Horas de trabajo pendientes



	Estado	Responsable				
1	Terminada	Luis	24	20	14	
2	Terminada	Luis	8	8	8	
3	Terminada	Luis	16	16	16	10
4	Terminada	Flores	16	8		

Gráfico de avance (burn down)

Indicador de progreso del sprint



Plan según las estimaciones al inicio del sprint

Gráfico de avance (burn down)

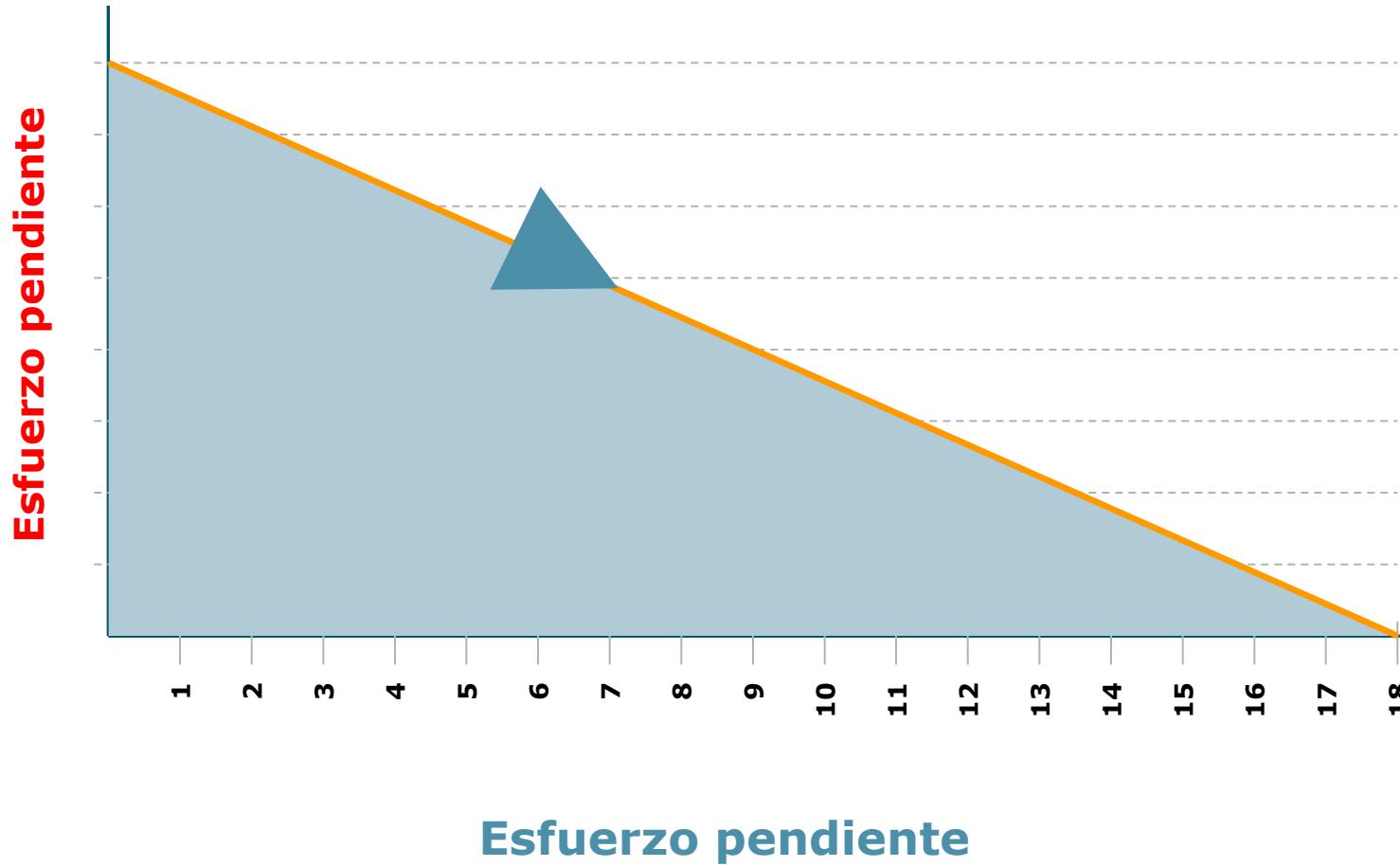


Gráfico de avance (burn down)

Indicador de progreso del sprint

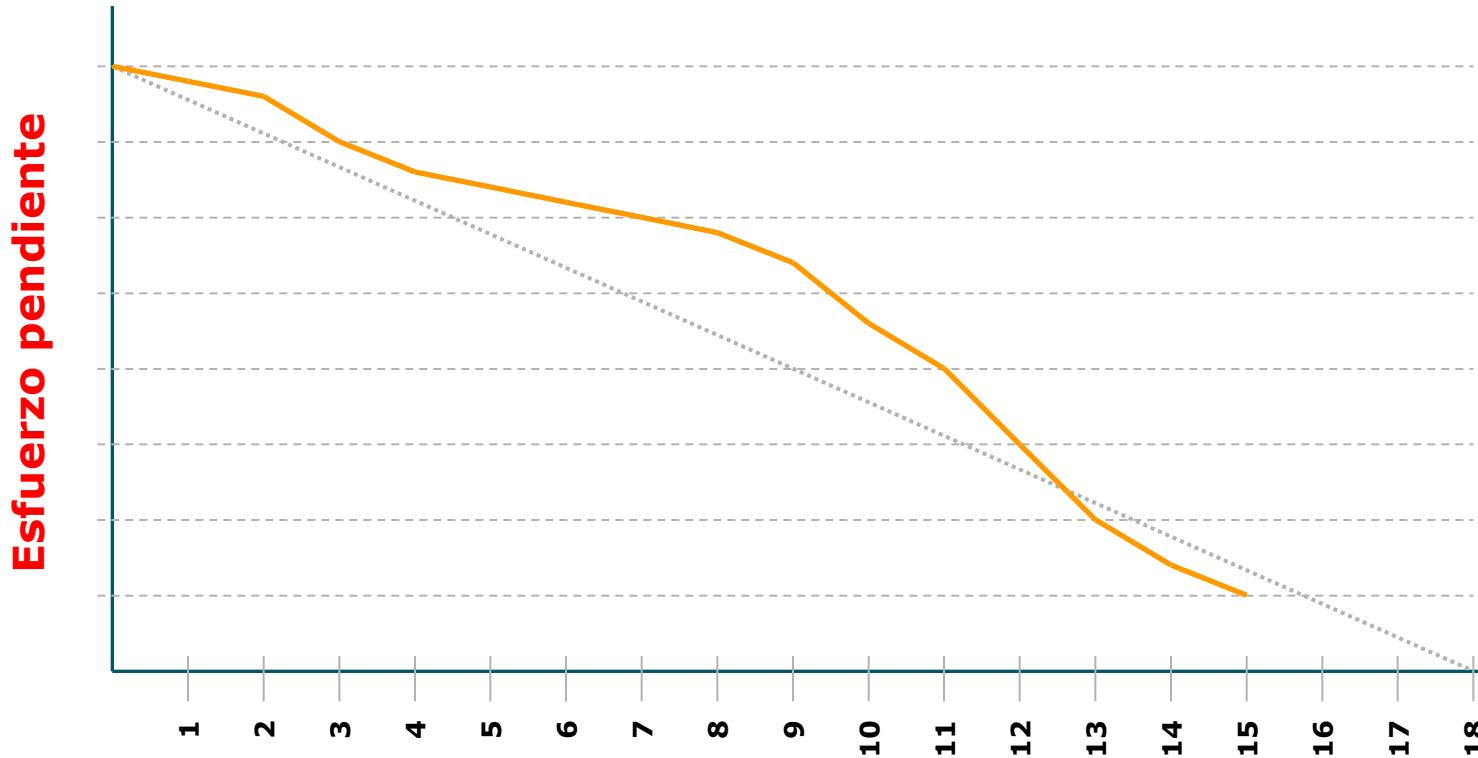


Gráfico de avance (burn down)

Indicador de progreso del sprint

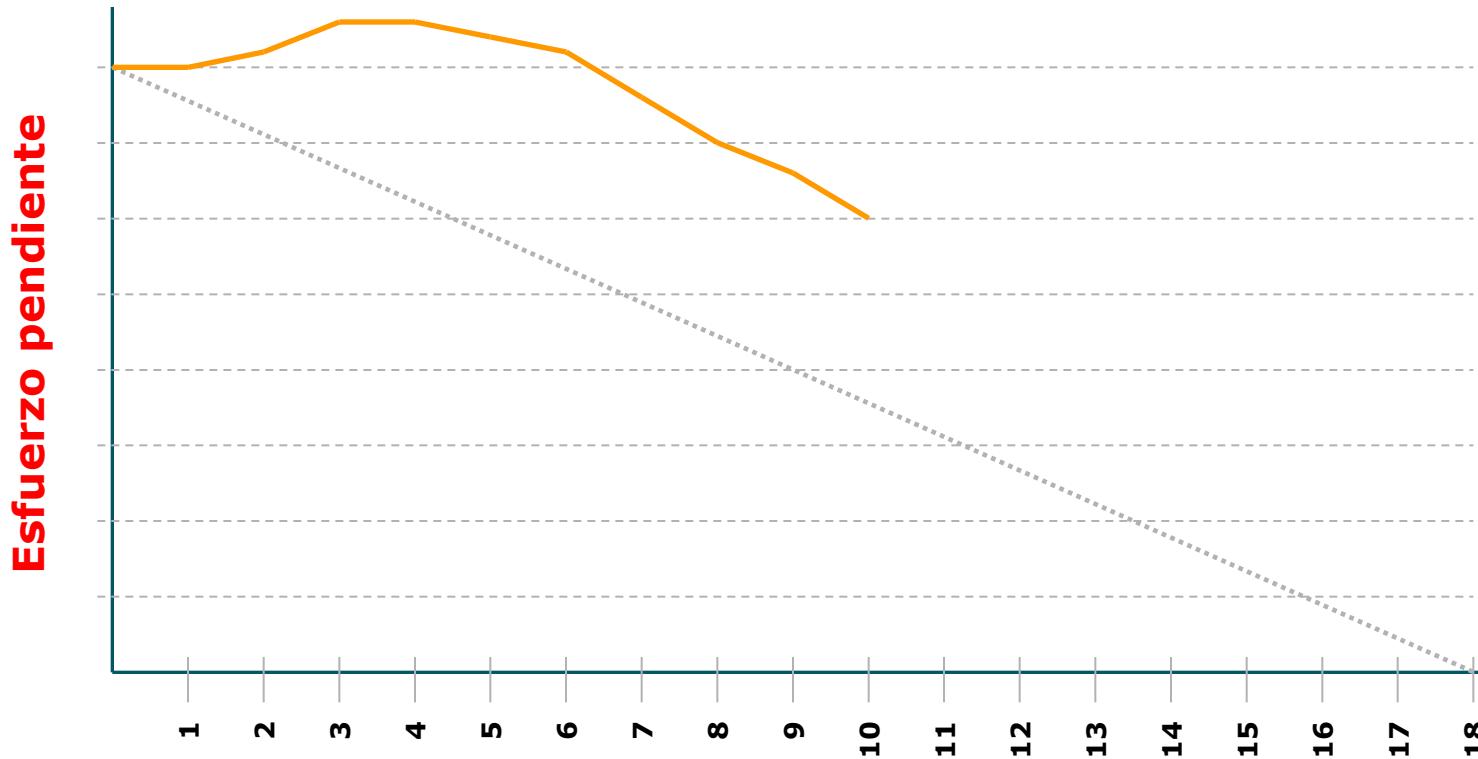
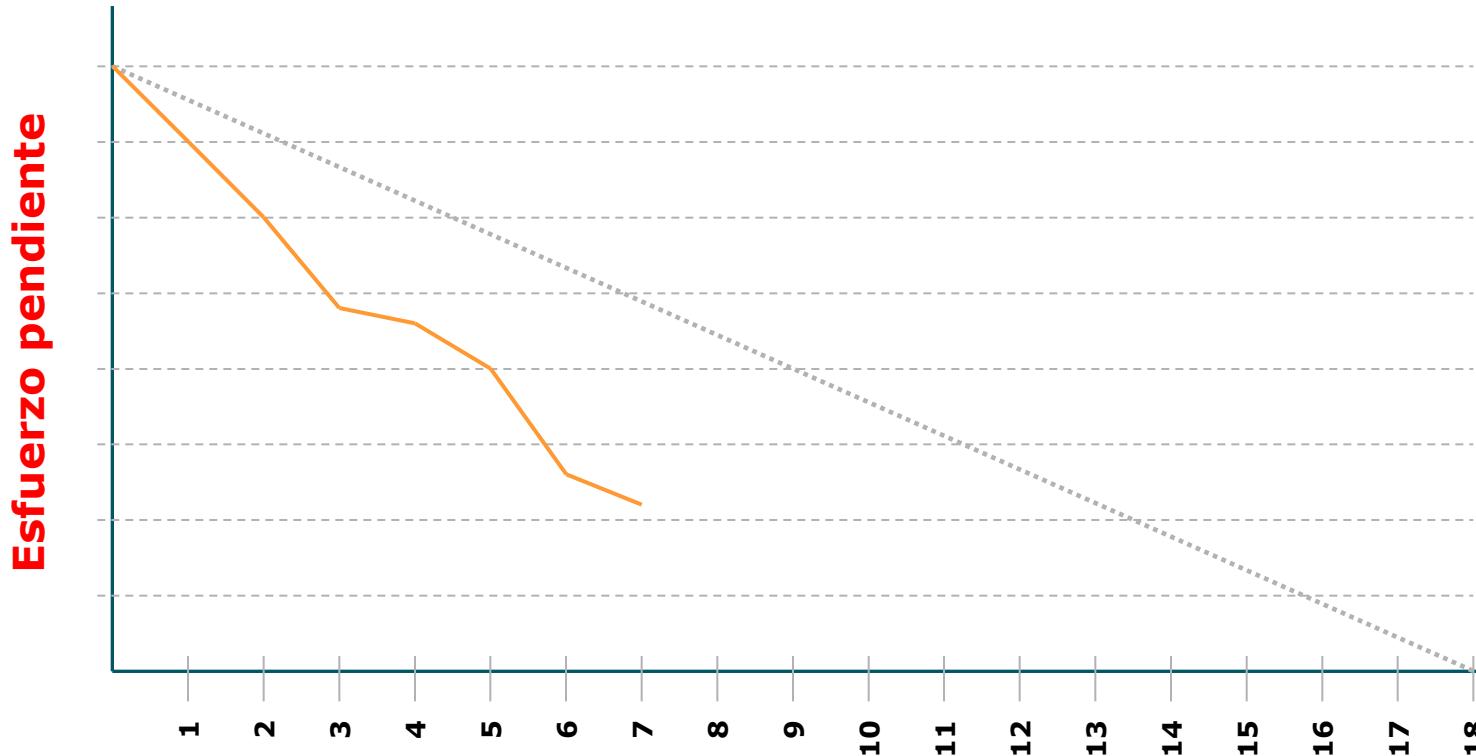


Gráfico de avance (burn down)

Indicador de progreso del sprint



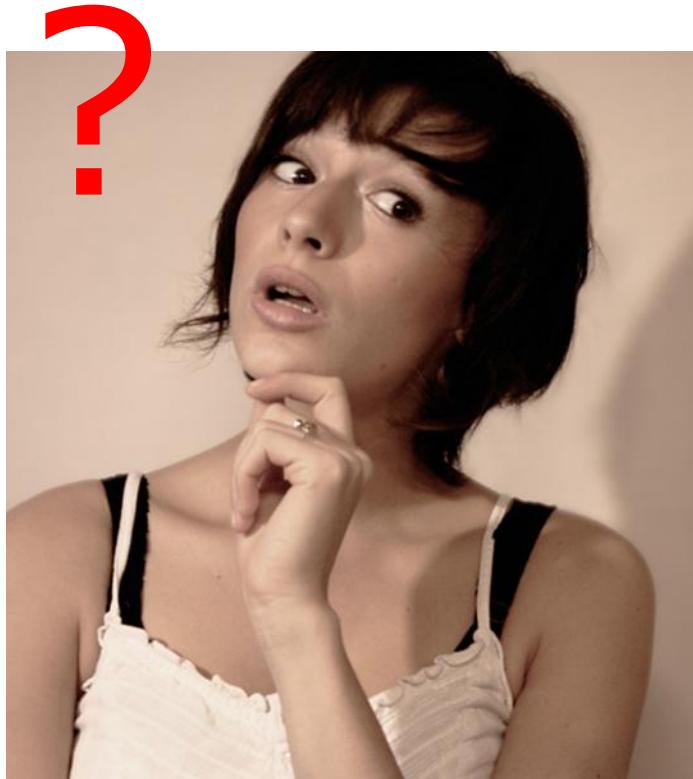
Kanban

Conceptos kanban y lean en gestión ágil

1.0



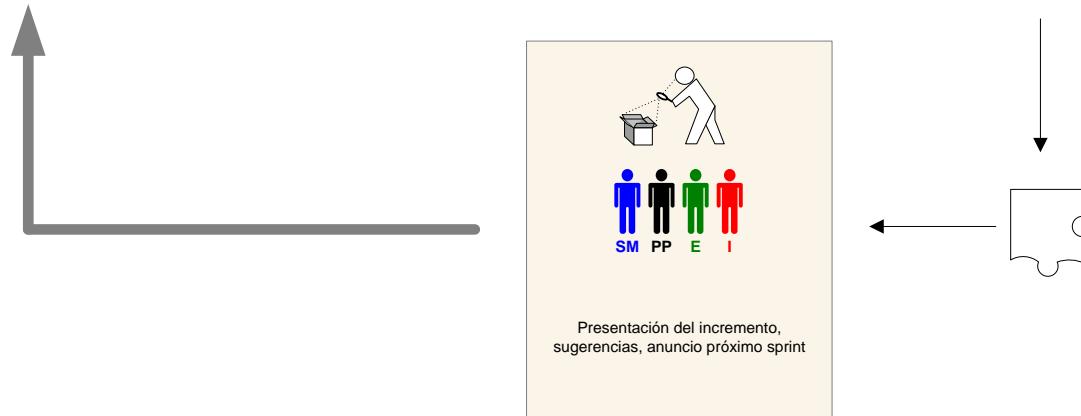
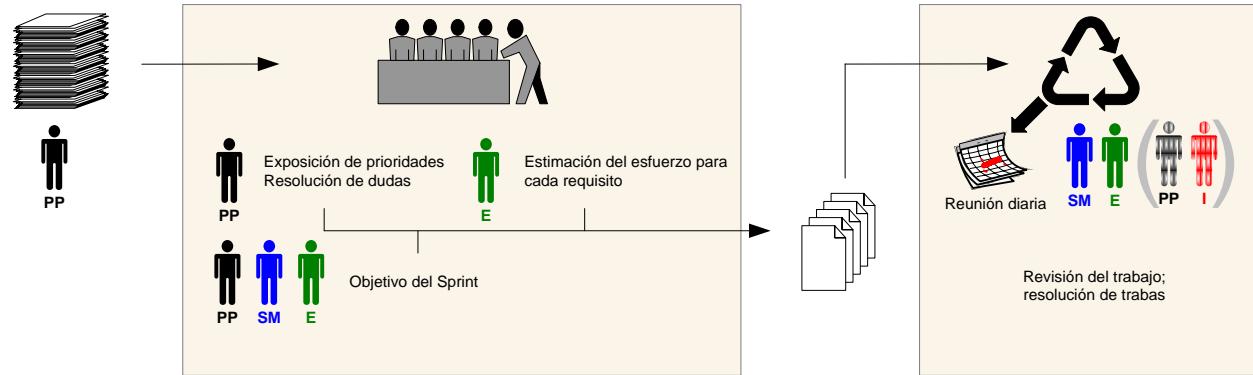
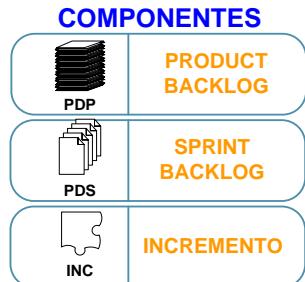
cc-by [Roel Paap](#)



cc-by: [Frédéric Dupont](#)

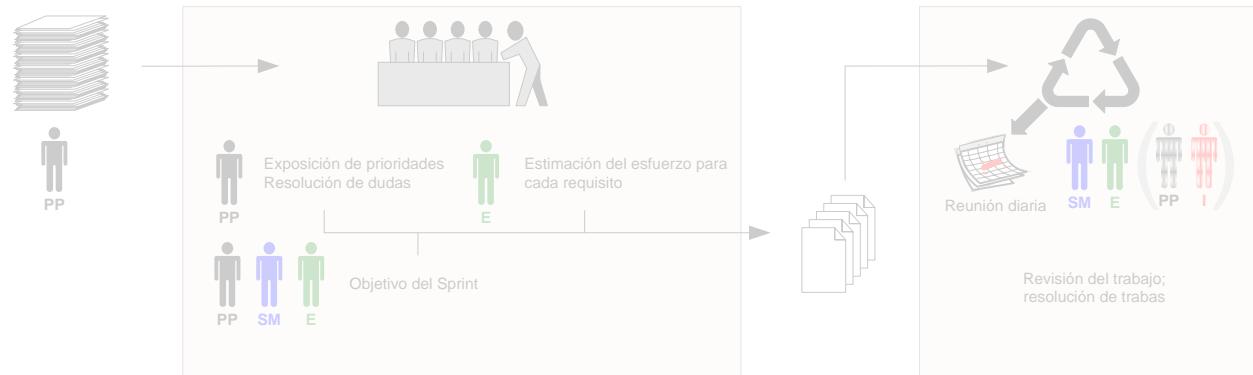


SCRUM



Kanban (Conceptos kanban y lean en gestión ágil)

KANBAN



SCRUM PRESCRIBE ROLES



Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



SCRUM LIMITA EL WIP POR ITERACIÓN

KANBAN LIMITA EL WIP POR ESTADO DEL FLUJO DEL TRABAJO



Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



Kanban and Scrum – making the most of both

Henrik kniberg & Mattias Skarin –

KANBAN



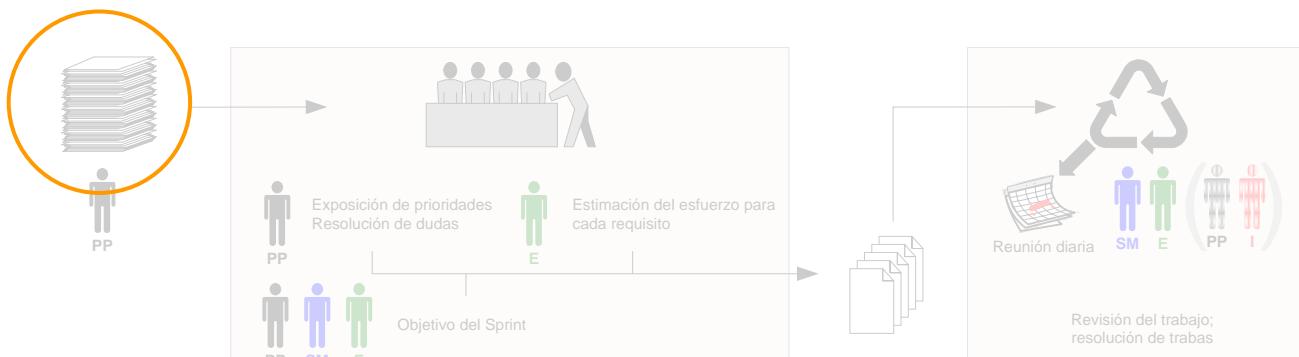
SCRUM NO PERMITE CAMBIAR TAREAS DEL SPRINT

