



Filosofía Ágil

Manifiesto

Ágil

El principio (sin Ingeniería)

1era. fila (de izq a der): Bill Gates, Andrea Lewis, Marla Wood, y Paul Allen. En el medio: Bob O'Rear, Bob Greenberg, Marc McDonald, and Gordon Letwin. Atrás: Steve Wood, Bob Wallace, and Jim Lane.



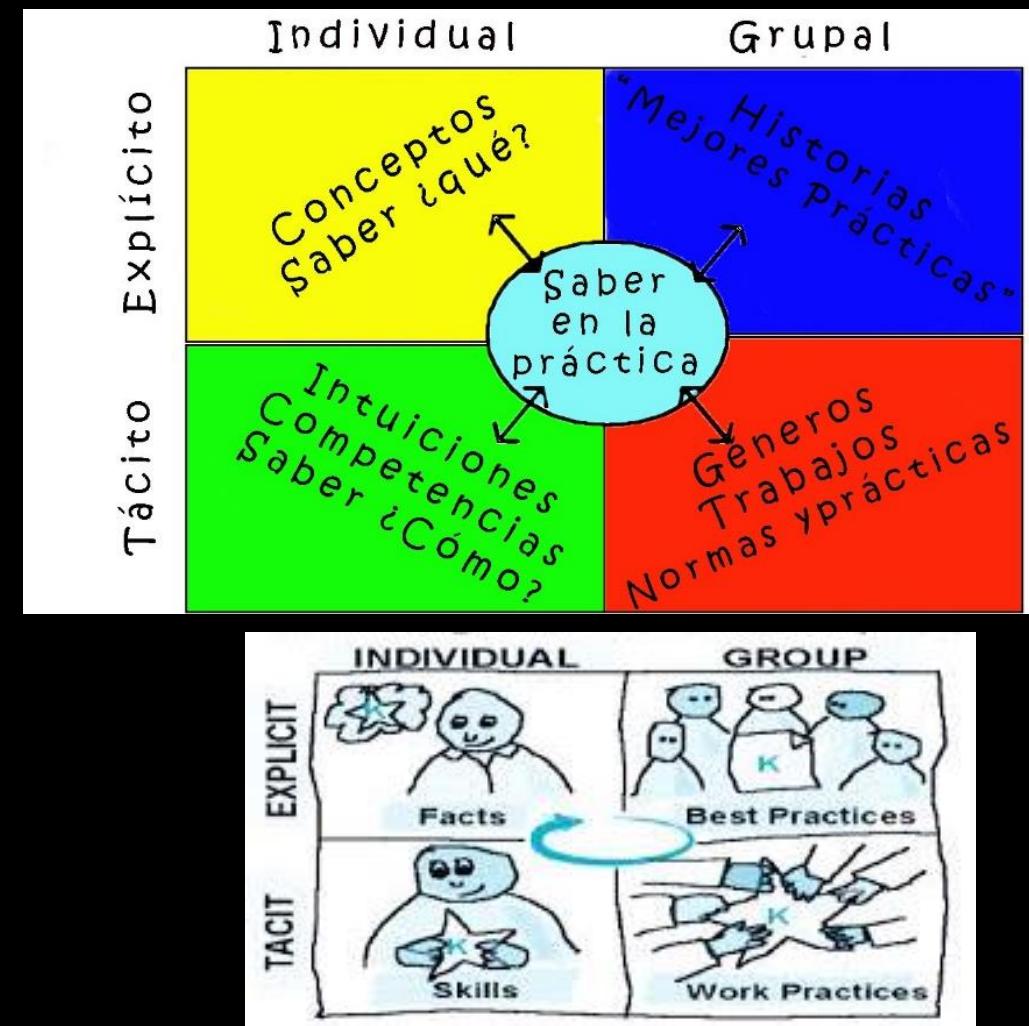
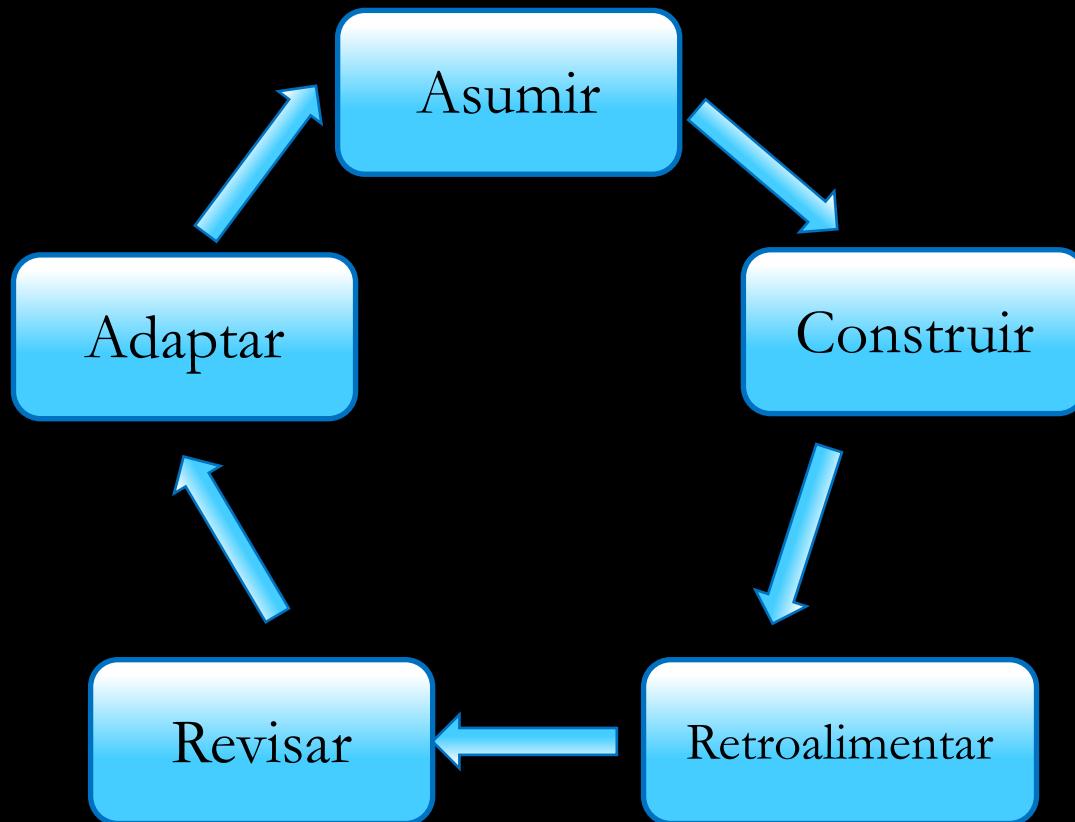
Desarrollo ágil de Software (Agile)