EN KANBAN LA TAREA PUEDE ALTERARSE HASTA ENTRAR EN EL FLUJO

Presentación del incremento, sugerencias, anuncio próximo sprint

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



EN SCRUM LA PILA DE PRODUCTO DEBE TENER LA LONGITUD DE AL MENOS UN SPRINT

EN KANBAN SE DEBE ATENDER AL RITMO DE ARRASTRE DE TAREAS

Presentación del incremento,
sugerencias, anuncio próximo sprint

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN

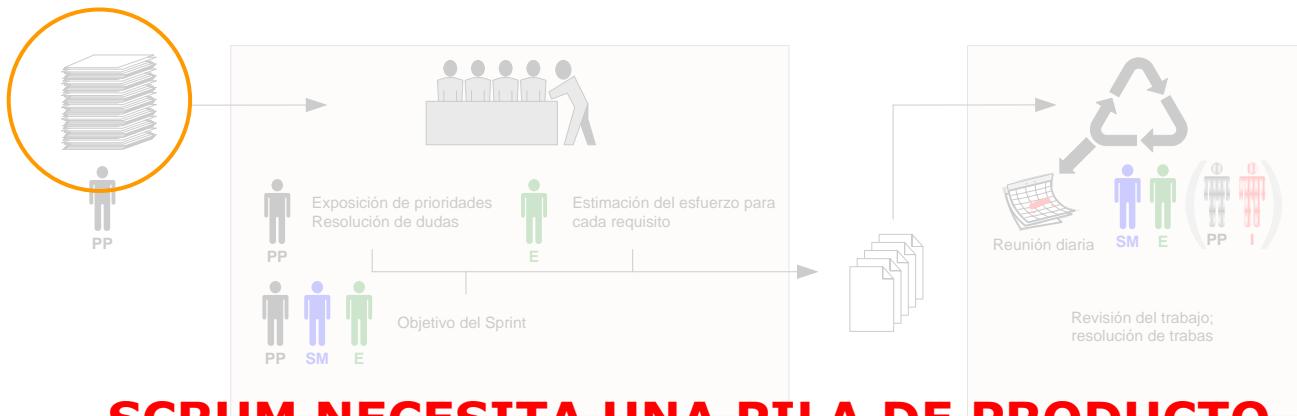


EN SCRUM SE DEBEN ESTIMAR LAS HISTORIAS Y LAS TAREAS Y CALCULAR LA VELOCIDAD

KANBAN NO MIDE TAREAS NI VELOCIDAD

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



SCRUM NECESITA UNA PILA DE PRODUCTO PRIORIZADA

EN KANBAN ES LA SIGUIENTE HISTORIA O TAREA ARRASTRADA DESDE EL CLIENTE

Presentación del incremento,
sugerencias, anuncio próximo sprint

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



SCRUM PRESCRIBE REUNIONES DIARIAS

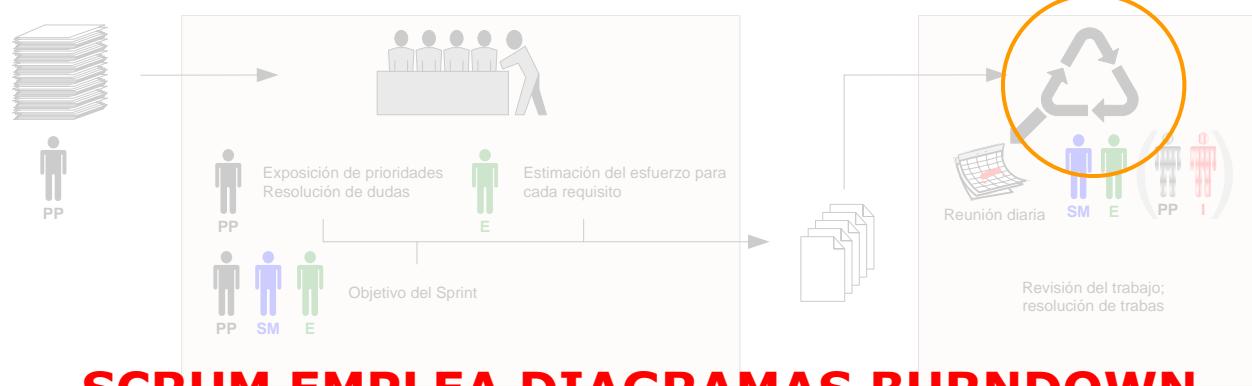


KANBAN NO

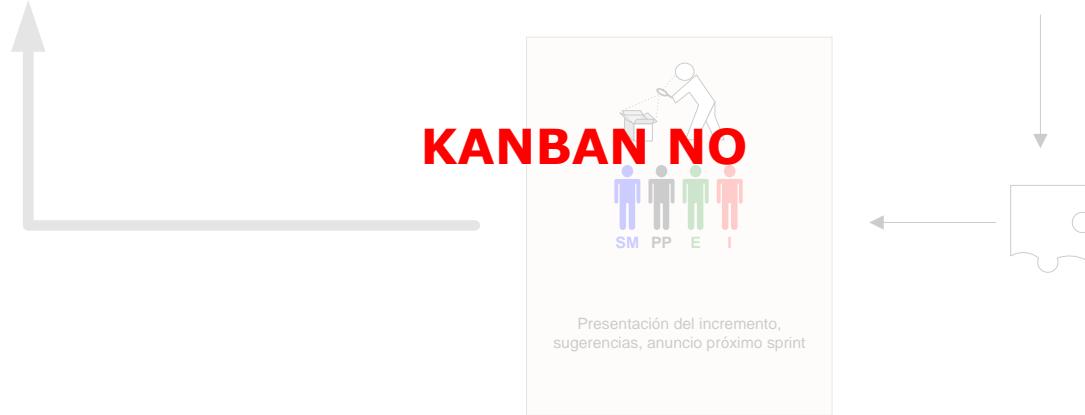
Presentación del incremento,
sugerencias, anuncio próximo sprint

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN

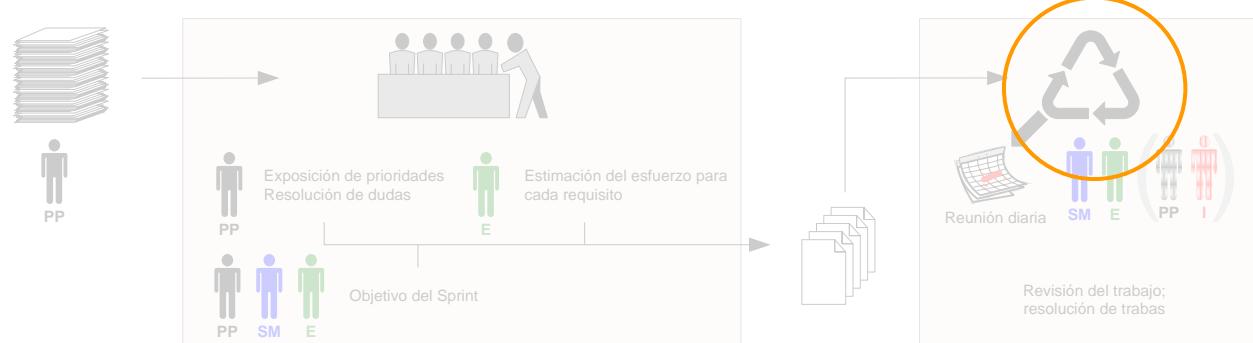


SCRUM EMPLEA DIAGRAMAS BURNDOWN

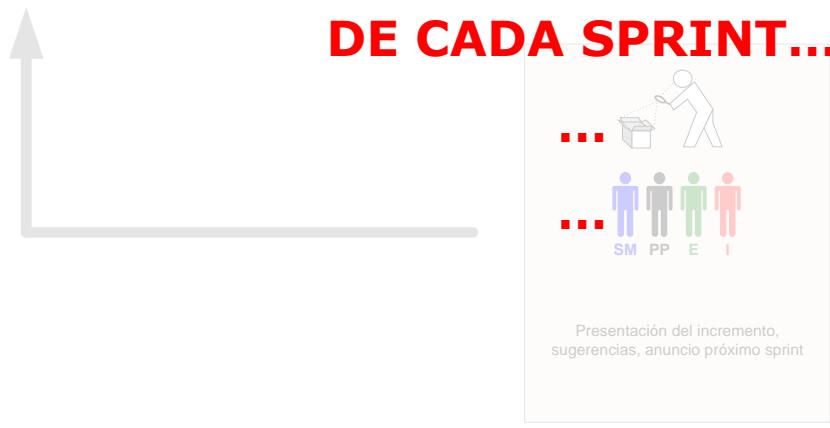


Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN

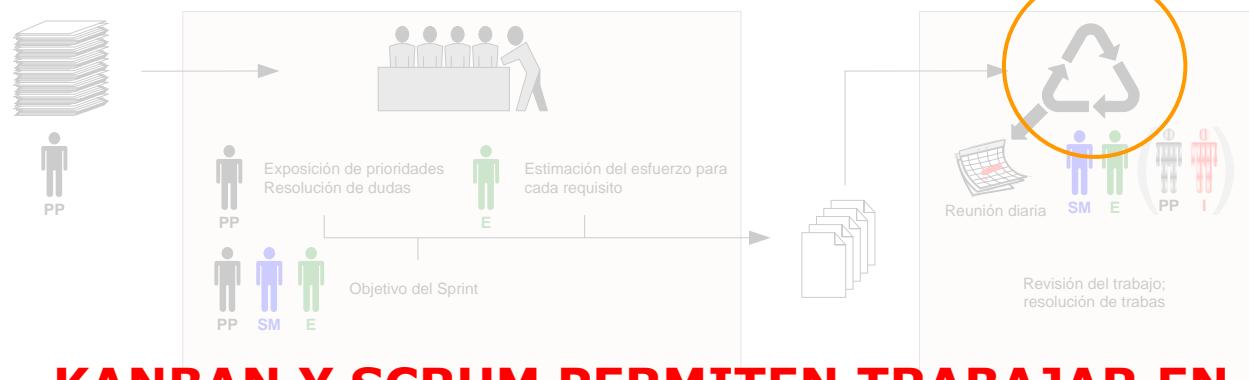


LOS TABLEROS SCRUM SE RESETEAN AL FINAL DE CADA SPRINT...



Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

KANBAN



KANBAN Y SCRUM PERMITEN TRABAJAR EN VARIOS PROYECTOS A LA VEZ

AMBOS SON LEAN Y ÁGILES

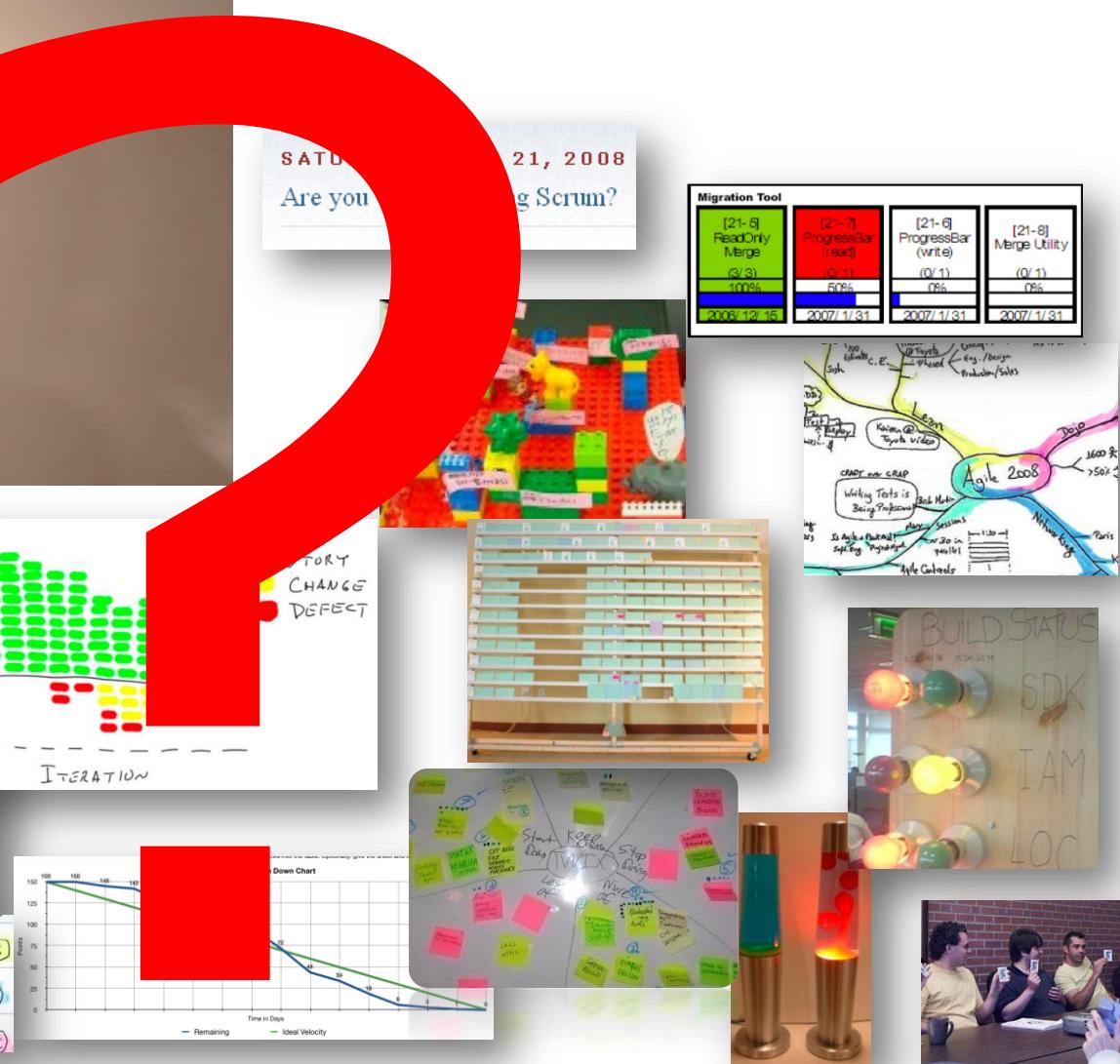
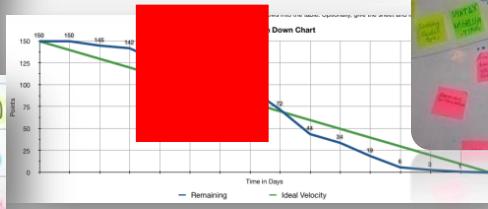
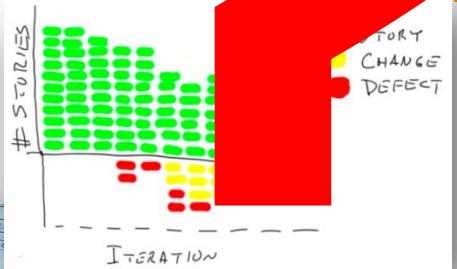
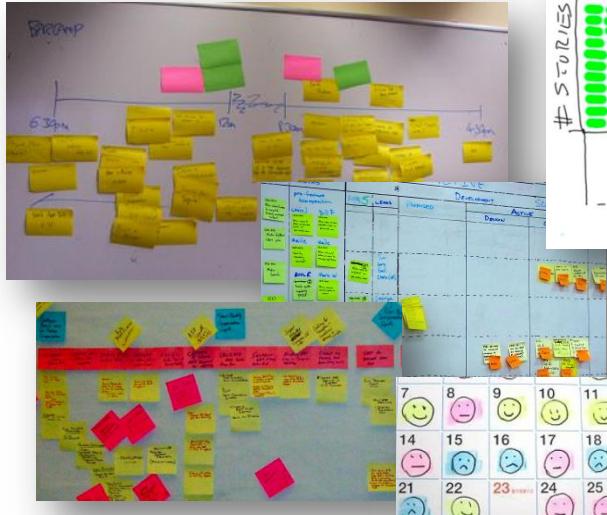
ETC..

Presentación del incremento,
sugerencias, anuncio próximo sprint

Kanban and Scrum – making the most of both
Henrik kniberg & Mattias Skarin –

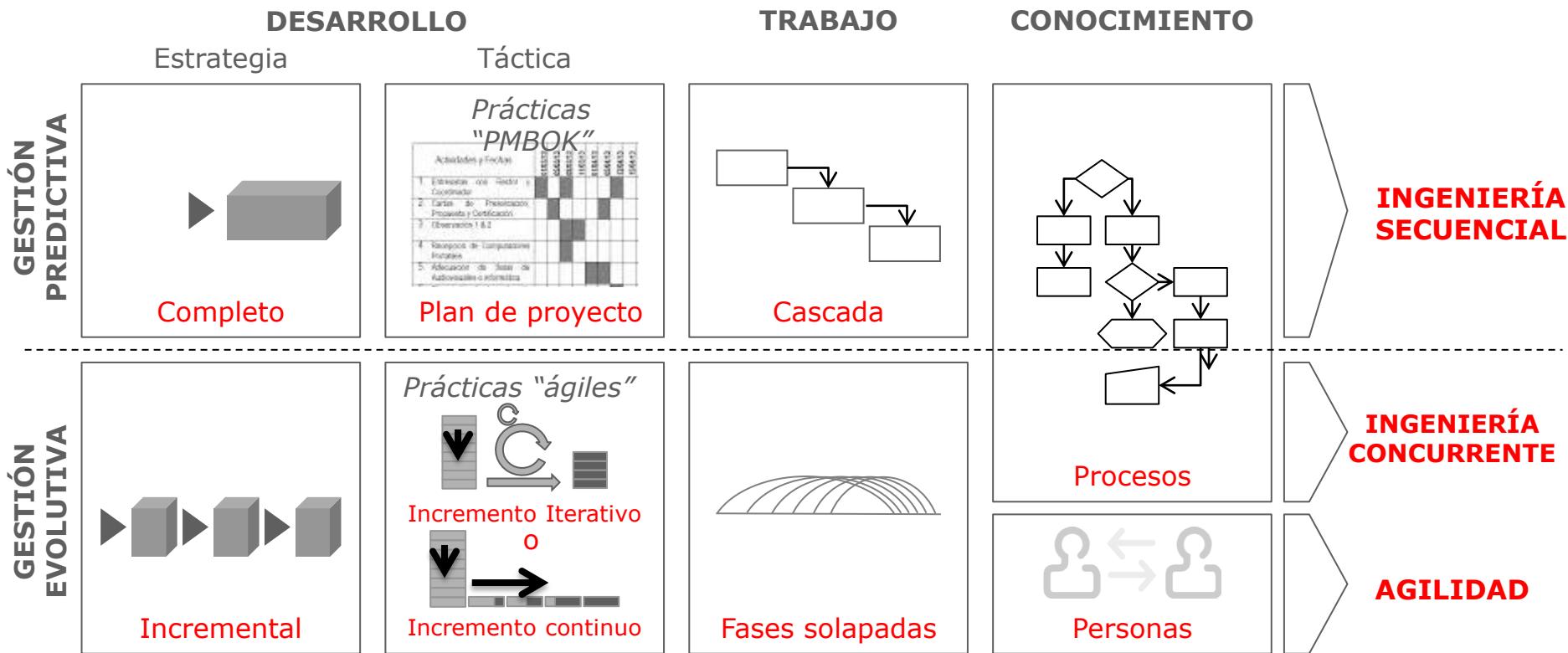


cc-by: Frédéric Dupont

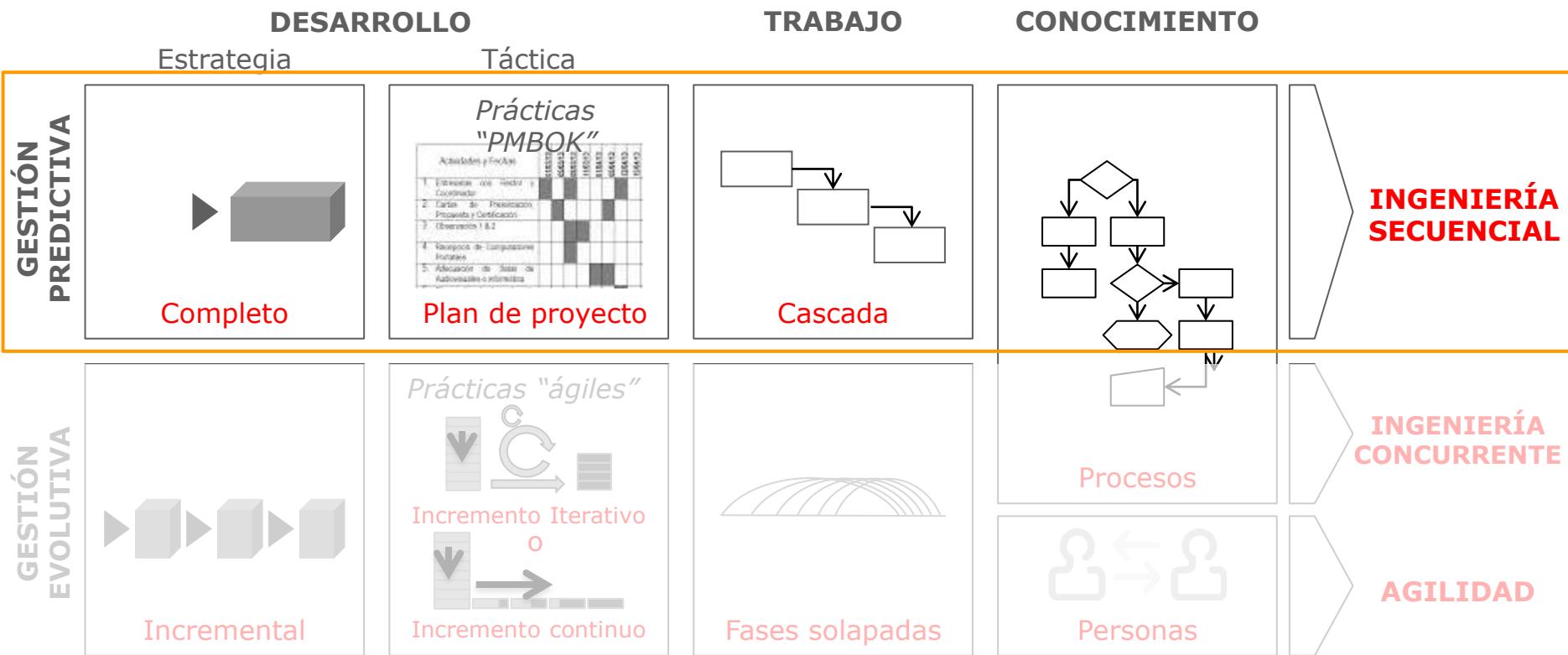


cc-by: [Frédéric Dupont](#)