- Un compromiso útil entre nada de proceso y demasiado proceso (Fowler, 2001)

Procesos Empíricos



Patrón de conocimiento en procesos empíricos



VALORES ÁGILES

INDIVIDUOS E
INTERACCIONES



SOBRE



PROCESOS Y
HERRAMIENTAS

SOFTWARE
FUNCIONANDO



SOBRE



DOCUMENTACION
EXTENSIVA

COLABORAR CON
EL CLIENTE



SOBRE



NEGOCIACION
CONTRACTUAL

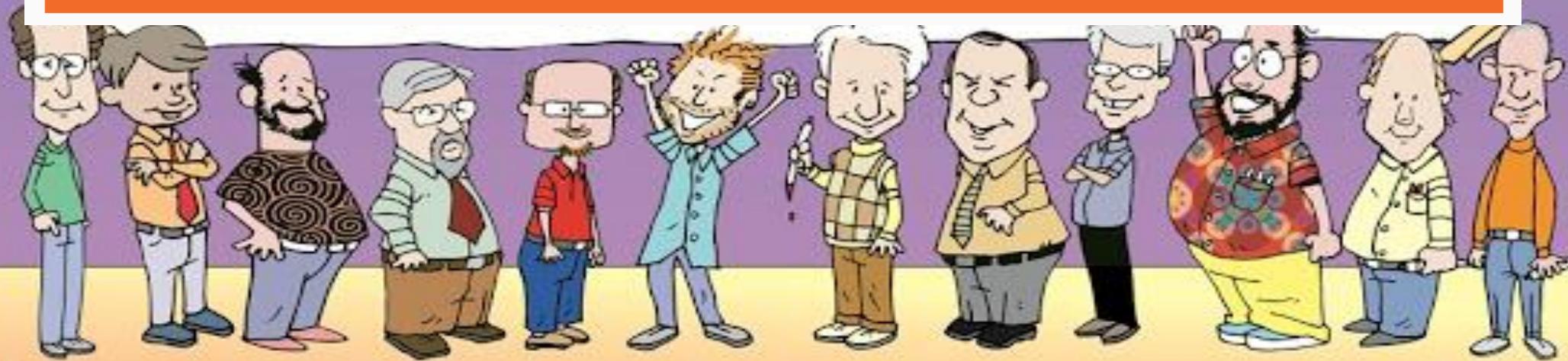
RESPONDER AL CAMBIO



SOBRE

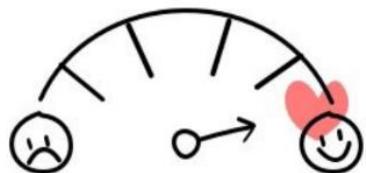


SEGUIR UN PLAN

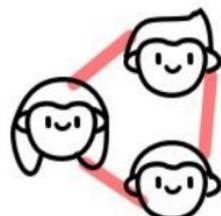


Los 12 principios del Manifiesto Ágil

1 Nuestra mayor prioridad es **satisfacer al cliente**.



4 Los responsables de negocios, diseñadores y desarrolladores deben **trabajar juntos** día a día durante el proyecto.



2 **Aceptar** que los requisitos cambien.



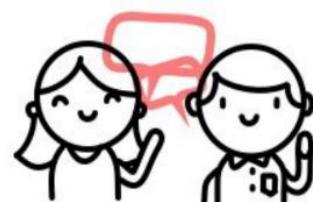
3 **Entregar** software funcional **frecuentemente**.



5 Desarrollamos proyectos en torno a **individuos motivados**.



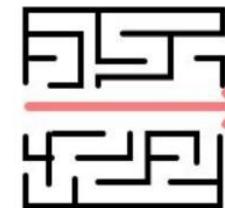
6 El método más eficiente de comunicar información es **conversaciones cara a cara**.