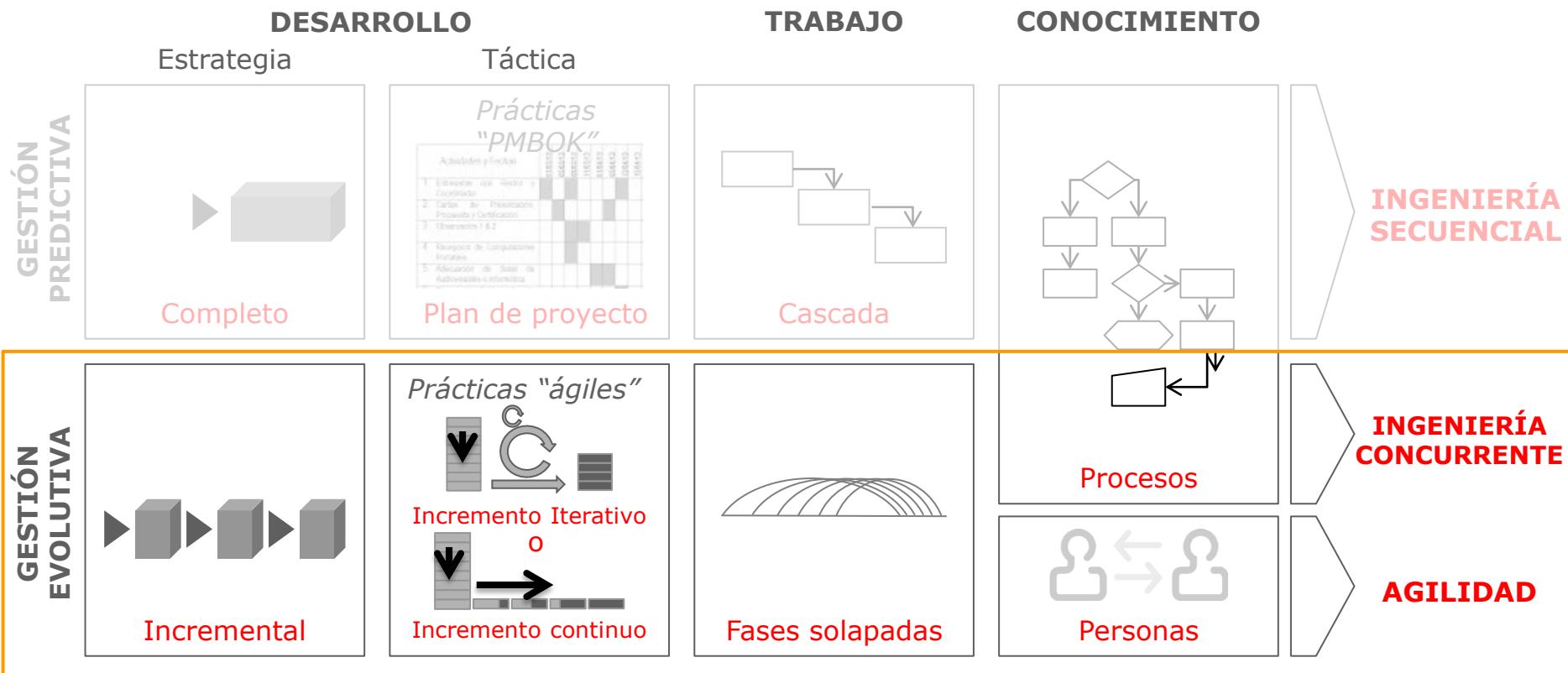
MAPA DE SITUACIÓN: GESTIÓN DE PROYECTOS



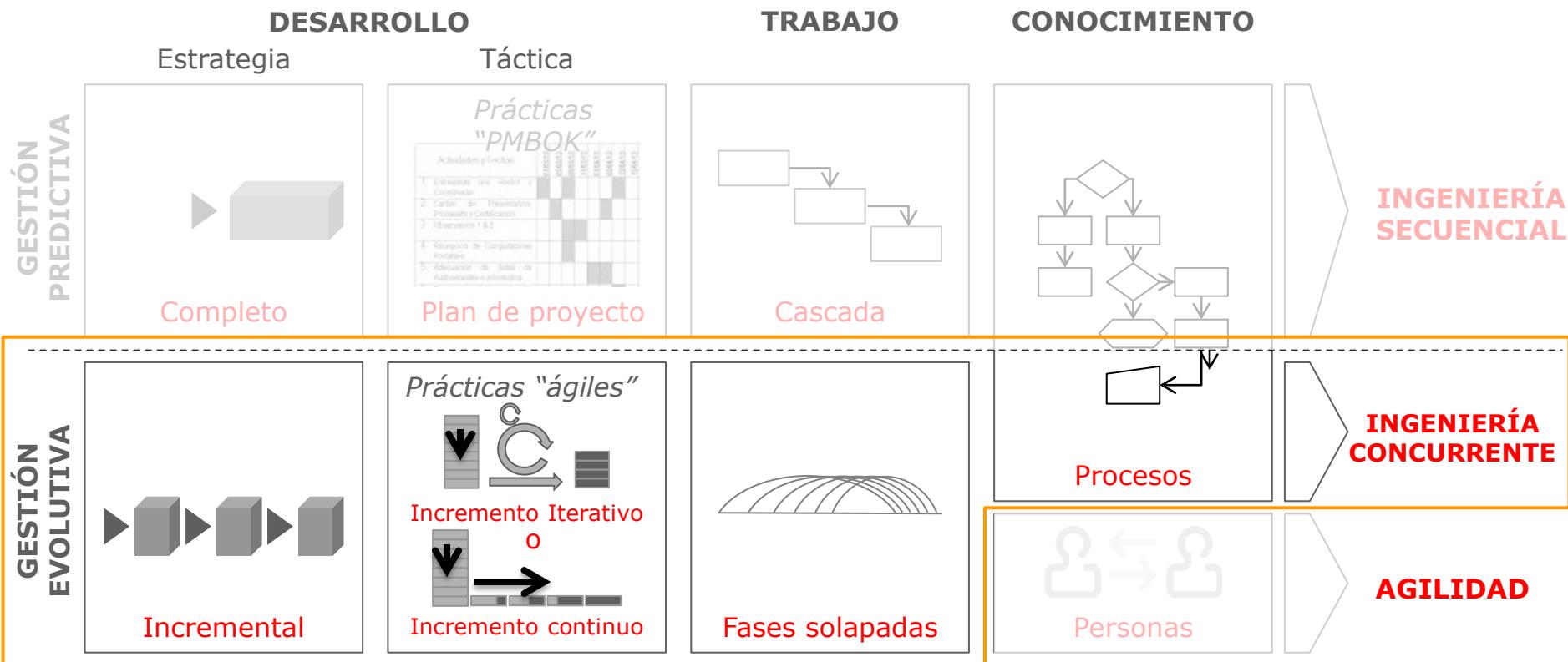
MAPA DE SITUACIÓN: GESTIÓN PREDICTIVA



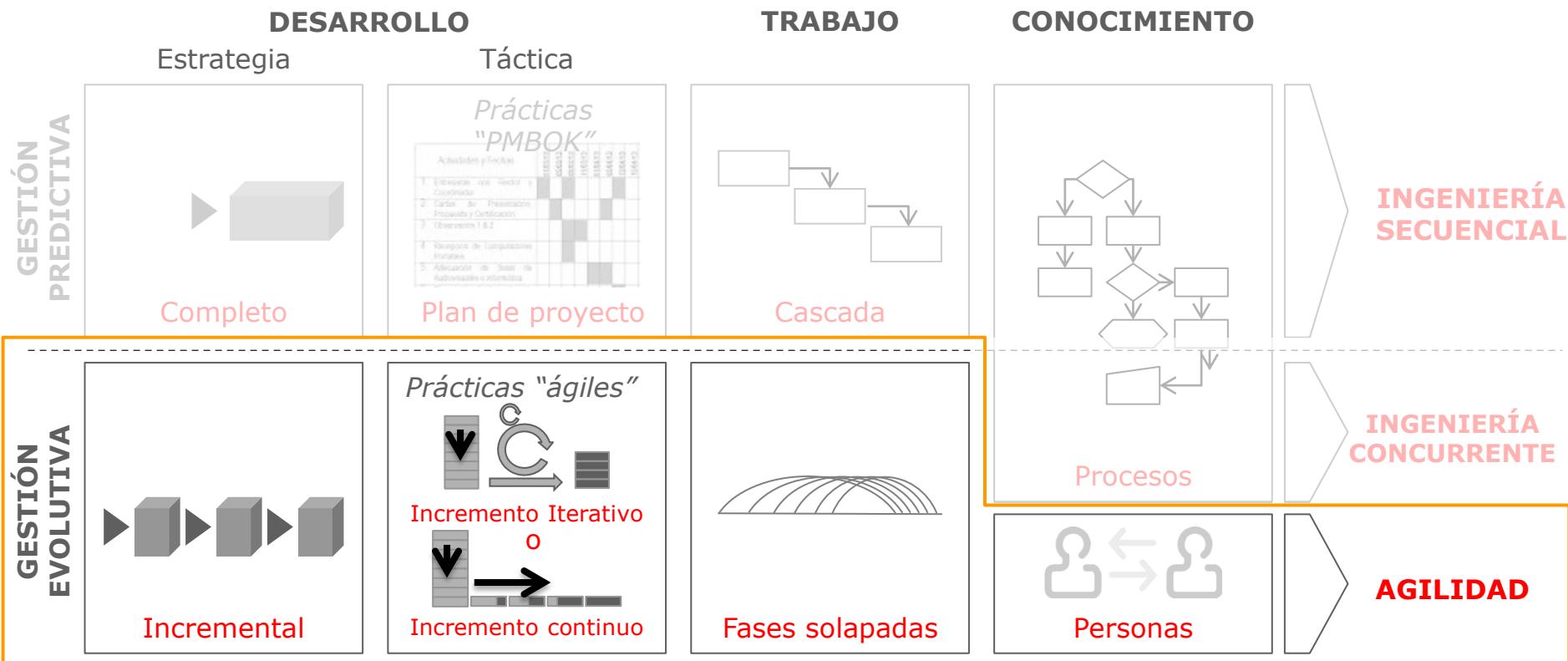
MAPA DE SITUACIÓN: GESTIÓN EVOLUTIVA



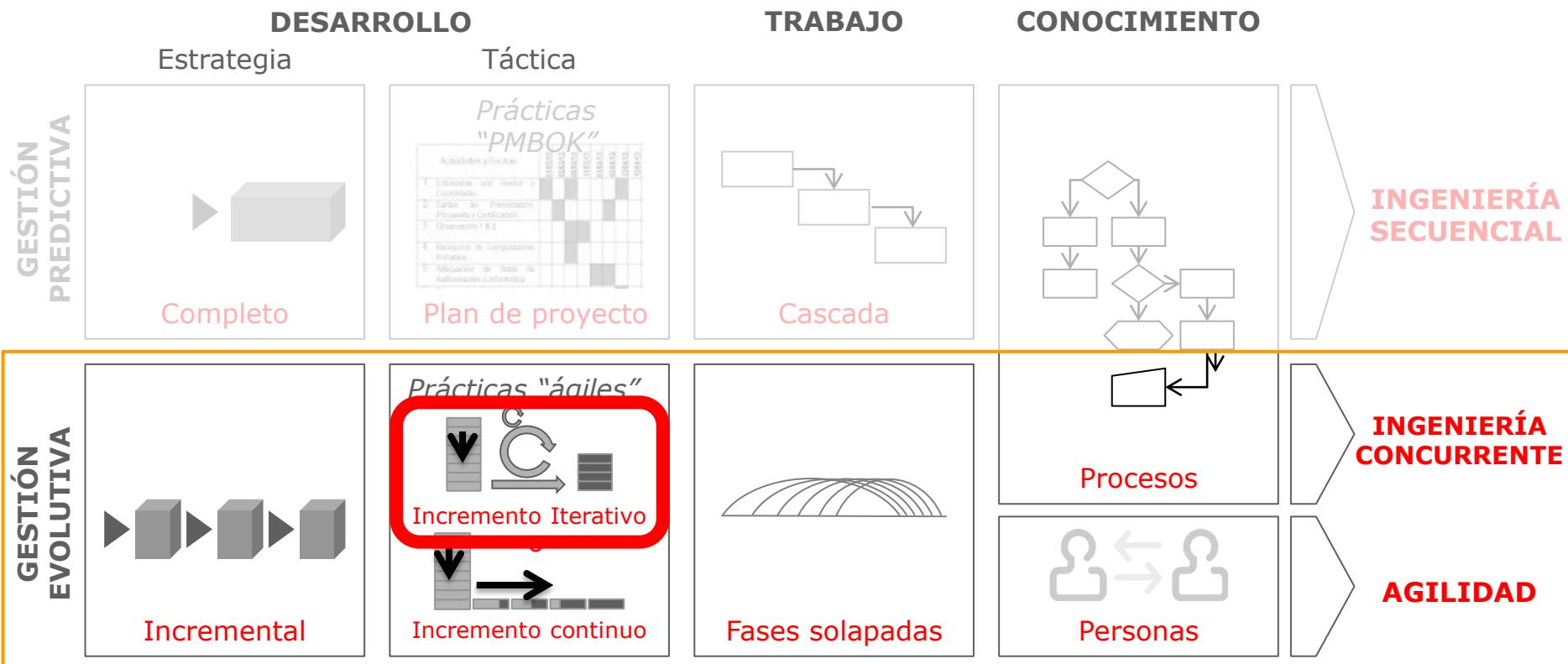
MAPA DE SITUACIÓN: INGENIERÍA CONCURRENTE



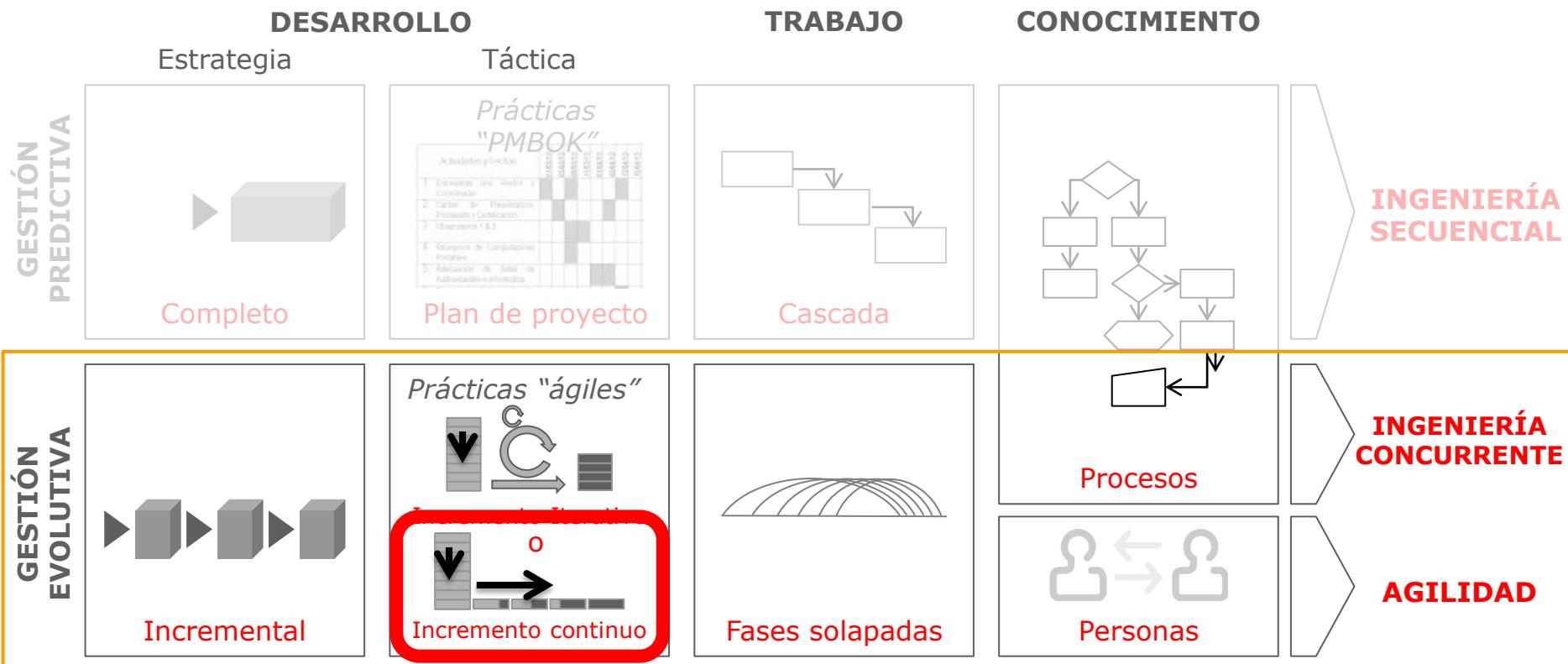
MAPA DE SITUACIÓN: AGILIDAD



MAPA DE SITUACIÓN: AGILIDAD: PRÁCTICAS SCRUM



MAPA DE SITUACIÓN: AGILIDAD: PRÁCTICAS KANBAN





A large red question mark is overlaid on the image of the woman. Inside the question mark, there is a smaller image showing the letters "CMMI" and "KANBAN" partially visible.



Modelos
Gestión c
proyectos

DSQM[®]
consortium

Ágil
(Producción basada en personas)

Ingeniería
consciente
(Producción basada en procesos)



mento iterativo
(Incremento: Sprint principio: "time boxing")

Incremento continuo
(Incremento: Historia de usuario principio: "flujo lean")



Carnegie Mellon
Software Engineering Institute



VÍDEO

cc-by: [Betsy Fletcher](#)



VÍDEO

cc-by: [Betsy Fletcher](#)



PRINCIPIOS

Kanban (Conceptos kanban y lean en gestión ágil)



METODOLOGÍAS



SCRUM



XP



TDD



PMI

PRINCE²



CMMI

Software Engineering Institute

SCRUMBAN

GESTIÓN

Experta
Principios
Conocimiento tácito

Técnica
Técnicas
Conocimiento explícito

DE LA ARTESANÍA
A LA MANUFACTURA
LEAN

1.0



cc-by State Library of South Australia

PRÁCTICA: De la artesanía a la manufactura lean



Fotografía cc-by: [LizMarie](#)



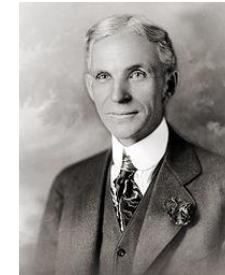
Los principales “culpables” de lean



Adam Smith (1723-1790)
División del trabajo



Frederick Taylor (1856 – 1915)
Organización científica del trabajo



Henry Ford (1863 – 1947)
Producción en cadena



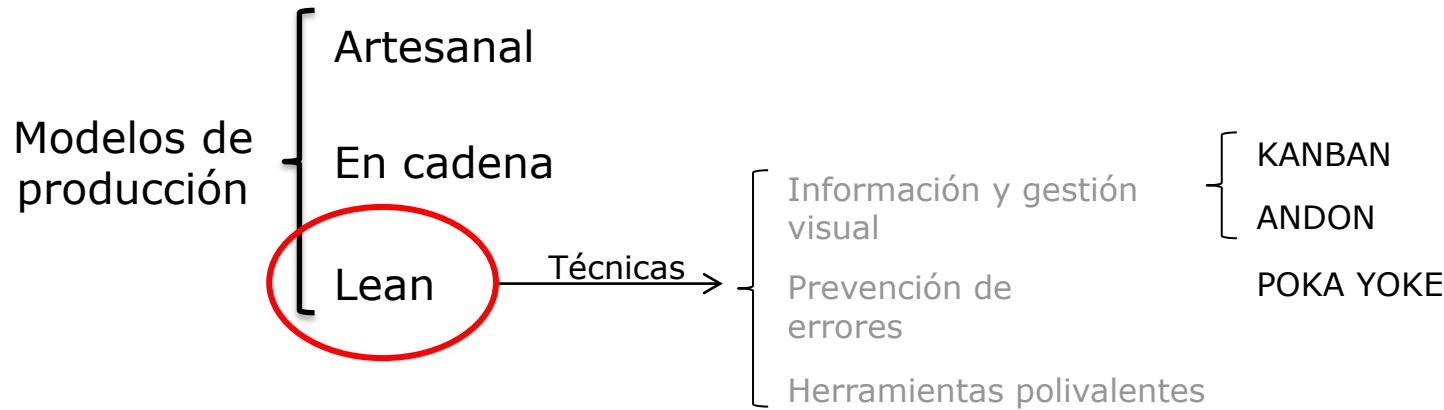
USTED
ESTÁ
AQUÍ



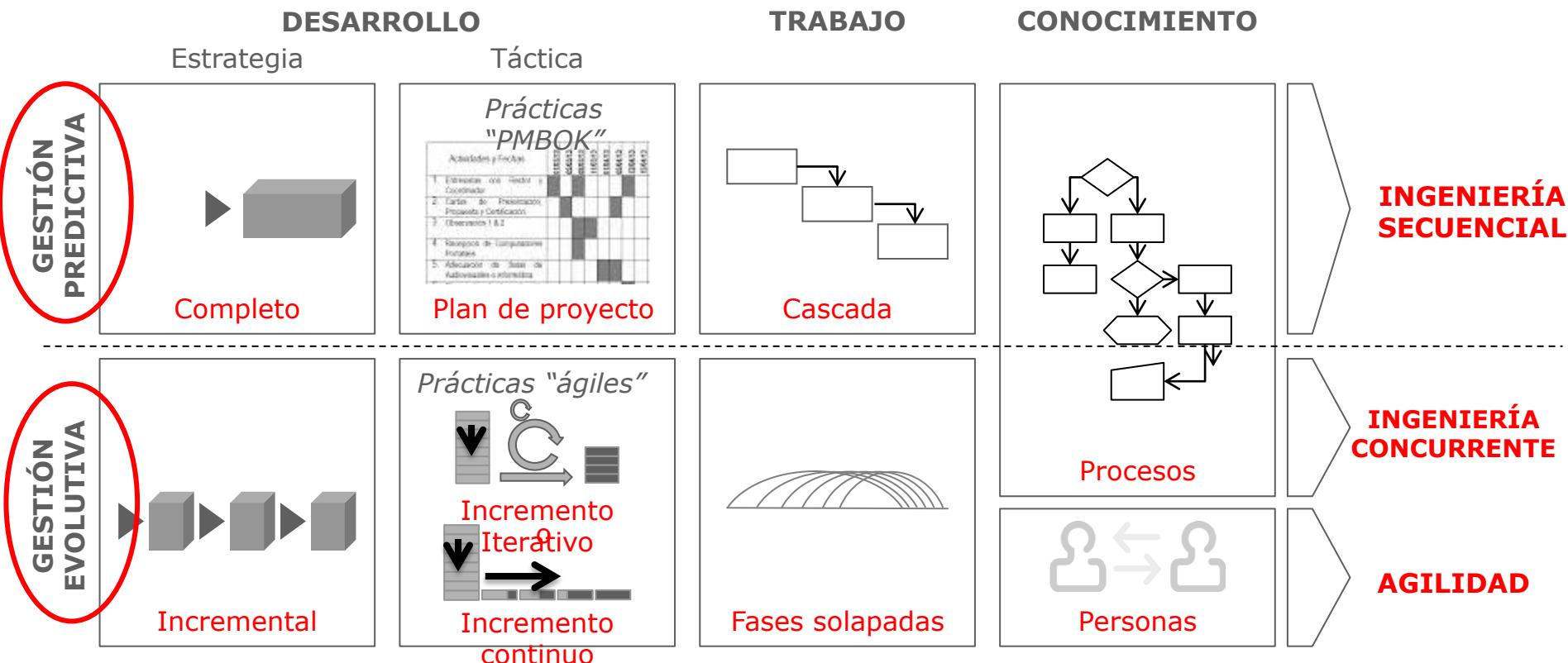
Kichiro Toyoda – Taichi Ohno
Producción: "just in time", "pull" y sin desperdicio
Modelo Toyota o Lean

Edwards Deming (1894 - 1993)
Búsqueda continua de la calidad

MAPA DE SITUACIÓN: MODELOS DE PRODUCCIÓN



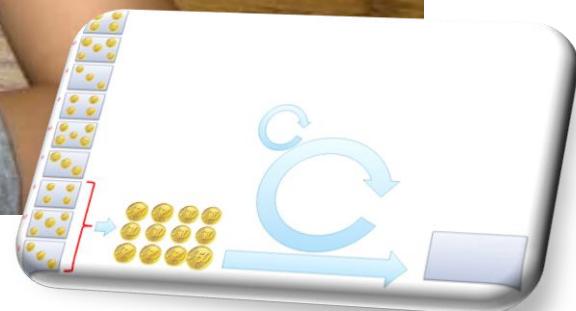
GESTIÓN PREDICTIVA / GESTIÓN EVOLUTIVA



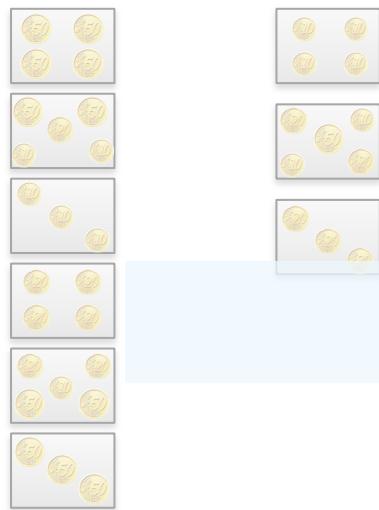
PRÁCTICA: Agile Coins



Fotografía cc-by: [LizMarie](#)

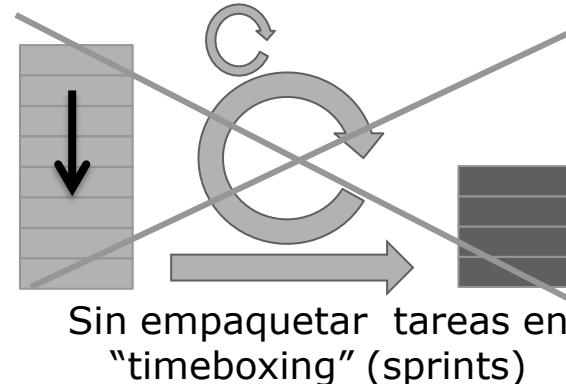
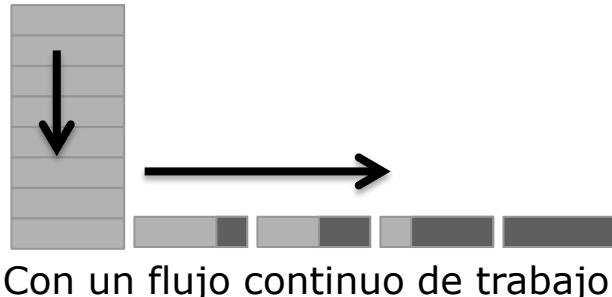


Kanban (Conceptos kanban y lean en gestión ágil)



	1	2	3	4	5	6	7	0	21	22	23	24	25	26	27	28

PRÁCTICA: Para gestionar un DESARROLLO INCREMENTAL



Son apropiadas las **técnicas de gestión visual kanban** para evitar los cuellos de botella y los tiempos muertos.

Ajustándolas con criterios de flexibilidad a las circunstancias de nuestro trabajo y equipo



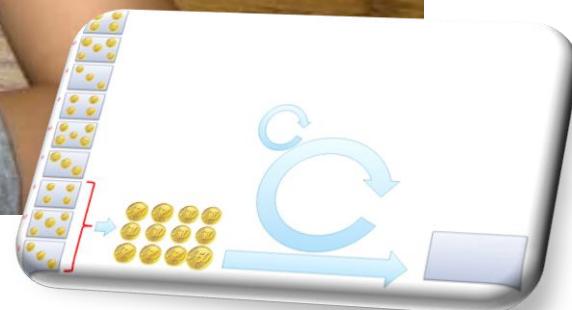
Trabajo {
Secuencial
Libre

Equipo {
Polivalente
Especialistas

PRÁCTICA: Agile Coins



Fotografía cc-by: [LizMarie](#)



CONTROL VISUAL

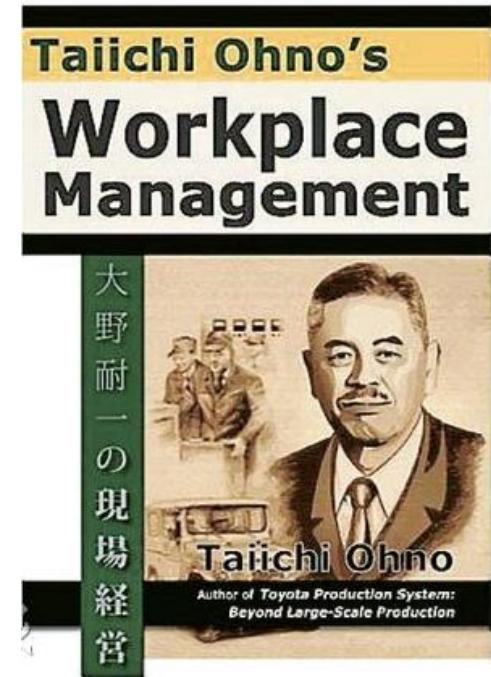
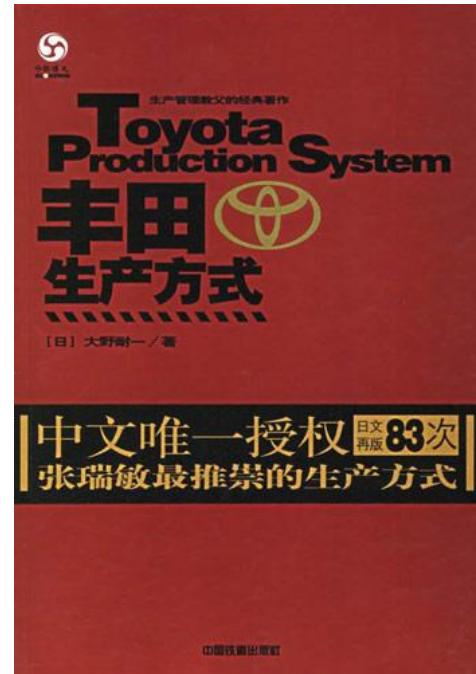


cc-by [Truth About](#)

EL ORIGEN DE KANBAN



Taiichi Ohno's



Los dos pilares del sistema de producción de Toyota son "just-in-time" y "automatización humanizada" o autonomatización (autonomation). La **herramienta** que utilizamos para operar con el sistema es **kanban**.

Taiichi Ohno's

EL ORIGEN DE KANBAN

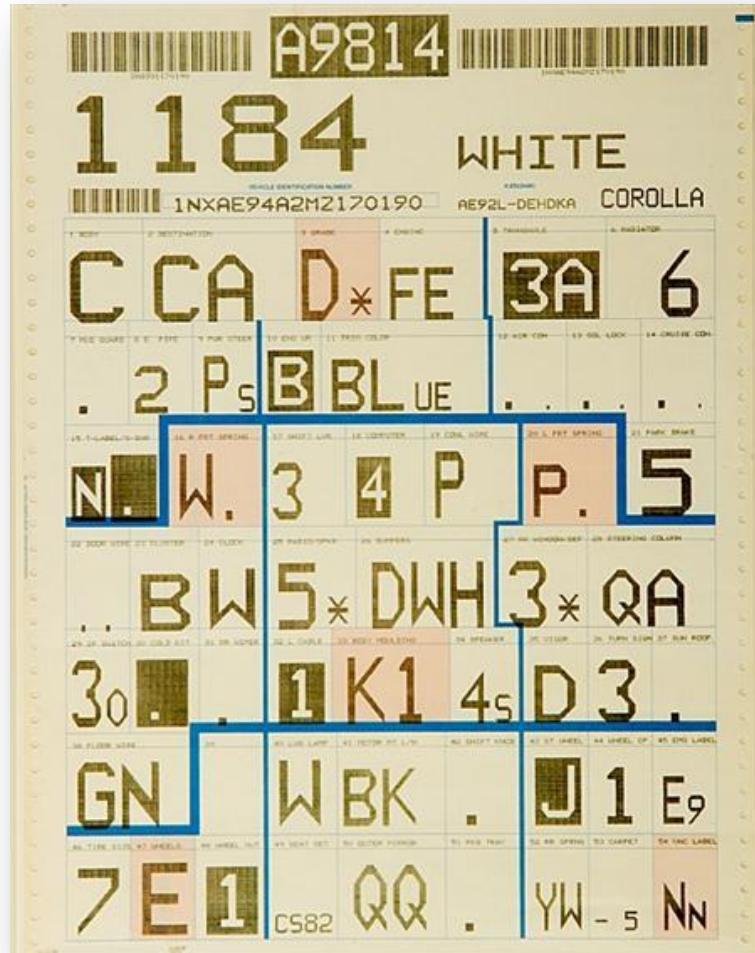
KANBAN

看板

Tarjeta de señal



EJEMPLO: KANBAN DE MONTAJE



Tarjeta Kanban de montaje
de un Toyota Corolla

Se adhiere al auto en la
cadena de montaje.

Contiene indicaciones
legibles para humanos y
máquinas de todos los
detalles del vehículo.

SISTEMA DE CONTROL VISUAL KANBAN



Almacén Estante n°:	5E21	Código Artículo:	A2-15
Artículo n°:	35670		
Artículo nombre:	PIÑÓN TRANSMISIÓN		
Tipo Producto:	SX50BC		
Cap.Caja:		Tipo Caja:	
20		B	Salida n°: 4/8

Kanban de transporte



Almacén Estante n°:	F26-18	Código Artículo:	A5-3
Artículo n°:	56790-1		
Artículo nombre:	EJE CIGÜEÑAL		
Tipo Producto:	SX50BC-150		

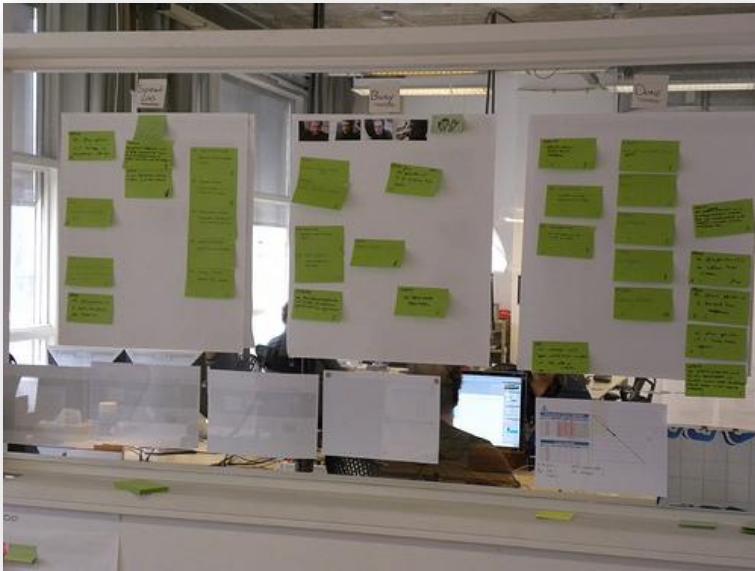
Kanban de producción

cc-by [Gregory Melle](#)

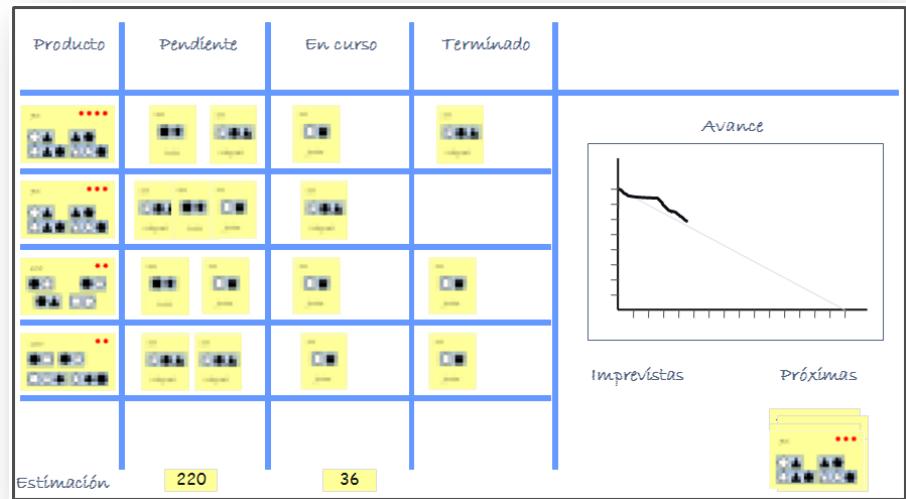
LA GESTIÓN VISUAL FAVORECE LA COMUNICACIÓN DIRECTA

TABLEROS KANBAN

Herramienta de gestión visual



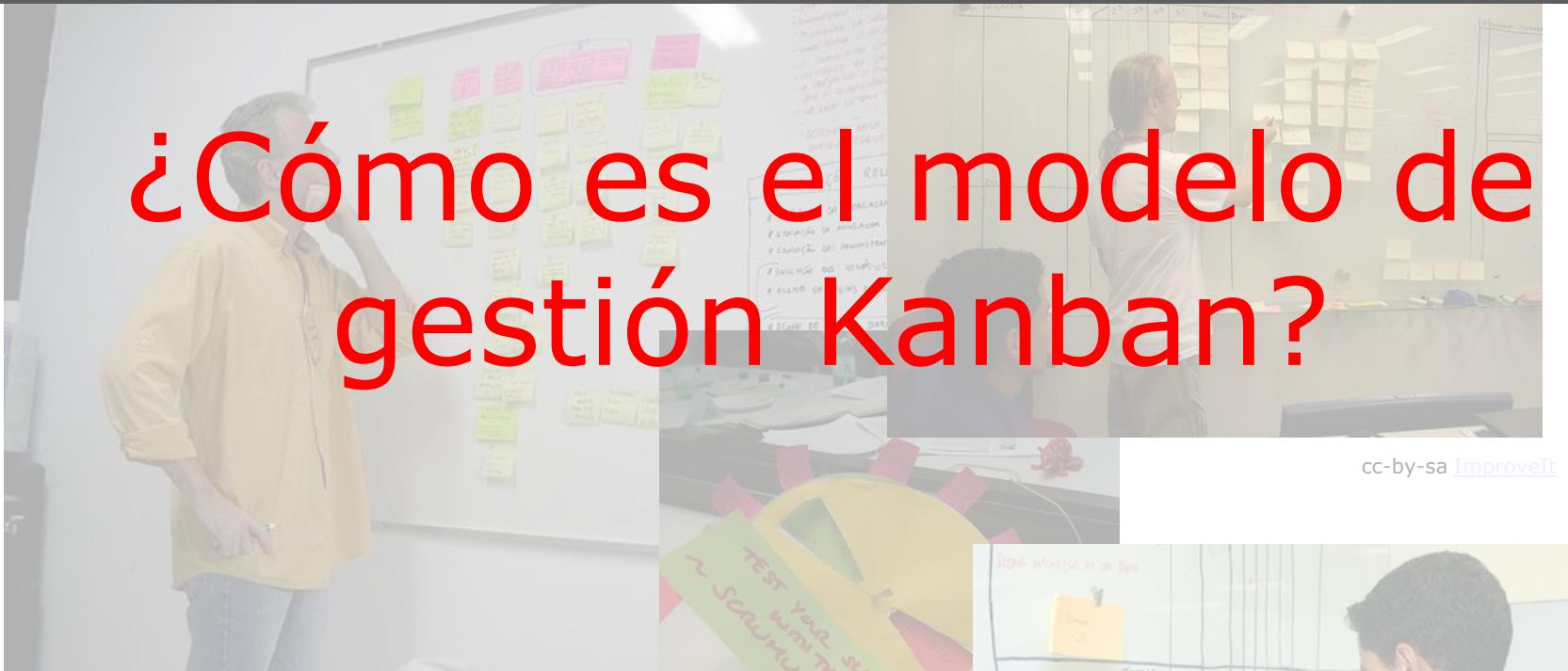
cc-by [Pool Paap](#)



Emplear etiquetas para registrar epics, historias o tareas, y diferentes marcas o su posición en el tablero para representar su estado de desarrollo.

KANBAN: ¿ES UN MODELO DE GESTIÓN?

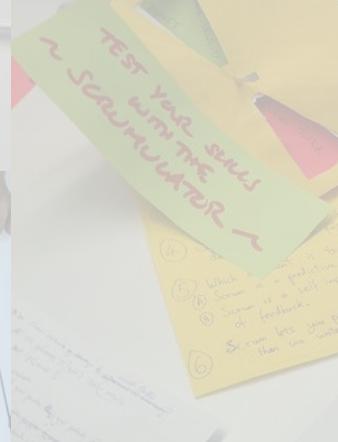
¿Cómo es el modelo de gestión Kanban?



cc-by-sa [ImproveIt](#)



cc-by [hellomedeiros](#)



cc-by [Gilgongo](#)

cc-by-sa [ImproveIt](#)

KANBAN NO ES UN MODELO DE GESTIÓN

¿Cómo es el ~~modelo~~ de gestión Kanban?

Un tablero kanban es una herramienta para
GESTIÓN ÁGIL



cc-by-sa ImproveIt



cc-by heliomedeiros

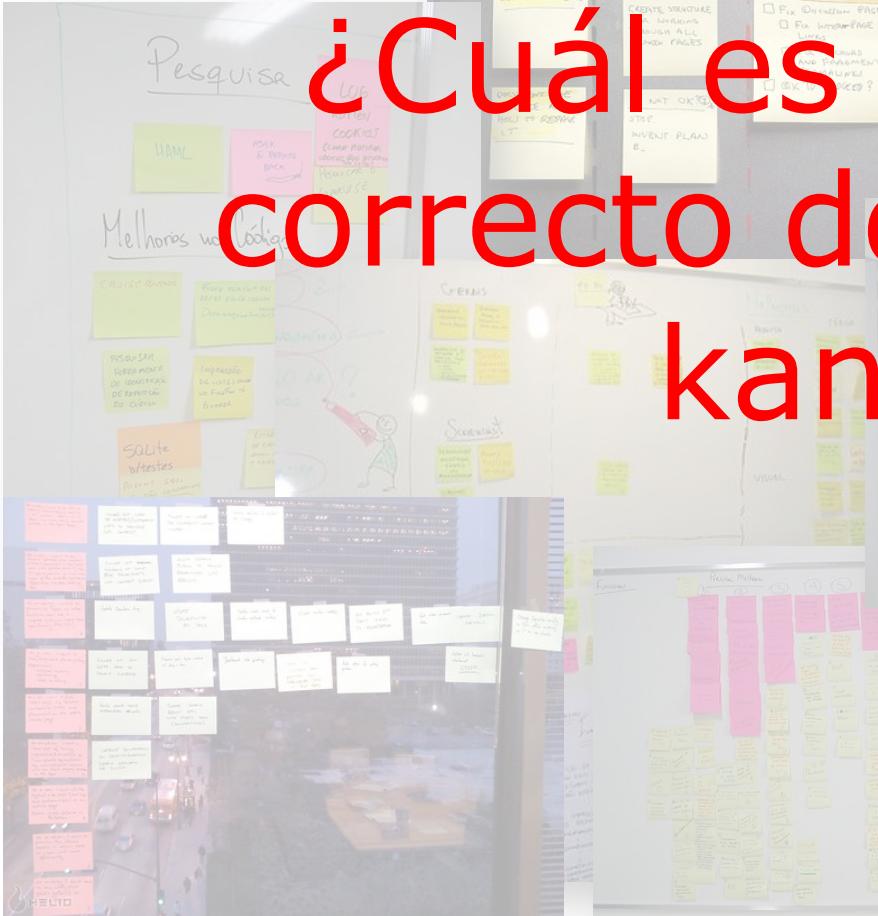


cc-by-sa ImproveIt

cc-by-sa [ImproveIt](#)

¿Cuál es el formato correcto de un tablero kanban?

READY | COOL WARM HOT | DOING



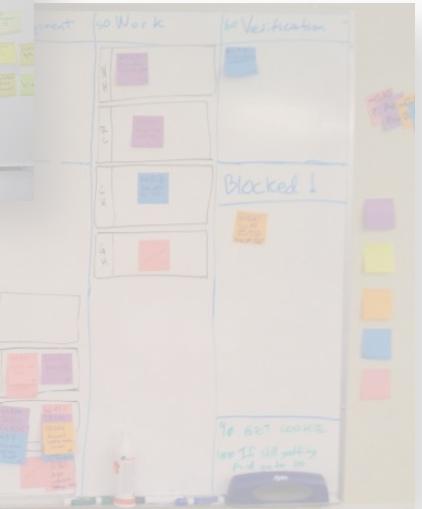
cc-by [Gregg](#)



cc-by-sa [improvet](#)



cc-by [improvet](#)



cc-by-sa [ImproveIt](#)

¿Cuál es el formato correcto de un tablero kanban?