7 El **software funcionando** es la principal **medida de éxito**.



10 La **simplicidad** es esencial.



8 Los procesos ágiles promueven el **desarrollo sostenible**.



11 Las mejores arquitecturas, requisitos, y diseños emergen de **equipos auto-organizados**.



9 La **atención continua a la excelencia técnica y al buen diseño** mejor la Agilidad.



12 A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo y de acuerdo a esto **ajustan su comportamiento**.



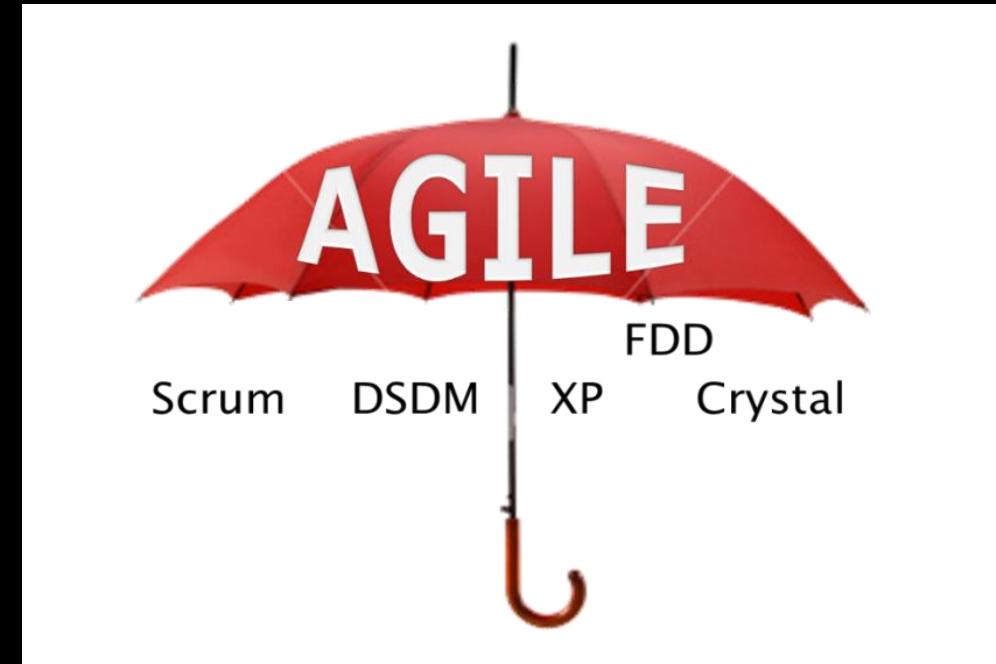
¿Qué es Ágil?

NO es una metodología o proceso

Ágil es una ideología con un conjunto definido de principios que guían el desarrollo del producto

Valores de los equipos ágiles ...

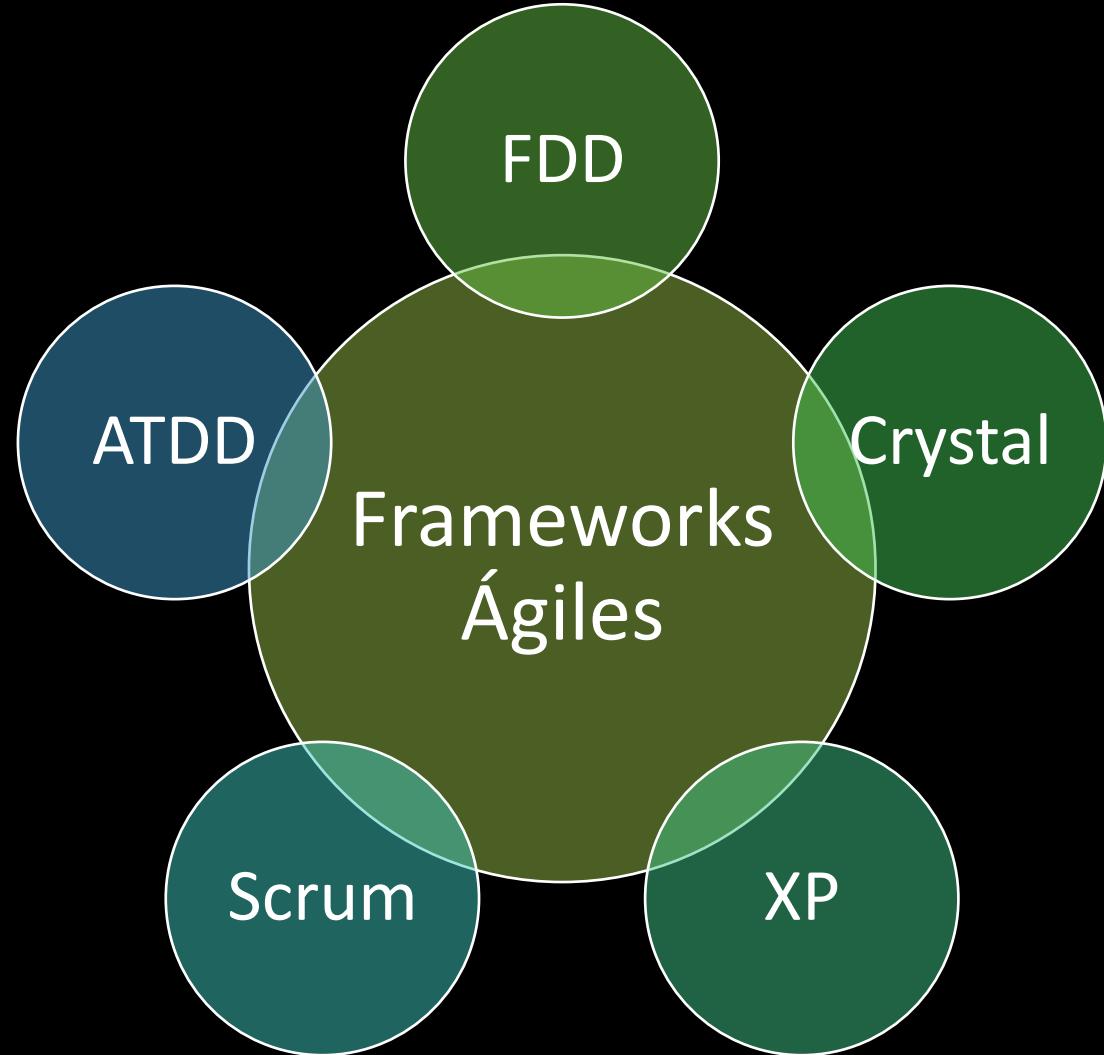
- Planificación continua, multi-nivel
- Facultados, auto-organizados, equipos completos
- Entregas frecuentes, iterativas y priorizadas
- Prácticas de ingeniería disciplinadas
- Integración continua
- Testing Concurrente