Depende del uso, información que gestiona y preferencias del equipo

FLEXIBILIDAD

cc-by [improveit](#)

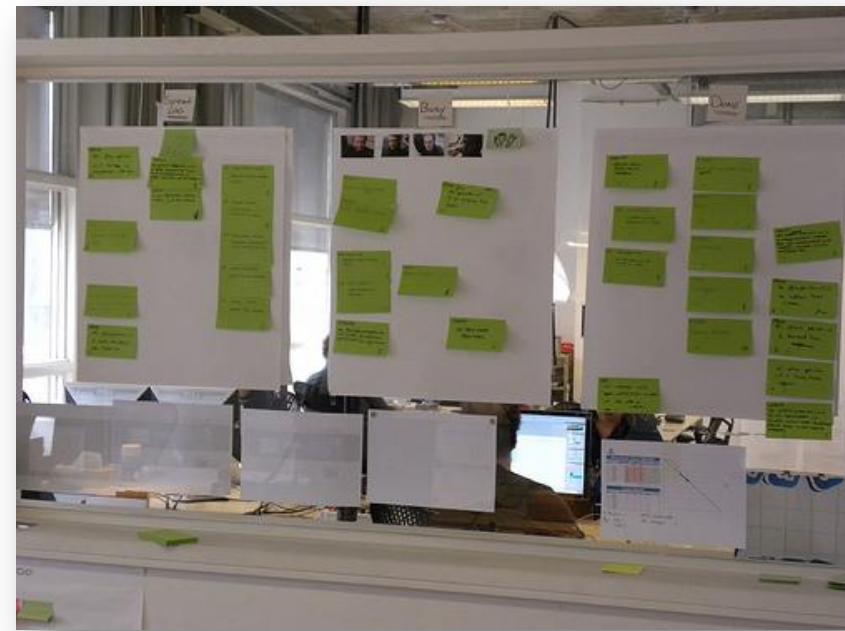
cc-by-sa

cc-by [Dennis Hamilton](#)

INFORMACIÓN – CONTROL – GESTIÓN VISUAL



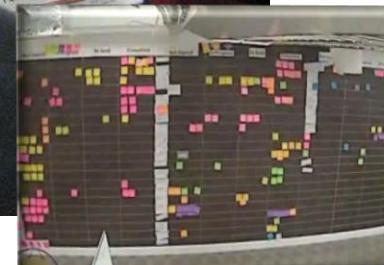
cc-by-sa HDN



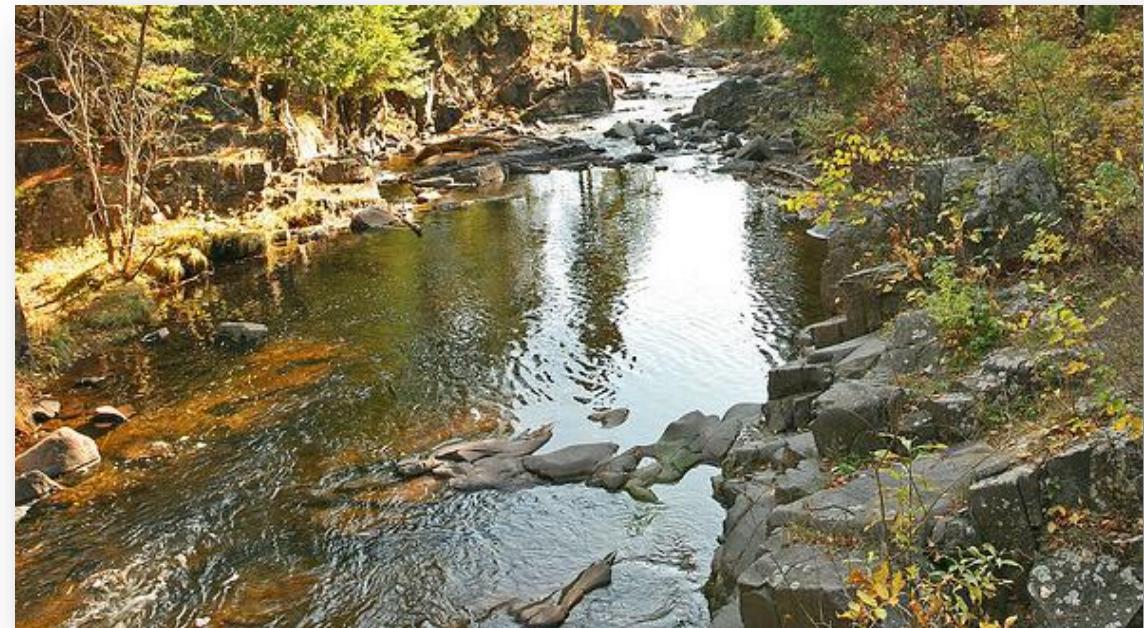
cc-by Rool Paap

VÍDEO

cc-by: Betsy Fletcher



Kanban para regular el
**FLUJO DE AVANCE
CONTINUO**



cc-by [Randen Pederson](#)

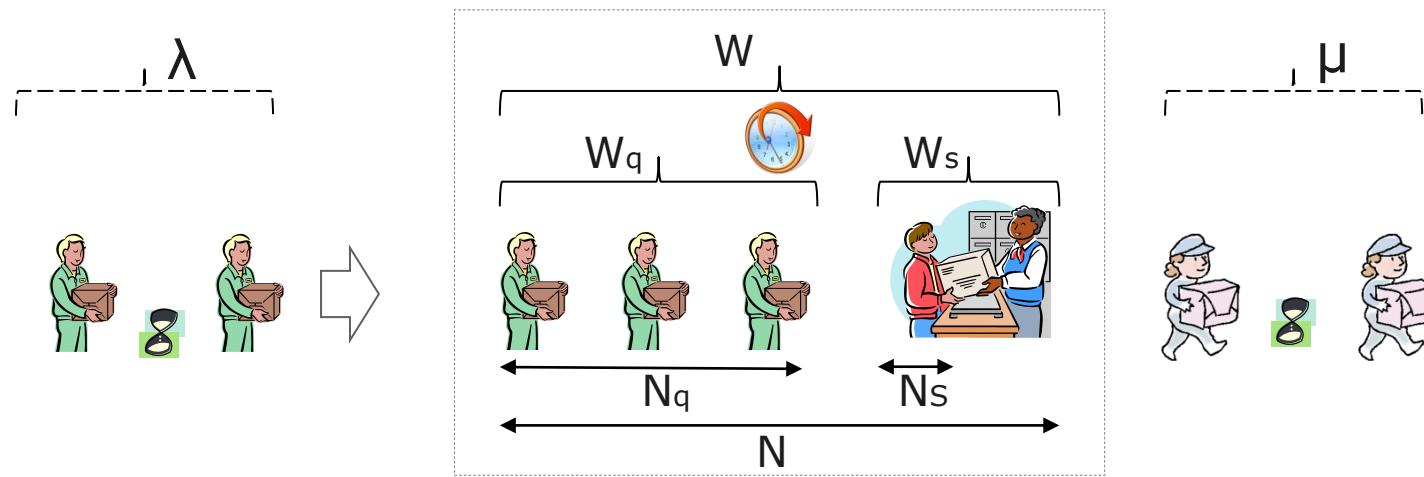
EJERCICIO



Fotografía cc-by: [LizMarie](#)



TEORÍA DE COLAS...



λ : Tasa media de entradas (nº op / unidad tiempo)

N_q : Operaciones en la cola

N_s : Operaciones siendo servidas

N : Operaciones en el sistema

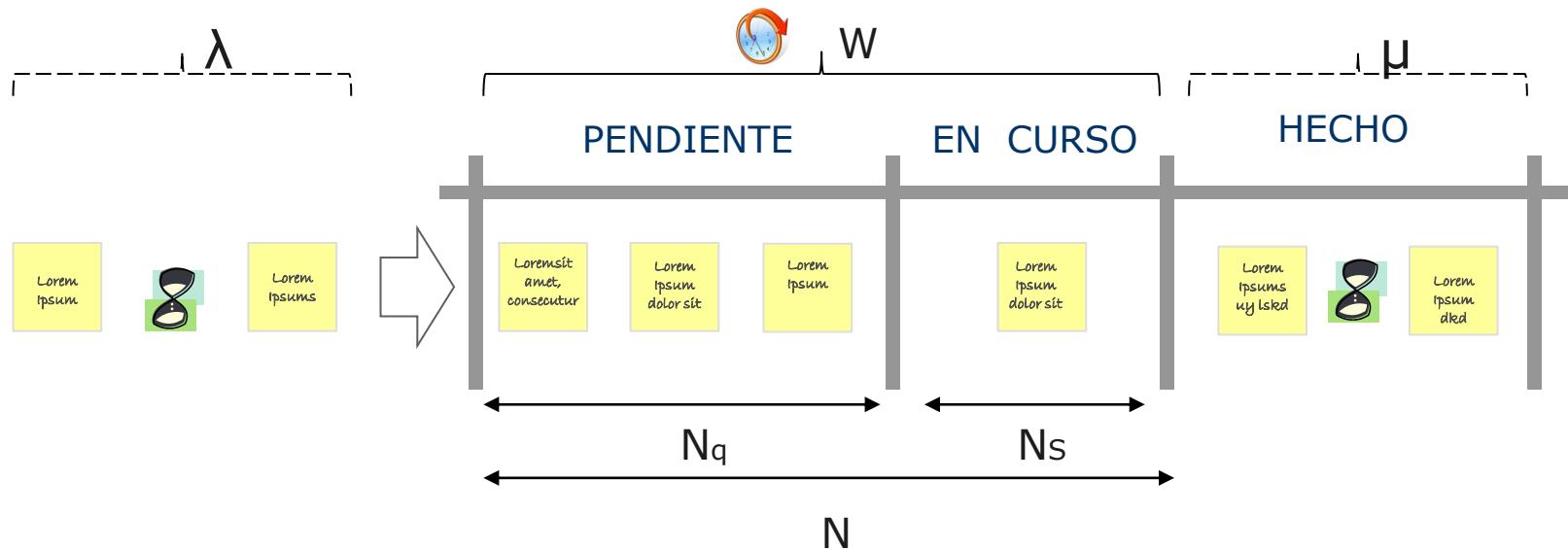
W_q : Tiempo en la cola

W_s : Tiempo se servicio operación

W : Tiempo total en el sistema

μ : Tasa media del servicio (nº op / unidad tiempo)

PARA AJUSTAR EL FLUJO DE UN TABLERO KANBAN



N_q : WIP de la pila

N_s : WIP de la fase de ejecución

N : WIP del sistema

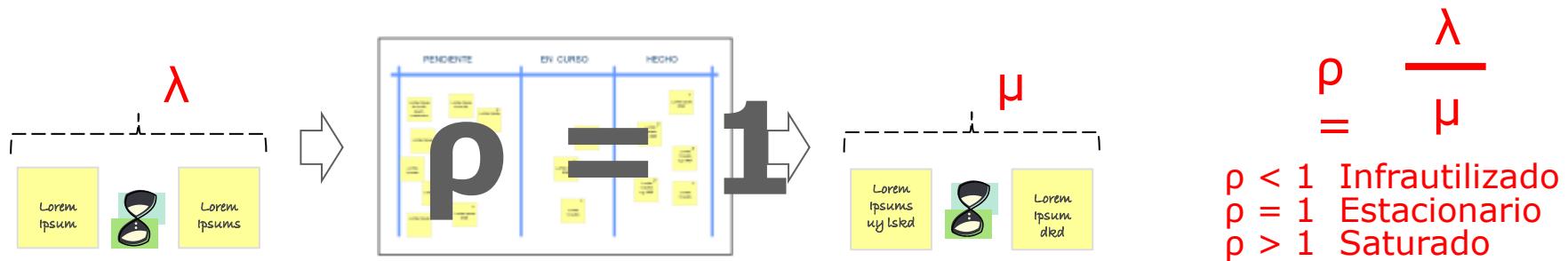
W : Lead time

λ : Ritmo de entrada

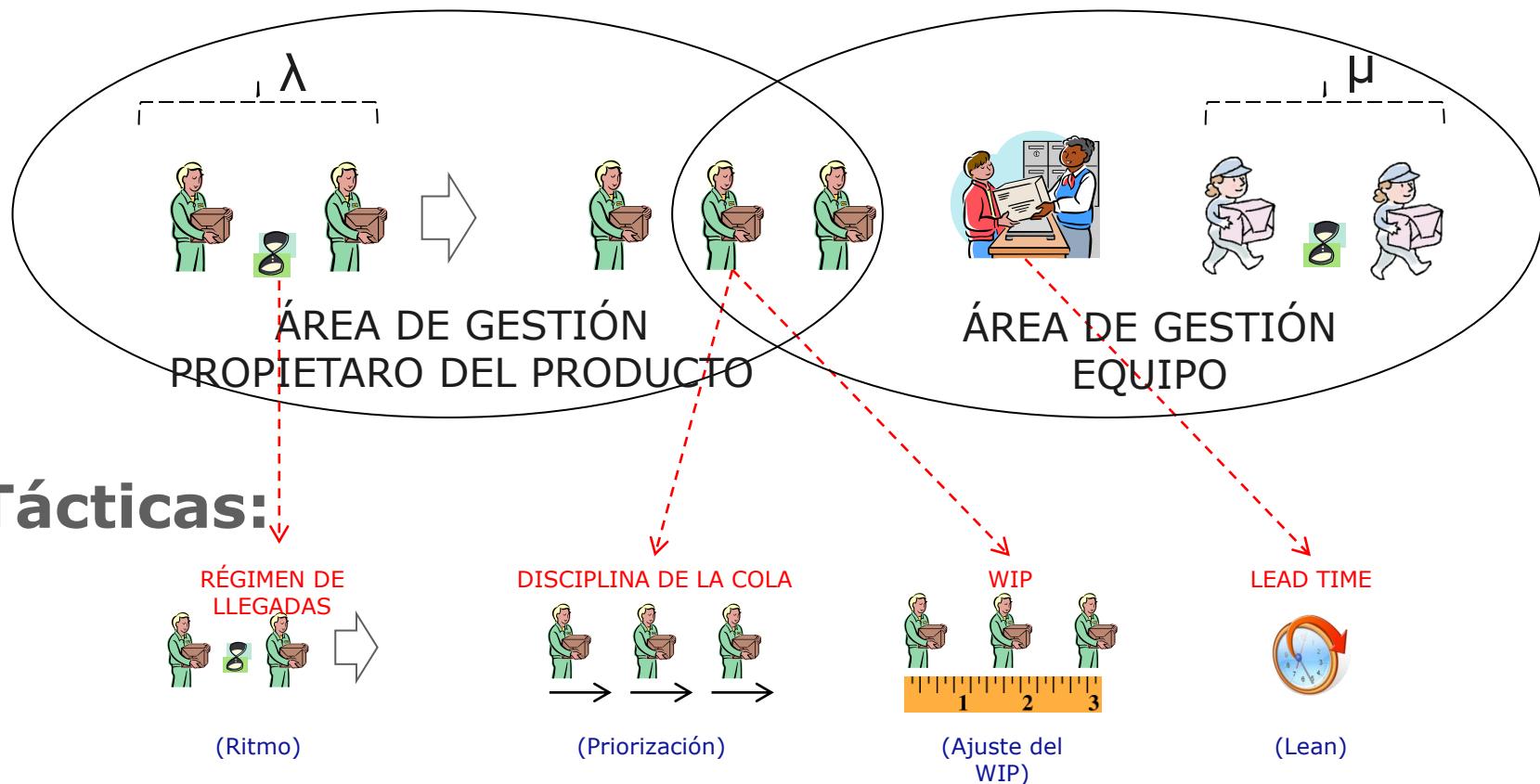
μ : Ritmo de salida

$$\rho \text{ (carga)} = \frac{\lambda}{\mu}$$

SISTEMA ESTACIONARIO DE FLUJO CONTINUO



... ACTUANDO SOBRE λ Y/O μ



EL PROPIETARIO DEL PRODUCTO

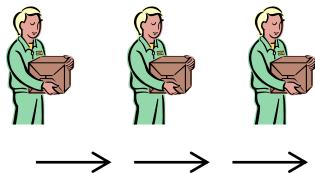
RÉGIMEN DE LLEGADAS



AJUSTANDO EL RITMO DE DEMANDA A LA CAPACIDAD DEL SISTEMA

Sin sobrepasar el WIP de entrada al sistema.

DISCIPLINA DE LA COLA



PRIORIZANDO LOS TRABAJOS. CRITERIOS POSIBLES:

- ? **FIFO** (First-In, First-Out) o FCFS (First-Come, First-Served). Las tareas se desarrollan en el orden van llegando.
- ✗ **LIFO** (Last-In, First-Out) o LCFS. Se hacen primero las últimas tareas en llegar.
- ✗ **SIRO** (Service In Random Order) Se atienden las tareas de forma aleatoria.
- ? **SIFO** (Shortest-in, First-out) o SJF (Shortest job first) Se da prioridad a las tareas que necesitan menos tiempo.
- ! **RR** (Round Robin, o turno robado). Se reparte el tiempo de trabajo por igual para atender a las tareas de la cola a la vez.
- ? **PR**. Según la prioridad.
 - Con interrupción. Al llegar una tarea de más prioridad se interrumpe la que está en curso
 - Sin interrupción. Al terminar la tarea en curso se atiende a la de mayor prioridad.



Inapropiado para gestión de tareas



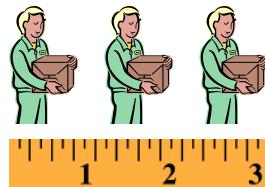
Incrementa el "lead time"



A evaluar por el propietario del producto

EL EQUIPO

WIP



ESTABLECIENDO LÍMITES WIP ADECUADOS ALA CAPACIDAD DEL SISTEMA

Establecer un WIP adecuado al tiempo de proceso aceptable por el cliente

- 1.- ¿ Tiempo de proceso del sistema (lead time) deseado por el cliente?
- 2.- $\mu =$ capacidad por unidad de tiempo (velocidad o nº de tareas en ese lead time)
- 2.- WIP del sistema aconsejado $> \mu < 2 \mu$

Ley de Little $L = \lambda * W$ En un sistema en equilibrio $\lambda = \mu \Rightarrow L = \lambda * W \Rightarrow$

WIP (μ) = Ritmo de salida * Lead time

WIP (2 μ) = Ritmo de salida * Lead time *2

TÉCNICAS LEAN



MEJORANDO EL TIEMPO DE PROCESO (LEAD TIME)

Prácticas "Lean Software Development" para la reducción de "desperdicio" en el proceso de desarrollo

LA ORGANIZACIÓN

NÚMERO DE SERVIDORES



DIMENSIONANDO EL SISTEMA ADECUADAMENTE

- ¿La situación de saturación ($\rho > 1$) es temporal o habitual?
- El incremento de personal en organizaciones de software debe ser parte del plan de crecimiento de la organización. La incorporación urgente como medida de contingencia para proyectos retrasados es contraproducente.
- En proyectos TIC los equipos pequeños son más productivos. Es recomendable emplear estructuras organizativas fractales.

FORTALEZAS DE LA GESTIÓN VISUAL KANBAN

GESTIÓN CONTROL

- Regula el flujo y la carga de trabajo / equipo
- Facilita un flujo de trabajo que lleva los problemas a la superficie
- Facilita el mantenimiento de un ritmo sostenido (evita la ley de Parkinson)
- Desarrollo continuo

INFORMACIÓN VISUAL

- Favorece la comunicación directa.
- Los problemas se detectan enseguida y no quedan ocultos.
- Favorece una cultura de colaboración y resolución.

CONCEPTOS LEAN APLIABLES A PROYECTOS TIC

Ajuste a través de los tres conceptos de la mejora continua kaizen

- **Muda**: Desperdicio.
- **Muri**: Tensión - Sobrecarga de trabajo que produce cuellos de botella.
- **Mura**: Discrepancia - Variabilidad del flujo de trabajo.

Mudas habituales en proyectos TIC

Burocracia	Procedimientos, documentación y papeleo innecesario que no aporta valor al resultado
Sobreproducción	Desarrollar más características de las necesarias
Multiproyecto	Cambio de proyecto / interrupciones del flujo de trabajo
Esperas	Tiempos de espera por falta de cadencia en el flujo de trabajo.
“Ir haciendo”	(Falsa solución de las esperas). Encargar trabajo para ir avanzando algo no definido y no tener paradas a las personas.
Desajustes de capacidad	Personas de gran talento asignadas a tareas rutinarias y viceversa.
Errores	Retrabajo por bugs.

EJERCICIO: DISEÑO DE TABLEROS KANBAN



Fotografía cc-by: [LizMarie](#)

Práctica 1

Información en el área de producto

Kanban (Conceptos kanban y lean en gestión ágil)



Información visual
Gestión visual

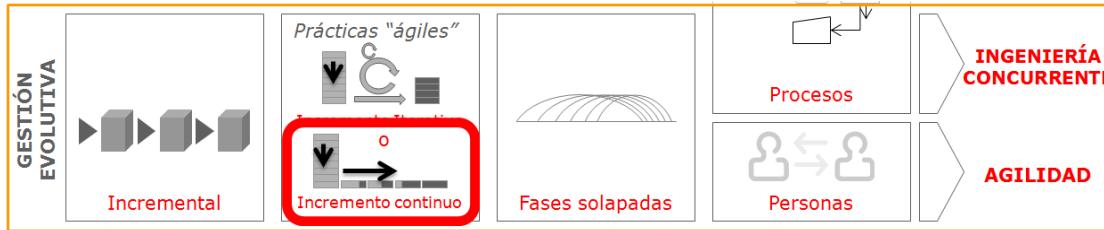
Posibles tableros mostrarán visualmente la siguiente información relativa al estado de desarrollo del producto.

- Posibles historias de usuario sugeridas que se encuentran en evaluación sin determinar aún si se incorporarán al producto.
- Historias de usuario aprobadas: se incorporarán al producto.
- Historias de usuario ya valoradas y priorizadas previstas para programar.
- Historias de usuario que se están programando actualmente.
- Historias de usuario ya programadas y que se pueden evaluar y ver en el servidor de pruebas.
- Historias de usuario ya evaluadas pendientes de desplegar.
- Historias de usuario desplegadas en las dos últimas versiones.

Práctica 2

Desarrollo de producto

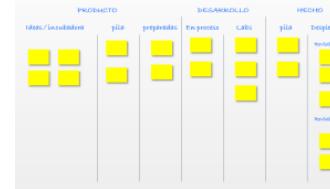
En incremento continuo



Possibles tableros para representar y guiar la gestión de trabajo de un equipo de desarrollo que reflejen:

- Pila de tareas.
- Tareas ya estimadas y preparadas para entrar a desarrollo.
- Tareas en análisis.
- Tareas en codificación.
- Tareas terminadas.
- Tareas integradas en el servidor de desarrollo (labs)
- Tareas integradas en producción.

Kanban (Conceptos kanban y lean en gestión ágil)



Información visual
Gestión visual

Práctica 3

Desarrollo de producto

En incremento iterativo



Kanban (Conceptos kanban y lean en gestión ágil)



Información visual
 Gestión visual

Possibles tableros para representar y guiar la gestión de trabajo de un equipo de desarrollo que reflejen:

- Pila de tareas.
- Tareas ya estimadas y preparadas para entrar a desarrollo.
- Tareas en análisis.
- Tareas en codificación.
- Tareas terminadas.
- Tareas integradas en el servidor de desarrollo (labs)
- Tareas integradas en producción.

Práctica 3

Equipo de operación y mantenimiento

Kanban (Conceptos kanban y lean en gestión ágil)



Posibles tableros para representar y guiar la gestión de trabajo de un equipo de operación y mantenimiento que refleje:

- Estado de las tareas programadas para la semana y persona que está trabajando con cada una.
- Estado de incidencias no previstas y urgentes y persona que está trabajando con cada una.

Gestión en organizaciones ágiles

1.0

cc-by [Roel Paap](#)

VÍDEO: ¿Gestión predictiva?

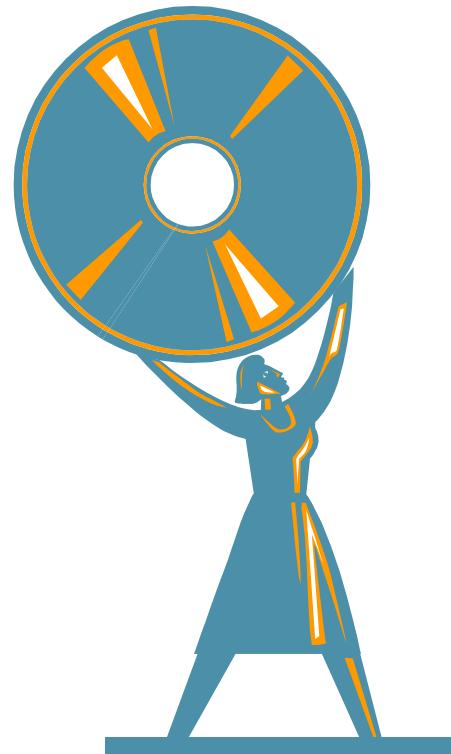
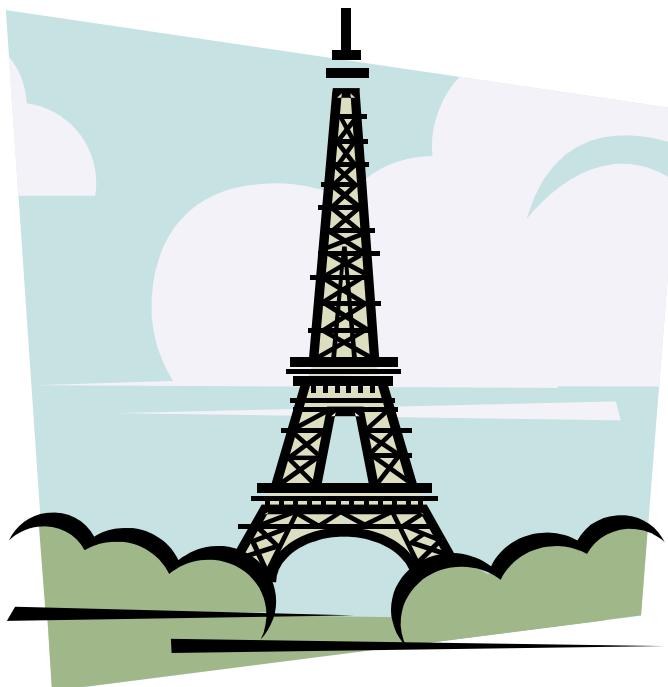


cc-by: [Betsy Fletcher](#)

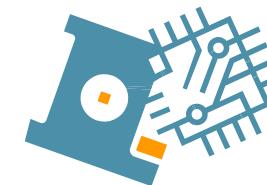
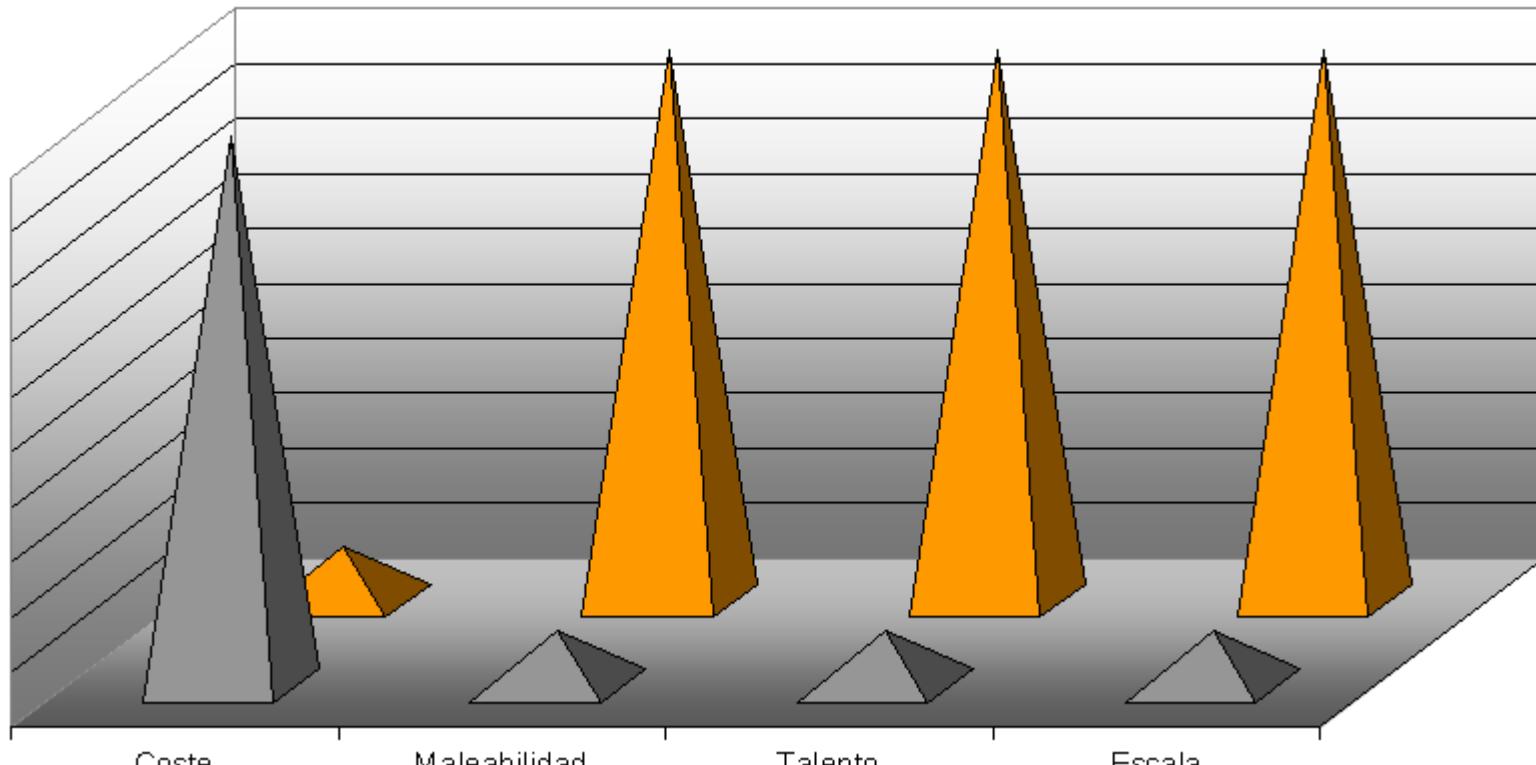
Vídeo: World Builder – Bruce Braint



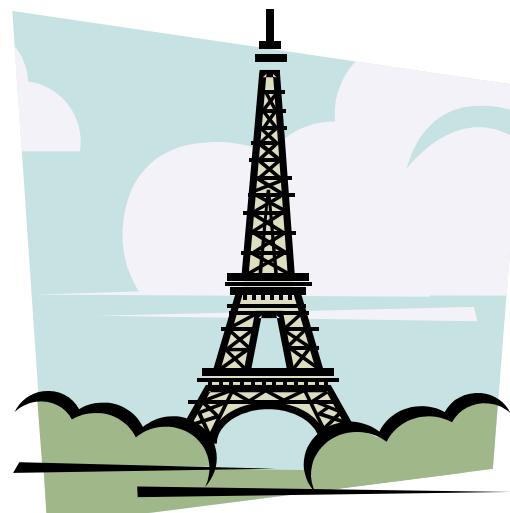
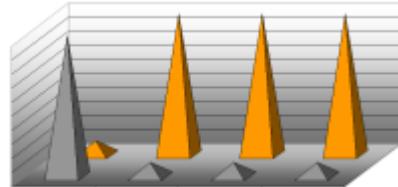
LAS CARACTERÍSTICAS DEL SOFTWARE



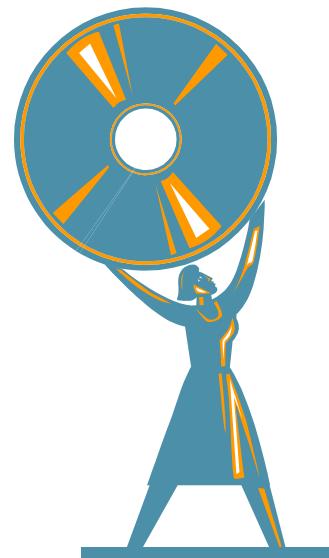
LAS CARACTERÍSTICAS DEL SOFTWARE



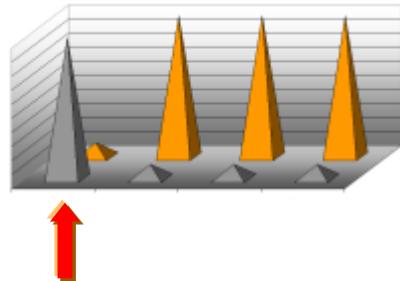
LAS CARACTERÍSTICAS DEL SOFTWARE



COSTE



LAS CARACTERÍSTICAS DEL SOFTWARE



COSTE



PROTOTIPADO

NO REHACER - PLANIFICAR

PROBAR Y EXPLORAR

LAS CARACTERÍSTICAS DEL SOFTWARE

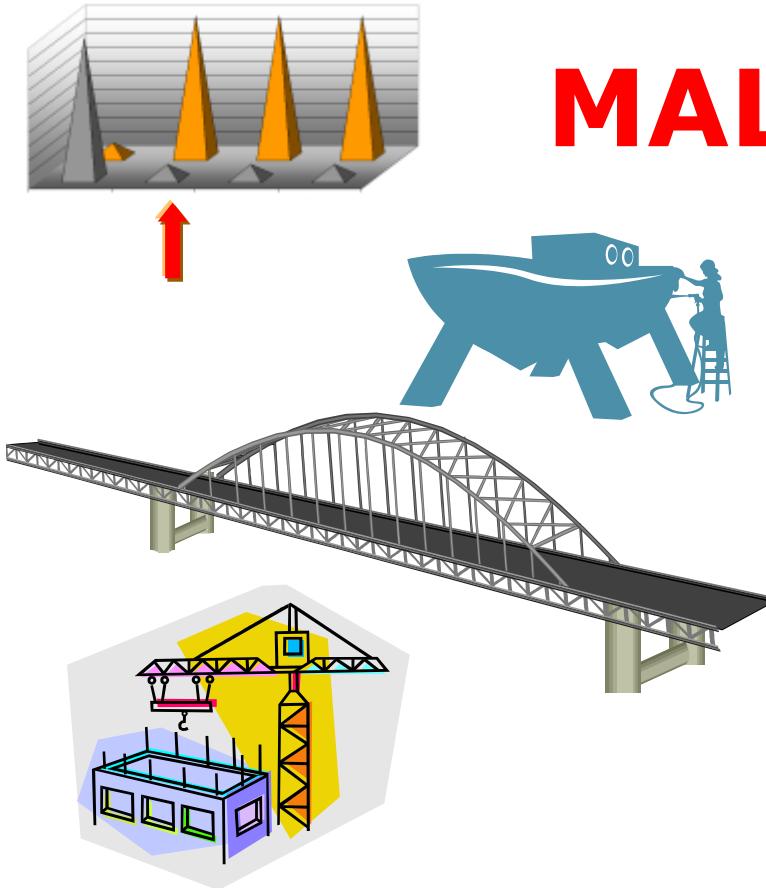
COSTE / BENEFICIO



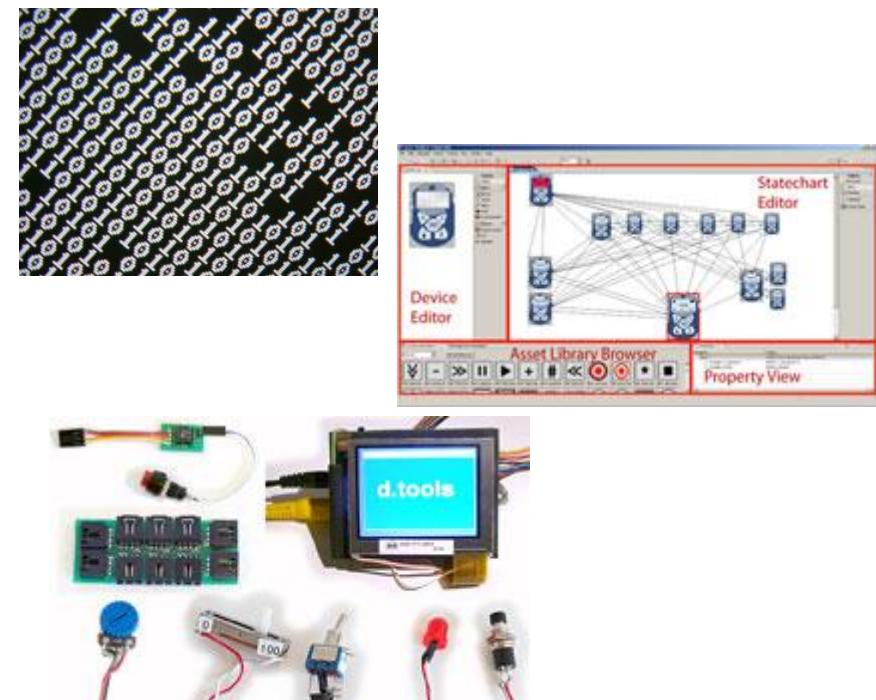
NO REHACER - PLANIFICAR

PROBAR Y EXPLORAR

LAS CARACTERÍSTICAS DEL SOFTWARE



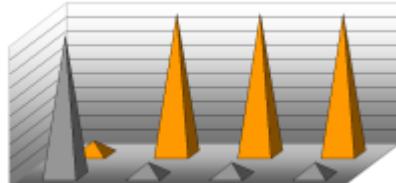
MALEABILIDAD



NO REHACER - PLANIFICAR

PROBAR Y EXPLORAR

LAS CARACTERÍSTICAS DEL SOFTWARE



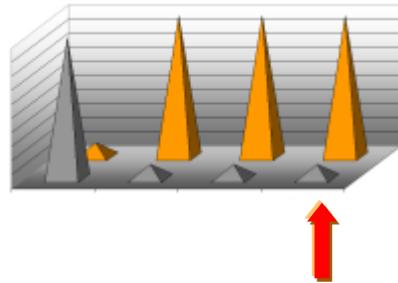
Tiempos y métodos
Optimización de procesos
Organización científica del trabajo

TALENTO

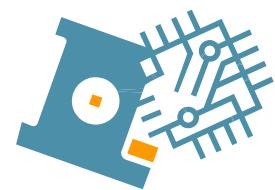
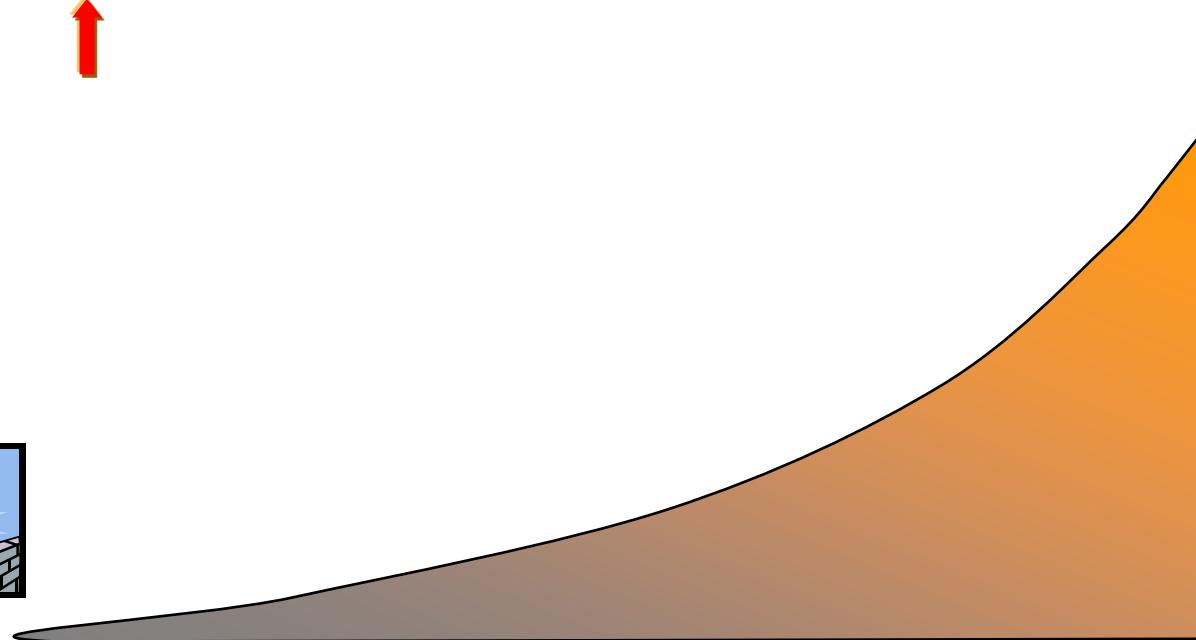


Gestión del talento
Excelencia
Cultura y entorno

LAS CARACTERÍSTICAS DEL SOFTWARE



ECONOMÍA DE ESCALA



EFICIENCIA

VALOR

CRITERIOS DE LA ESTRATEGIA PREDICTIVA / INDUSTRIAL



**Gestión de
proyectos
predictiva**



**Producción
basada en
procesos**

PRÁCTICA: LOS PROCESOS



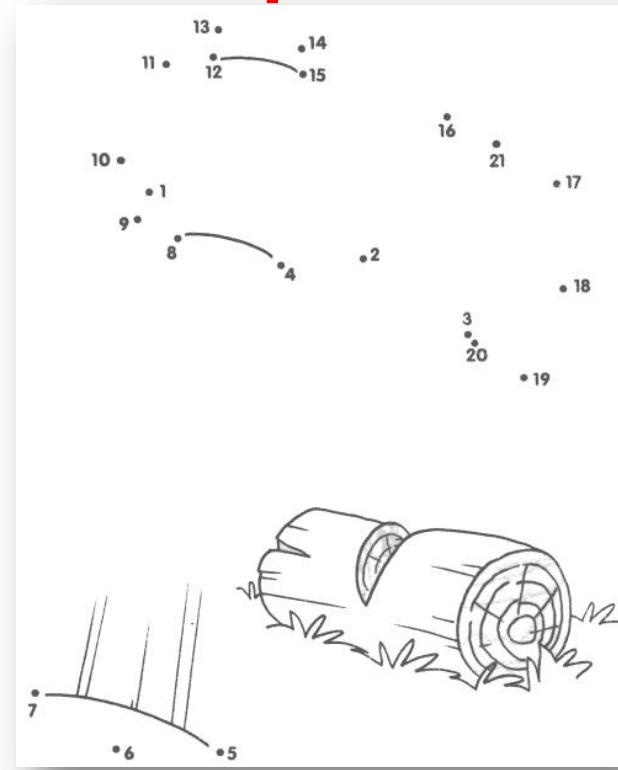
cc-by: [LizMarie](#)

LOS PROCESOS COMO RESPONSABLES DEL RESULTADO

Tácito



Explícito



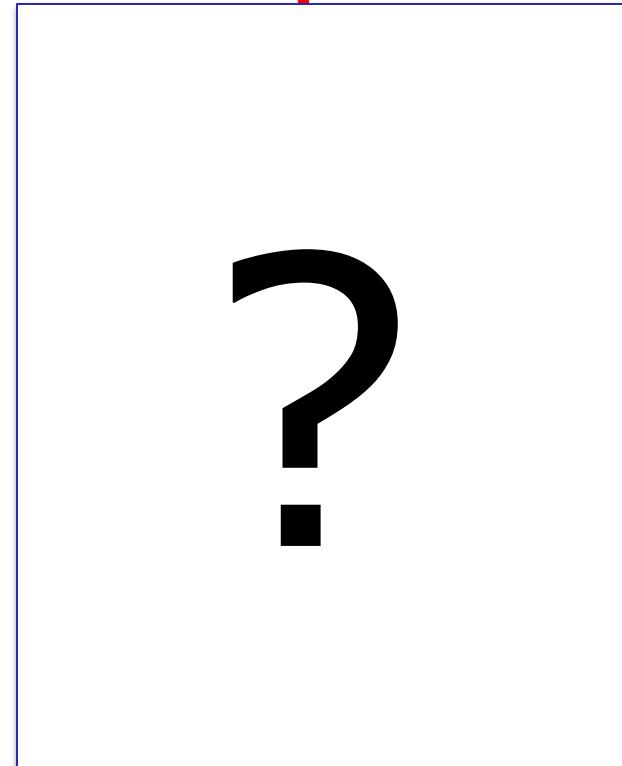
Externalización

¿LOS PROCESOS COMO RESPONSABLES DEL RESULTADO?

Táctico

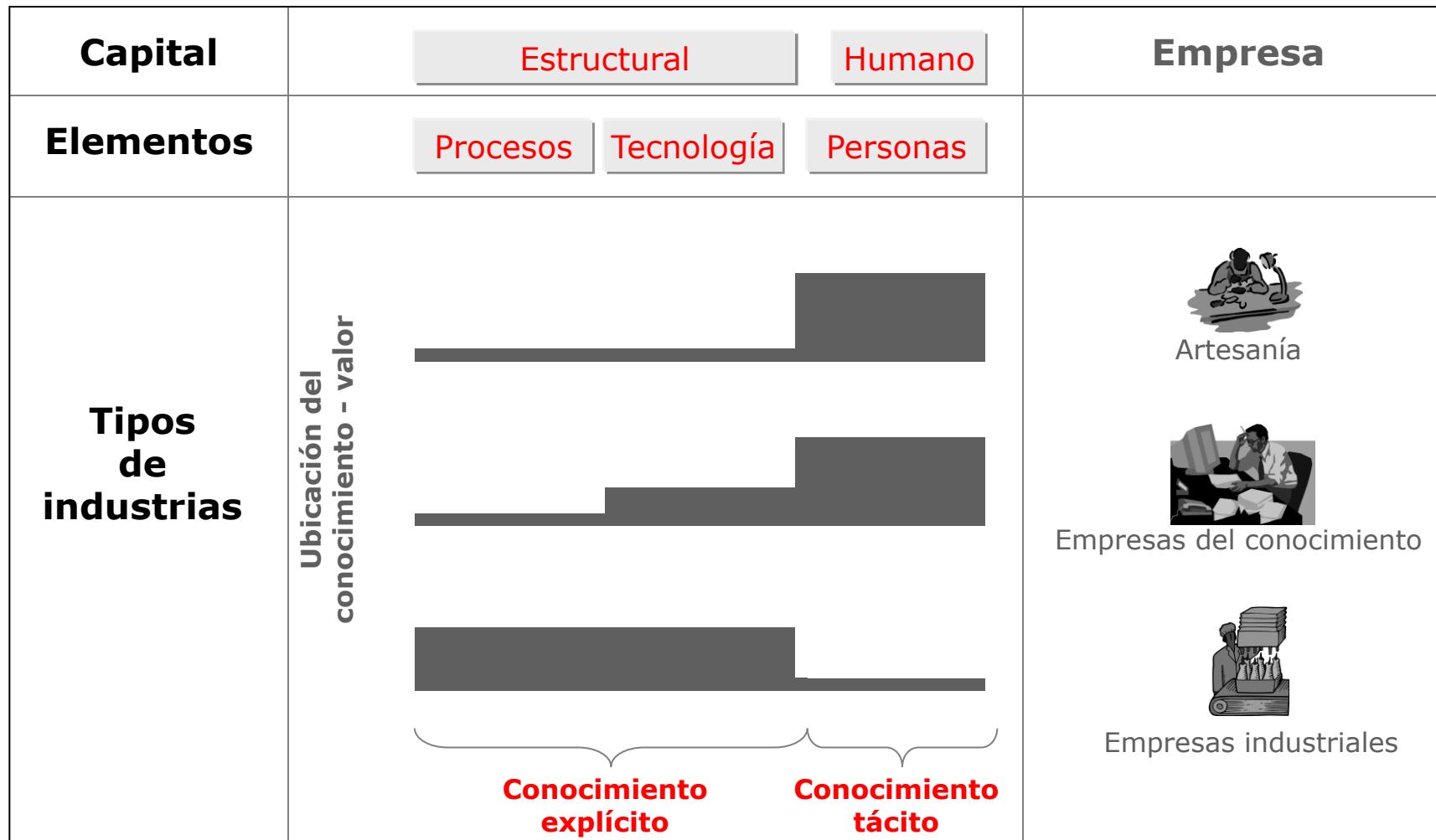


Explícito



Externalización

KNOW HOW



PROCESOS COMO RESPONSABLES DEL RESULTADO



The image shows a grid of six cards, each containing a question and an answer. The cards are arranged in two rows of three. The top row contains:

- A card with the letters "aeio" in cursive script, with the question "¿Cuántos tienes?" below it.
- A card with the word "SAS" repeated in blue and orange, with the question "¿Qué son?" below it.
- A card with a large number "2" crossed out with a red X, with the question "Queso italiano" below it.