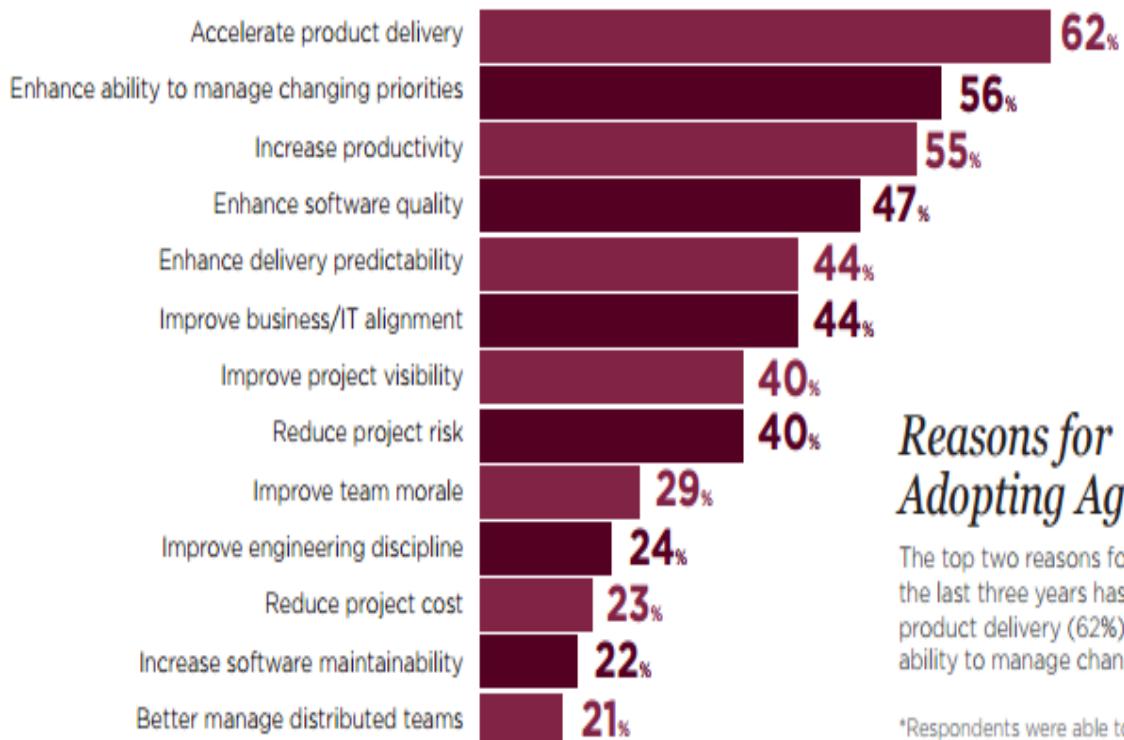
¿Pero qué significa Ágil?

- Balance entre ningún proceso y demasiado proceso. La diferencia inmediata es la exigencia de una menor cantidad de documentación, sin embargo no es eso lo más importante:
 - Los métodos ágiles son adaptables en lugar de predictivos.
 - Los métodos ágiles son orientados a la gente en lugar de orientados al proceso.

Algunos frameworks ágiles



¿Por qué ir a Agile?



Reasons for Adopting Agile

The top two reasons for adopting the last three years has been to aid product delivery (62%) and enhance ability to manage changing priorities (56%).

*Respondents were able to make multiple selections.

Top 3 Benefits of Agile



87%

Ability to manage changing priorities



85%

Increased team productivity



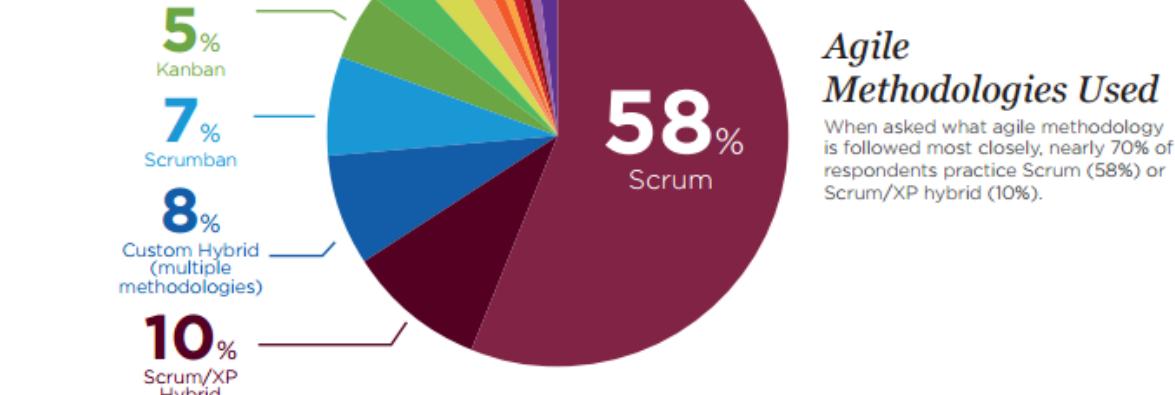
84%

Improved project visibility

AGILE METHODS AND PRACTICES

1% DSDM/Atern
1% Feature-Driven Development (FDD)
1% Agile Modeling
2% Lean Development
3% Other

3% Iterative Development



Agile Methodologies Used

When asked what agile methodology is followed most closely, nearly 70% of respondents practice Scrum (58%) or Scrum/XP hybrid (10%).

Técnicas efectivas

Agile Techniques Employed

More than 39% of the respondents practiced Kanban within their organizations, up from 31% in 2014. Conversely, iteration planning dropped slightly from 71% in 2014 to 69% in 2015, likely indicating a transition to more flow-based methods such as Lean and Kanban.