The bottom row contains:

- A card with a large letter "A" where the red dot is replaced by the word "CAD", with the question "Me ha regalado una..." below it.
- A card with four playing cards (two aces and two spades) and the word "NO" in red, with the question "Fue condenado por..." below it.
- A card with a large letter "g" inside a circle, with the question "¿Cómo se llama su padre?" below it.

Below the grid, there are faint, overlapping images of other cards with questions like "¿Puedes ver el sol?", "¿Cuál es la diferencia entre un obelisco y un obelisco?", and "¿Qué es un cuadro de colores?".

ESCENARIO ACTUAL

VELOCIDAD



Fotografía cc-by:[Amnemona](#)

ESCENARIO ACTUAL

INCERTIDUMBRE

Fotografía cc-by: [Frédéric Dupont](#)

EJERCICIO: VELOCIDAD



Fotografía cc-by: [LizMarie](#)

VELOCIDAD



1930 - 1940 - 1950 - 1960 - 1970 - 1980 - 1990 - 2000

VELOCIDAD



1930 - 1940 - 1950 - 1960 - 1970 - 1980 - 1990 - 2000
VELOCIDAD

VELOCIDAD



1930 - 1940 - 1950 - 1960 - 1970 - 1980 - 1990 - 2000
VELOCIDAD



EJERCICIO: INCERTIDUMBRE



Fotografía cc-by: LizMarie

VELOCIDAD / INCERTIDUMBRE



INCERTIDUMBRE
1930 - 1940 - 1950 - 1960 - 1970 - 1980 - 1990 - 2000
VELOCIDAD

THE NEW NEW PRODUCT DEVELOPMENT GAME

Muchas compañías han descubierto que para mantenerse en el actual mercado competitivo necesitan algo más que los conceptos básicos de calidad elevada, costes reducidos, diferenciación.

Además de esto también es necesario velocidad y flexibilidad

Nonaka y Takeuchi, 1986



Harvard Business Review

EL ESCENARIO ACTUAL

Eficiencia y previsibilidad ya no son las claves del éxito.

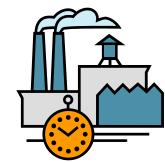
1960

1985

Lanzamiento constante de novedades



Reducción del tiempo de desarrollo



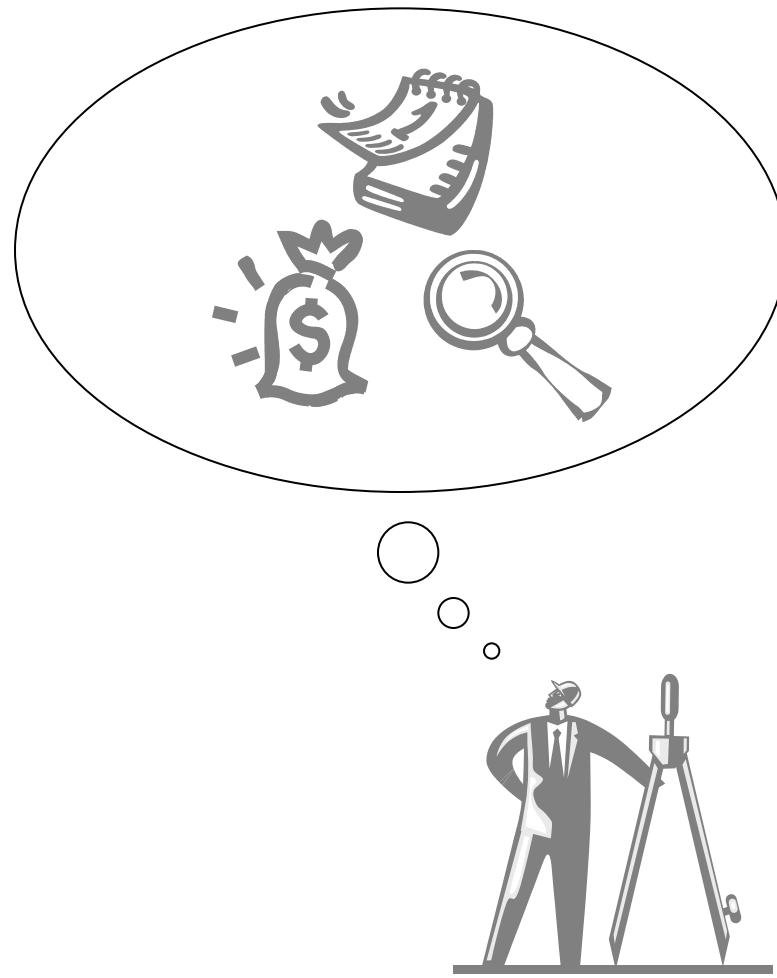
Cambios y mejoras rápidas



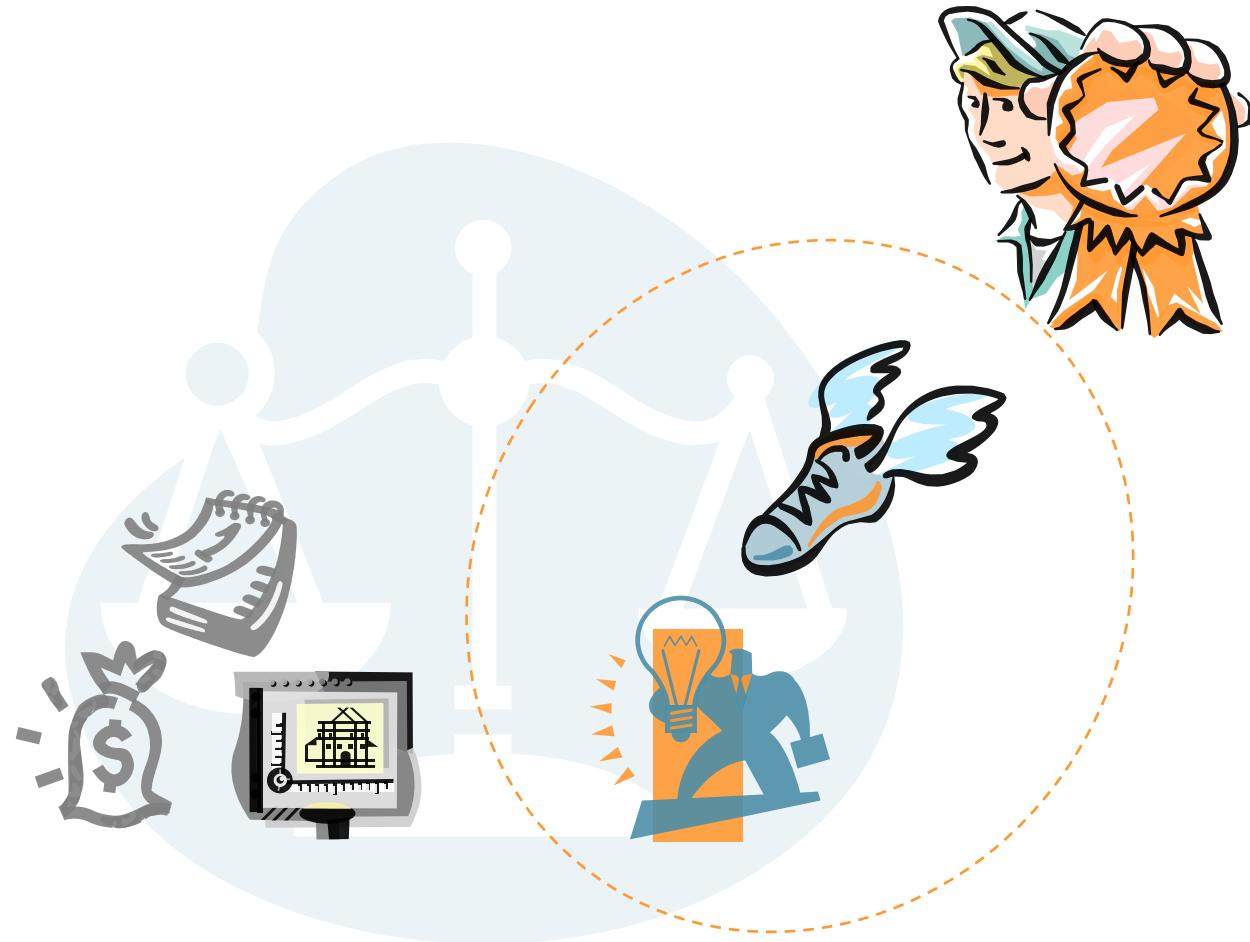
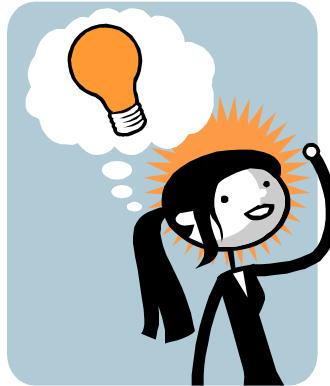
Componente innovador



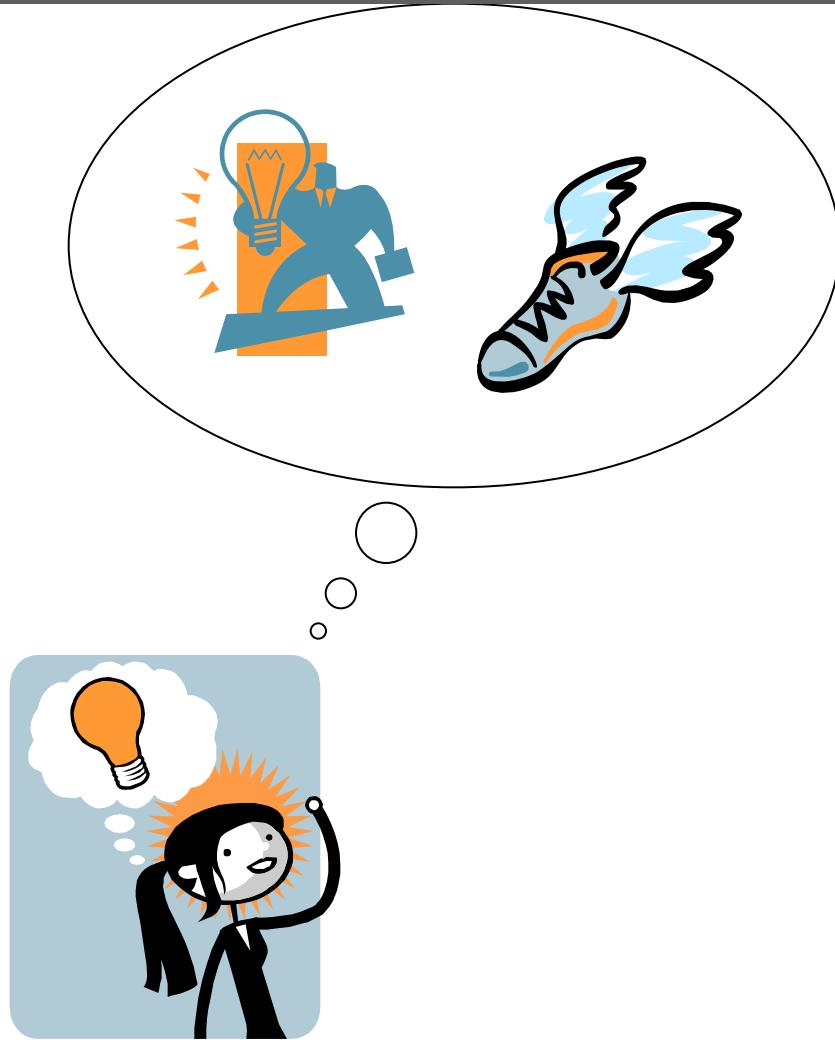
DAMO PREVISIÓN Y CALIDAD CON PROCESOS



LO QUE NECESITA EL CLIENTE



VALOR RÁPIDO Y CONTINUO



VÍDEO: FLEXIBILIDAD



cc-by: Betsy Fletcher

Vídeo: Club de los poetas muertos



FLEXIBILIDAD EN LAS PRÁCTICAS

PRINCIPIO ÁGIL

**Seguimiento próximo
(diario)**

PRÁCTICAS ÁGILES

Burn-down
Kanban de tareas
Gráfico de carrera

...

Fundamentalismo (RAE): Exigencia intransigente de sometimiento a una doctrina o práctica establecida

FLEXIBILIDAD EN LAS PRÁCTICAS

PRINCIPIO ÁGIL

**Seguimiento próximo
(diario)**

PRÁCTICAS ÁGILES

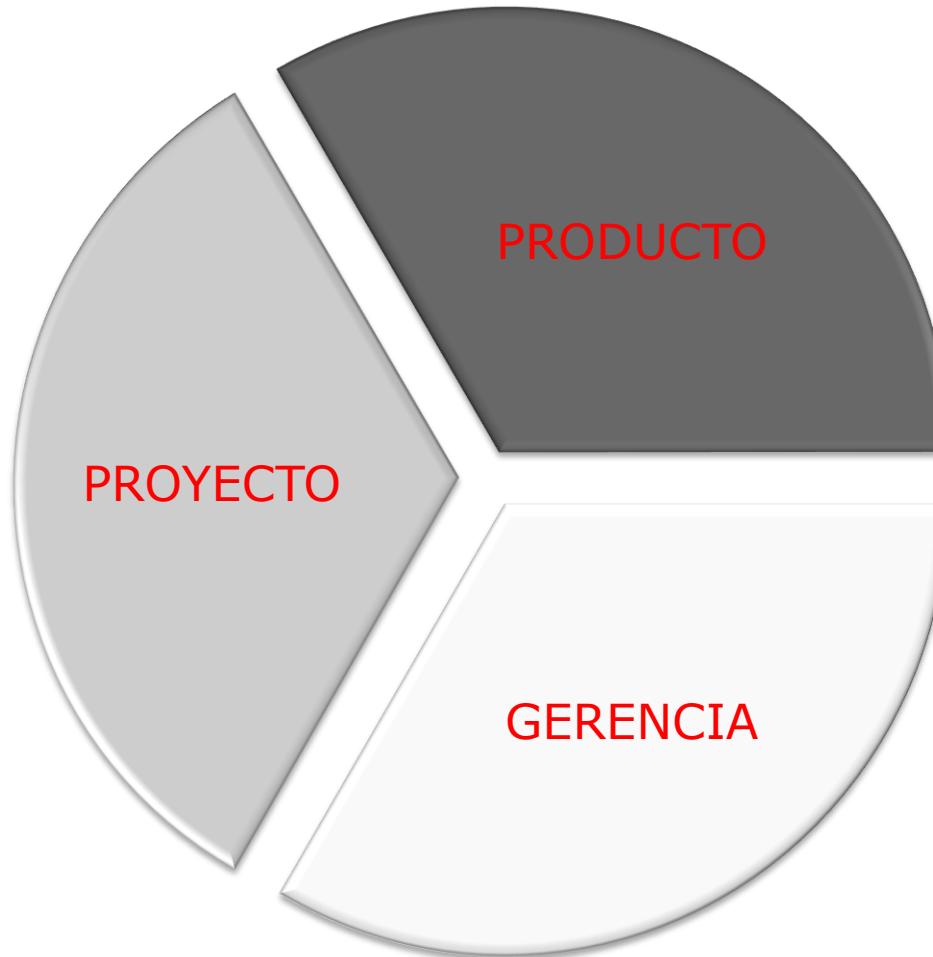
**Burn-down
Kanban de tareas
Gráfico de carrera**

...

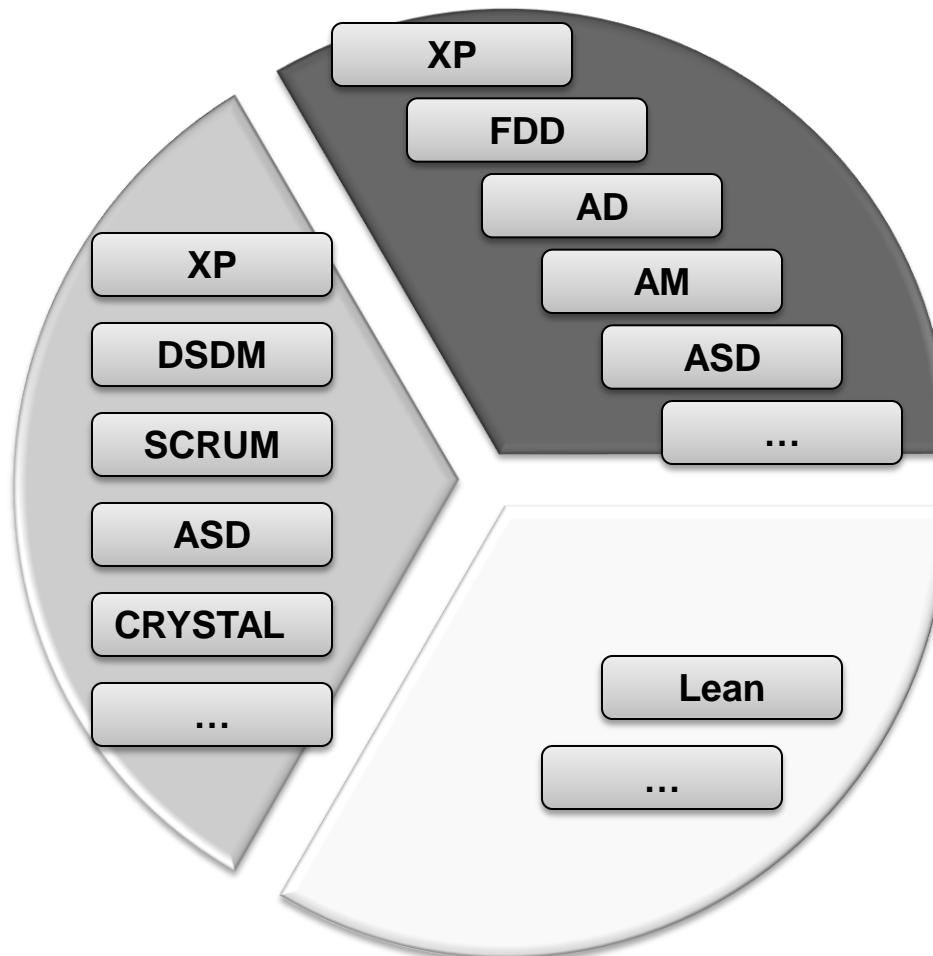
FLEXIBILIDAD

Fundamentalismo (RAE): Exigencia intransigente de sometimiento a una doctrina o práctica establecida

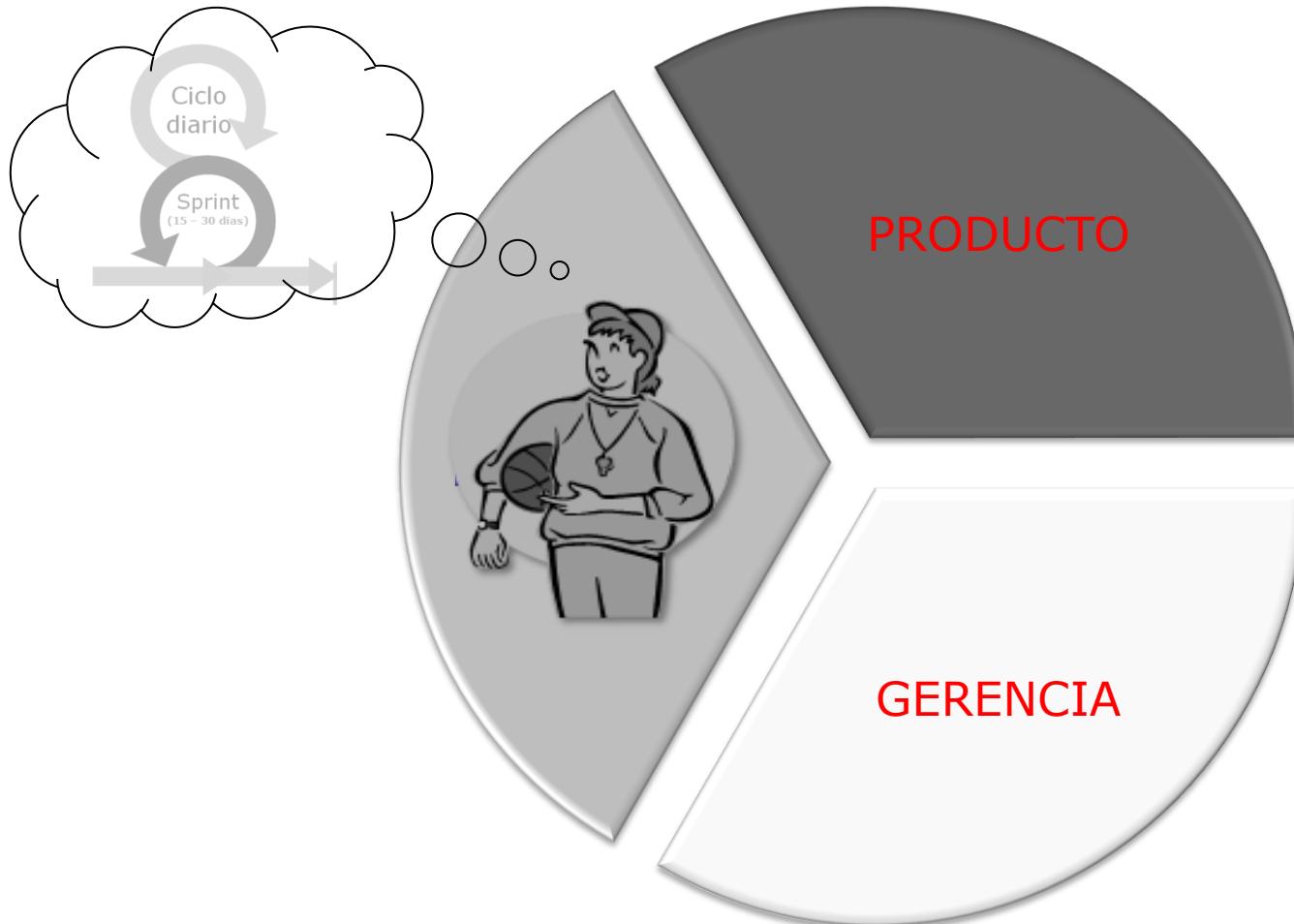
AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN



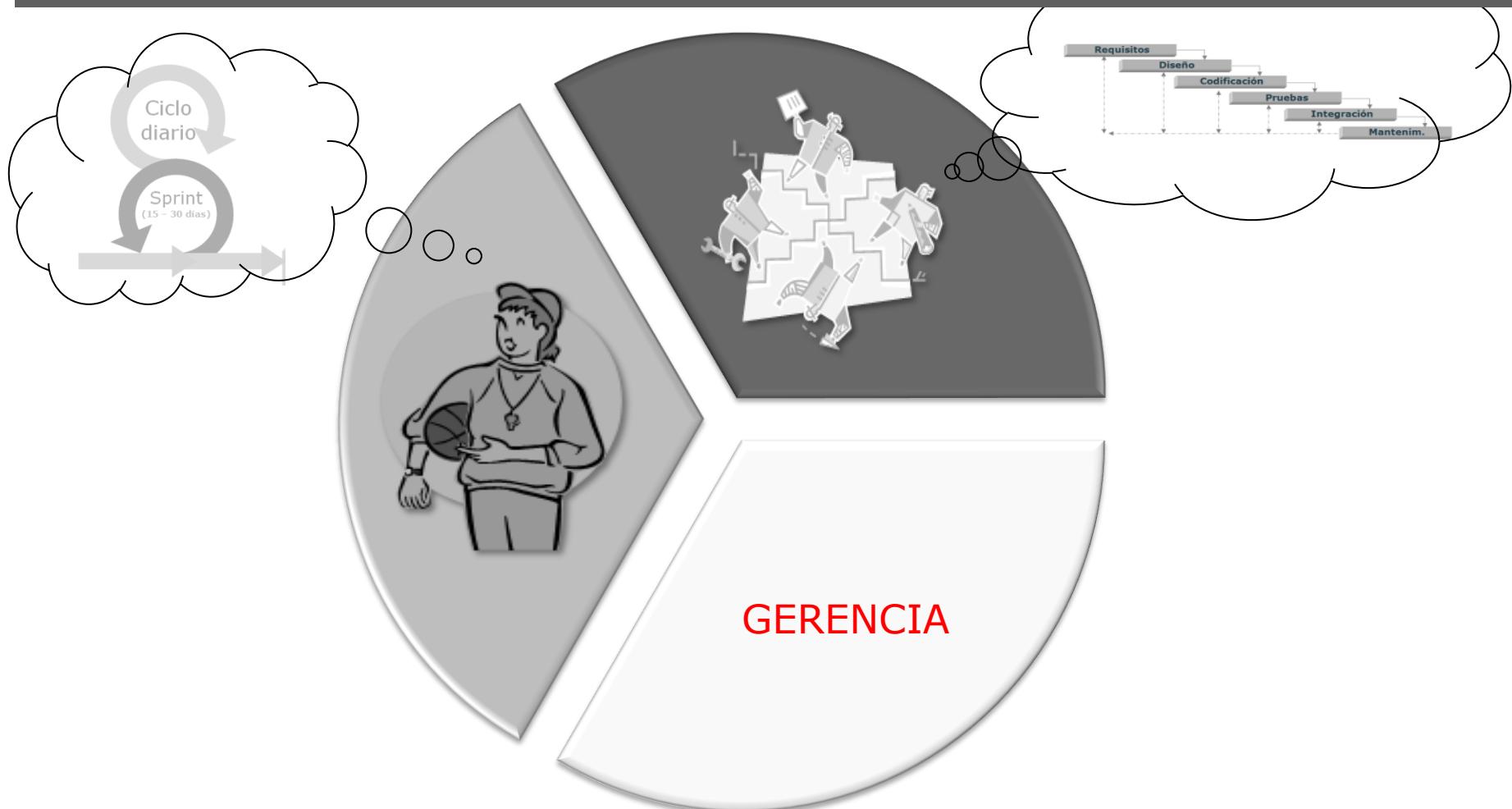
AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN



AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN



AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN

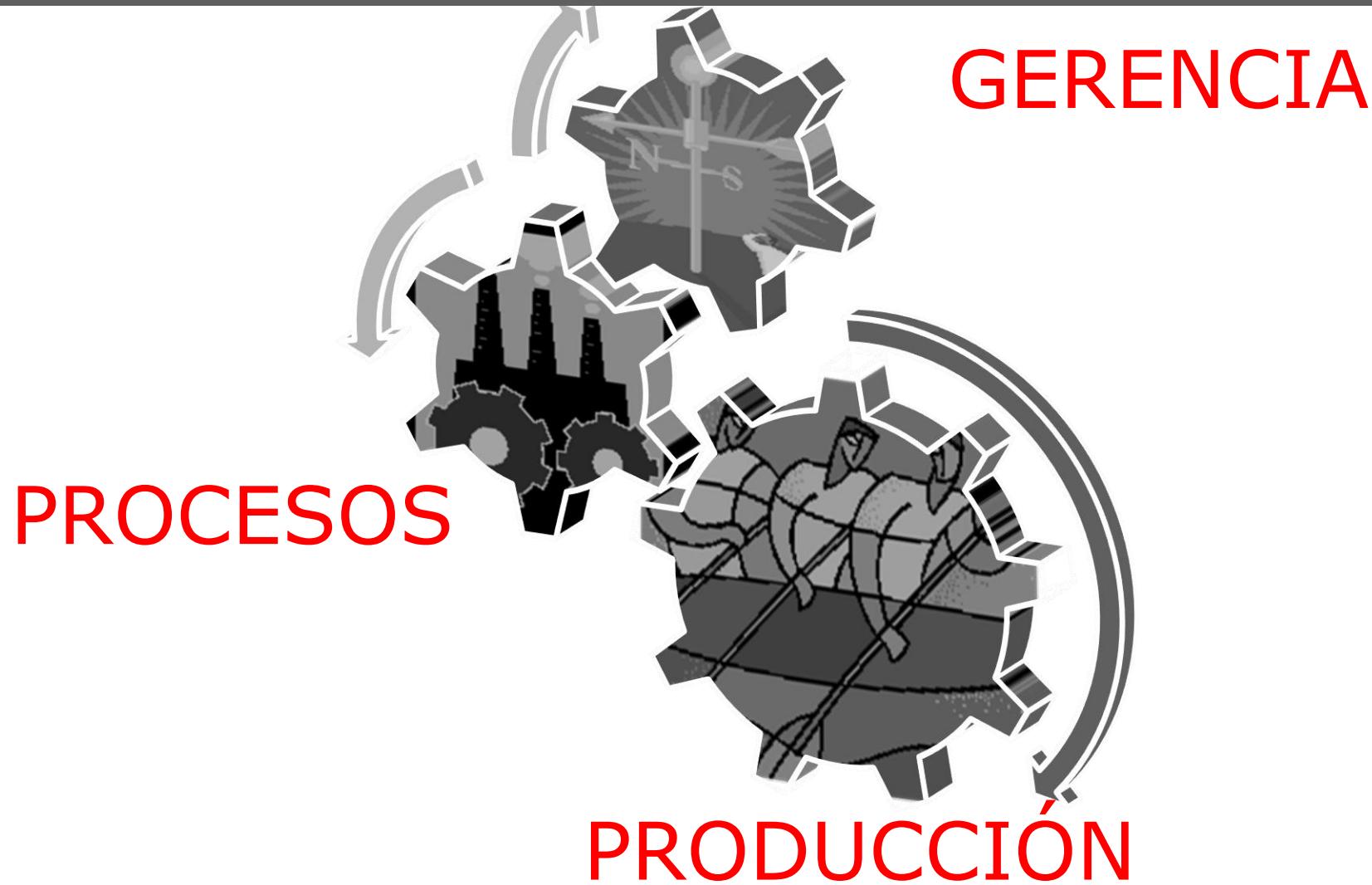


AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN



AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN



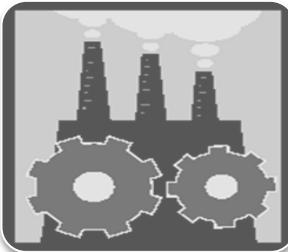
AGILIDAD: GESTIÓN Y ALINEACIÓN DE LA ORGANIZACIÓN

ÁREAS DE RESPONSABILIDAD



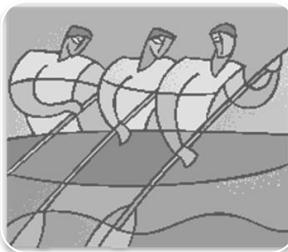
Gerencia

- Equilibrio sistémico de la empresa
- Coherencia del modelo
- Medios y formación



Procesos / calidad

- Configuración de Scrum
- Mejora continua
- Garantía de funcionamiento de Scrum



Ejecución

- Visión del producto
- Autoorganización
- Tecnología ágil

ÁREAS DE RESPONSABILIDAD / ROLES SCRUM

Dirección



Equilibrio sistémico de la empresa

Coherencia del modelo

Medios y formación

Scrum Manager



Configuración de Scrum

Mejora continua

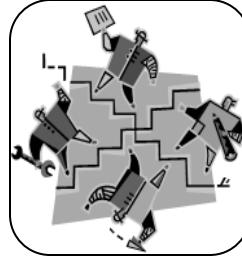
Garantía de funcionamiento de Scrum

Propietario producto



Visión del producto

equipo



Autoorganización

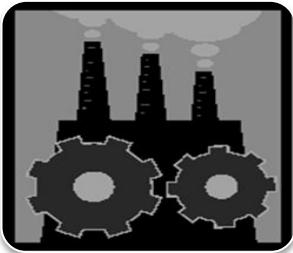
Tecnología ágil

IMPLANTACIÓN DE SCRUM BASADA EN RESPONSABILIDADES



Management

- Equilibrio sistémico de la empresa
- Coherencia del modelo
- Medios y formación



Procesos / calidad

- Configuración de Scrum
- Mejora continua
- Garantía de funcionamiento de Scrum

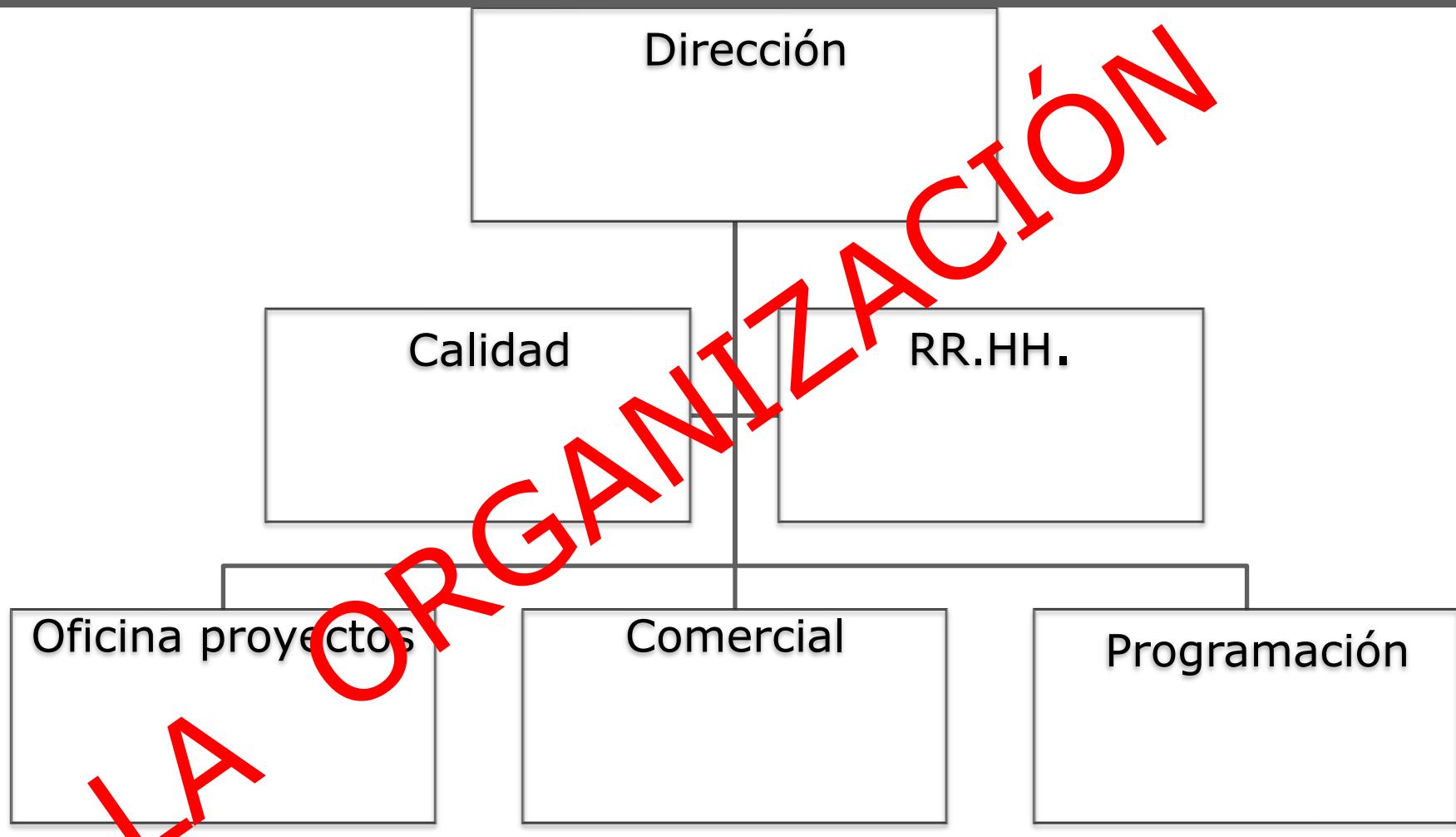


Ejecución

- Visión del producto
- Auto-organización
- Tecnología ágil

RESPONSABILIDADES

IMPLANTACIÓN DE SCRUM BASADA EN RESPONSABILIDADES



IMPLANTACIÓN DE SCRUM BASADA EN RESPONSABILIDADES

Dirección
**LA ASIGNACIÓN DE
RESPONSABILIDADES
SE ADAPTA A LA
ORGANIZACIÓN**

Calidad

RR.HH.

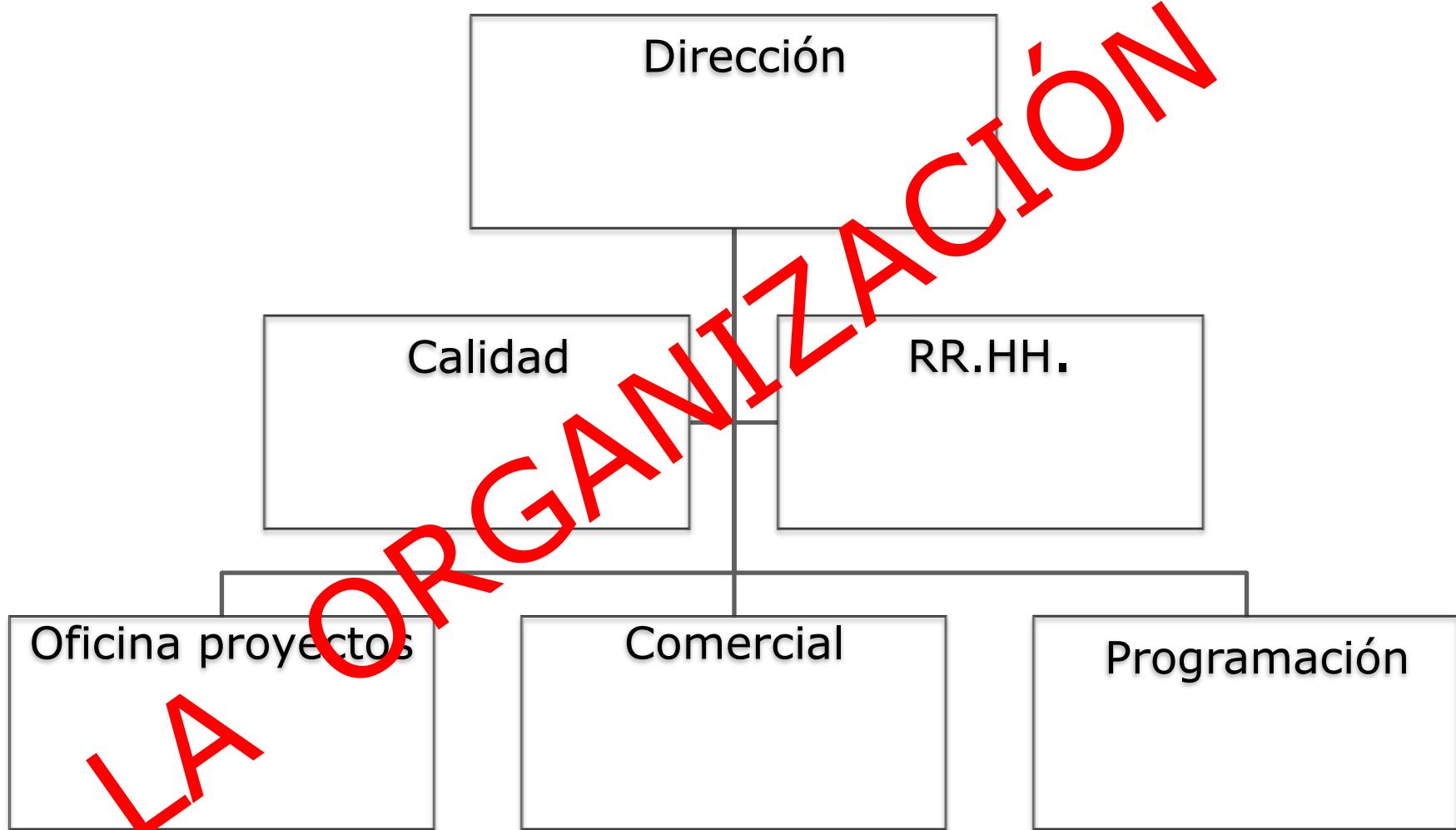
Oficina proyectos

Comercial

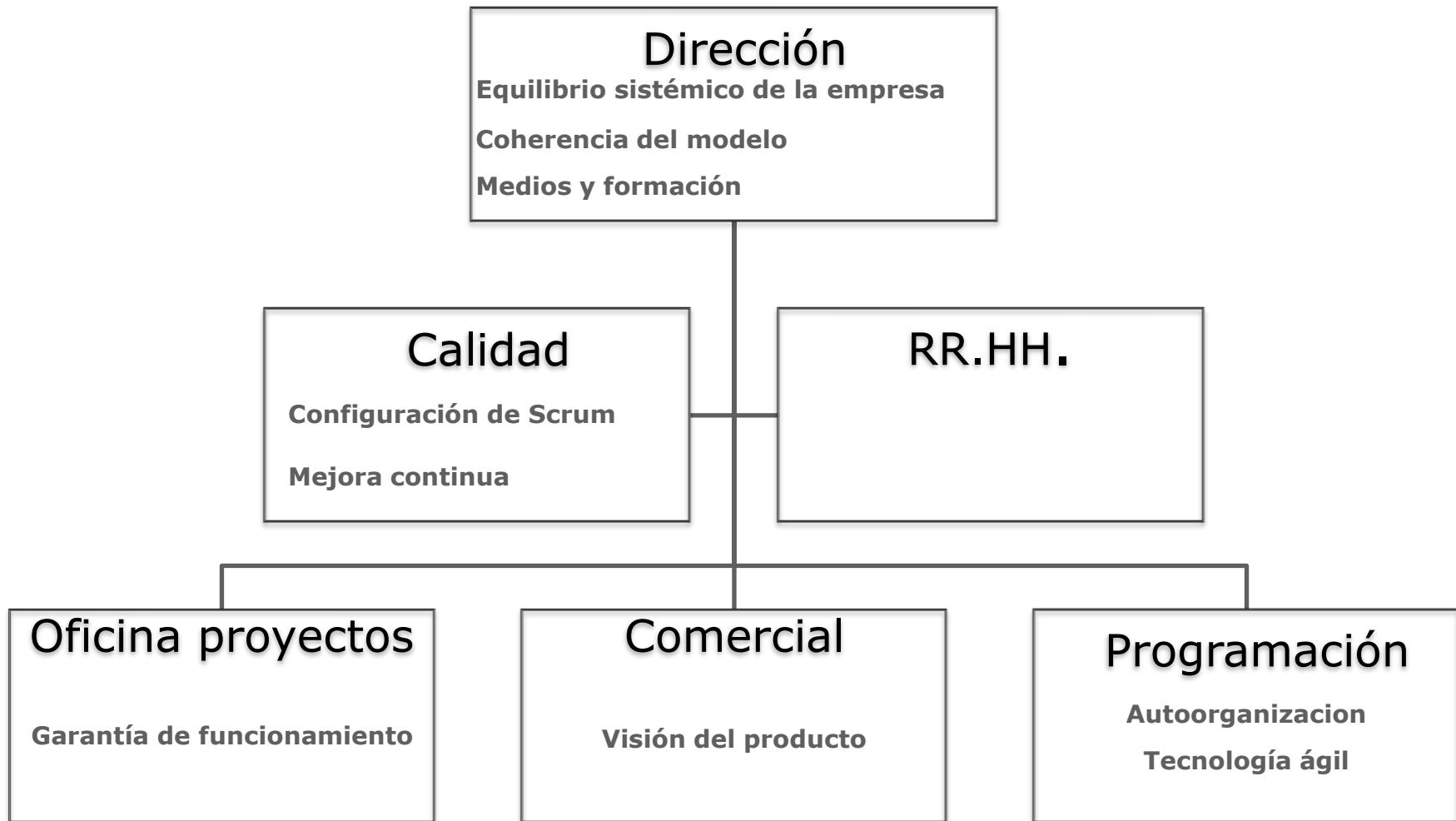
Programación

FLEXIBILIDAD / GLOBALIDAD

IMPLANTACIÓN DE SCRUM BASADA EN RESPONSABILIDADES



ROLES SCRUM BASADOS EN RESPONSABILIDADES: un ejemplo



ADOPTAR PRÁCTICAS MÁS O MENOS ÁGILES...

PREVISIBILIDAD

AGILIDAD

NO ES EL OBJETIVO

TRABAJAR DE FORMA MÁS O MENOS ÁGIL



TRABAJAR DE FORMA ADECUADA A LA ORGANIZACIÓN

PREVISIBILIDAD

AGILIDAD



ES EL OBJETIVO

VÍDEO

cc-by: Betsy Fletcher

Vídeo: Forgtek



INFO ABOUT RIGHTS

1401289956290

www.safecreative.org/work

© Juan Palacio

Para consultar los usos autorizados de este trabajo o contactar con el autor:
<http://www.safecreative.org/work/1401289956290>