TOP 5 AGILE TECHNIQUES



83%

DAILY STANDUP



82%

PRIORITIZED BACKLOGS



79%

SHORT ITERATIONS



74%

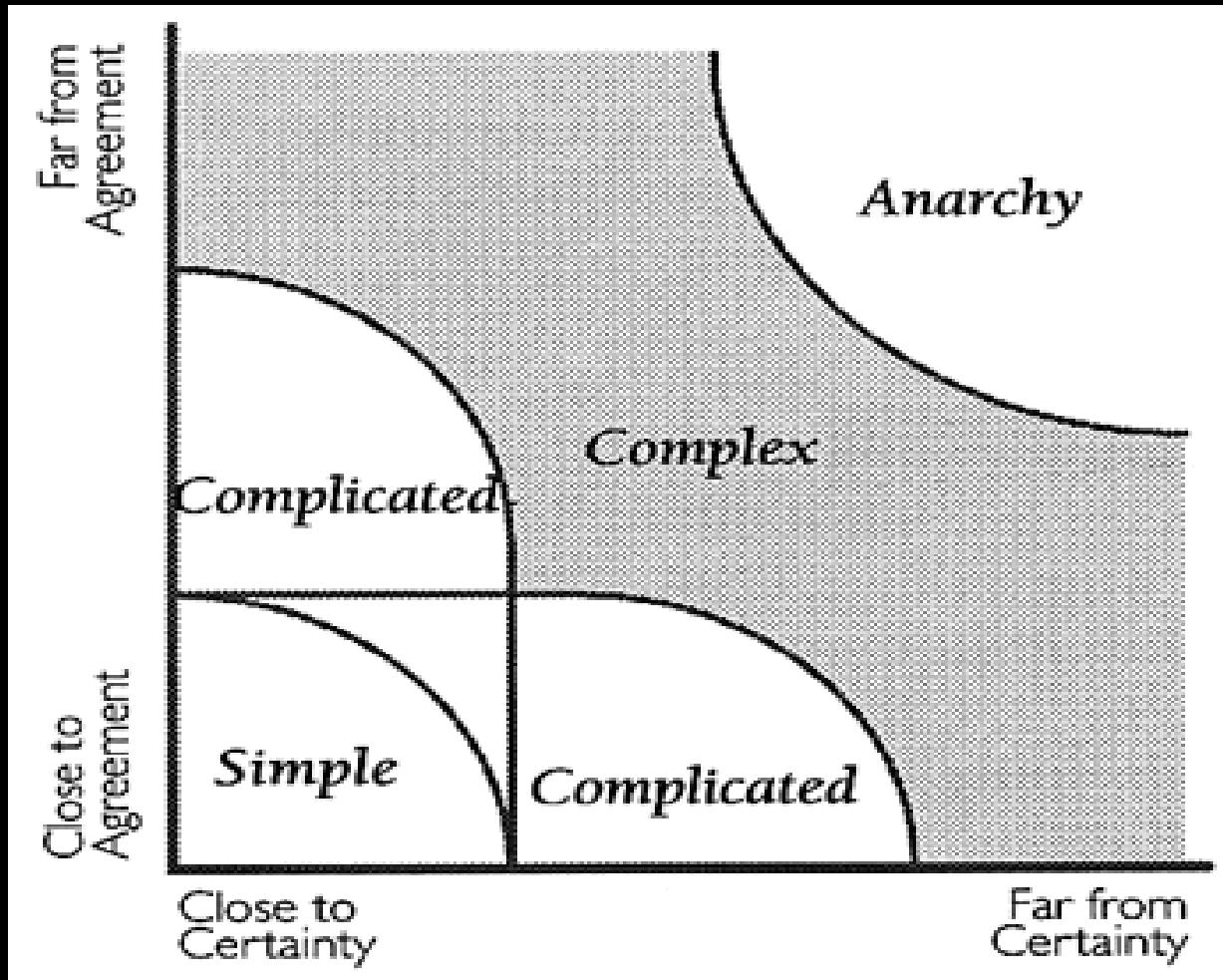
RETROSPECTIVES



69%

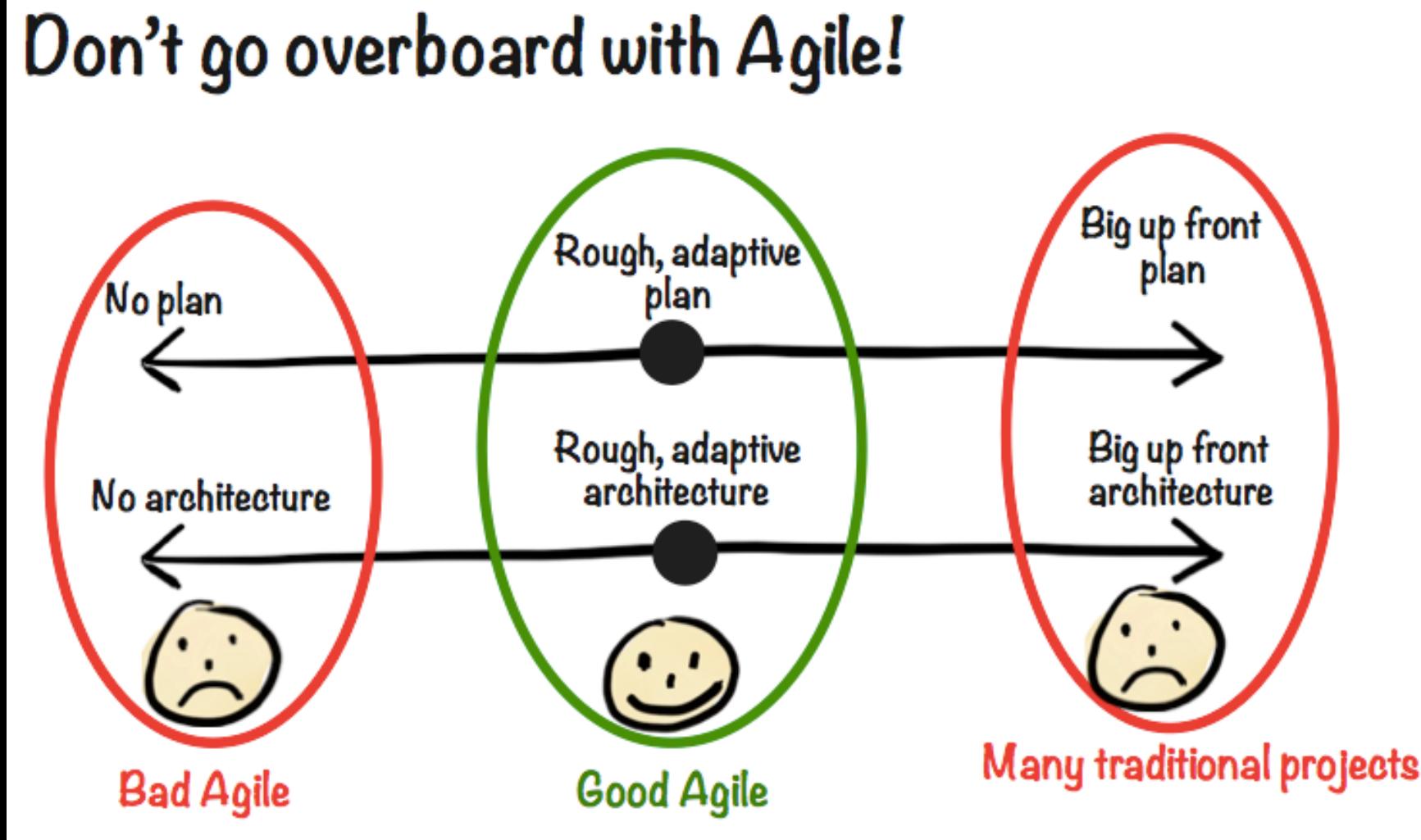
ITERATION PLANNING

¿Cuándo Agile es aplicable?



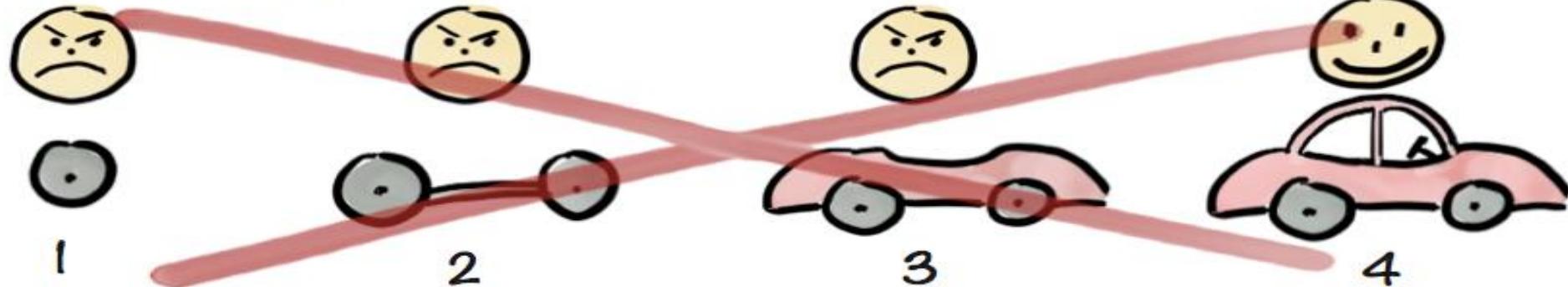
- Agile da mejores resultados cuando los problemas a ser resueltos caen dentro del espacio “Complex”.
- El desarrollo de nuevos productos y Knowledge Work tienden a estar en el espacio Complex.
- Investigación está dentro de Anarchy
- Mantenimiento cae en Simple (siempre????)

Ser Ágil no es ser indisciplinado.

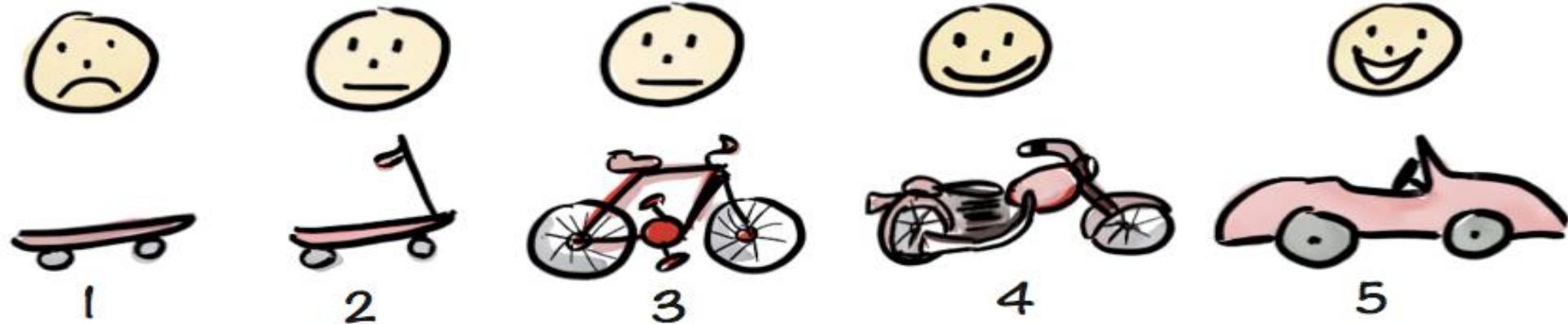


Entonces hacemos todo por pedacitos y
somos agiles!!!?? NO!

Not like this....

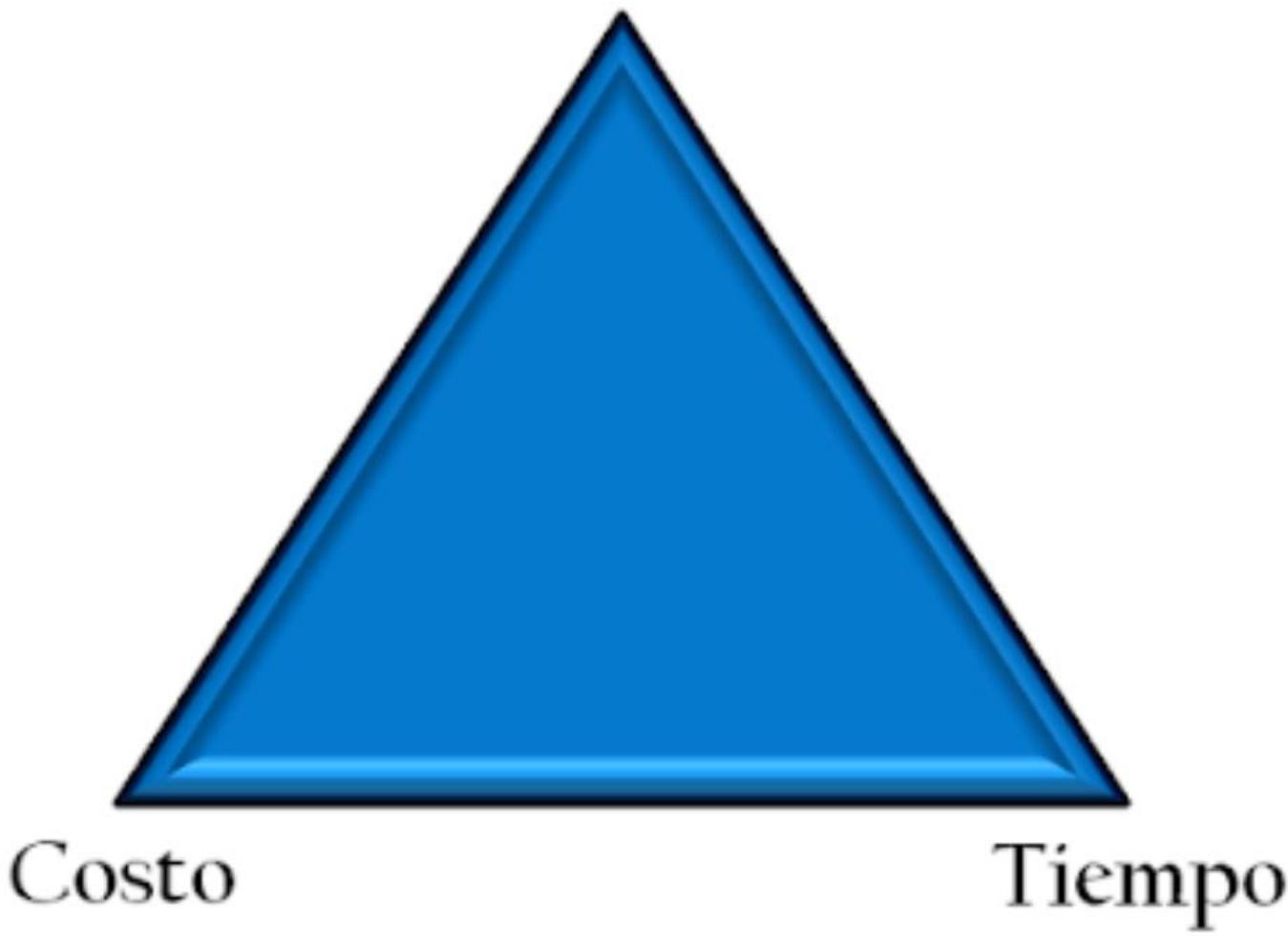


Like this!



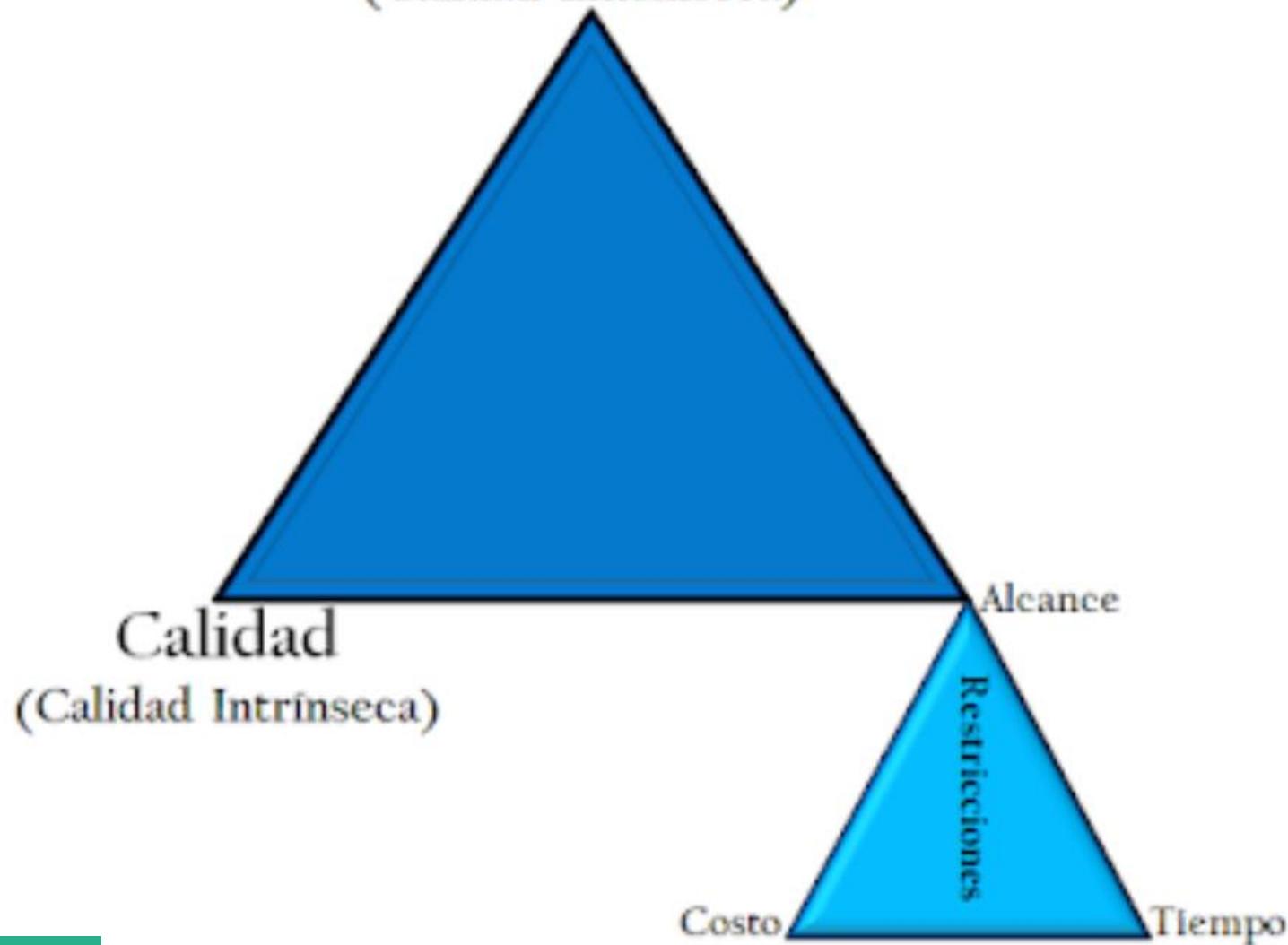
El Triángulo de Hierro

Alcance



El Triángulo Ágil

Valor
(Calidad Extrínseca)



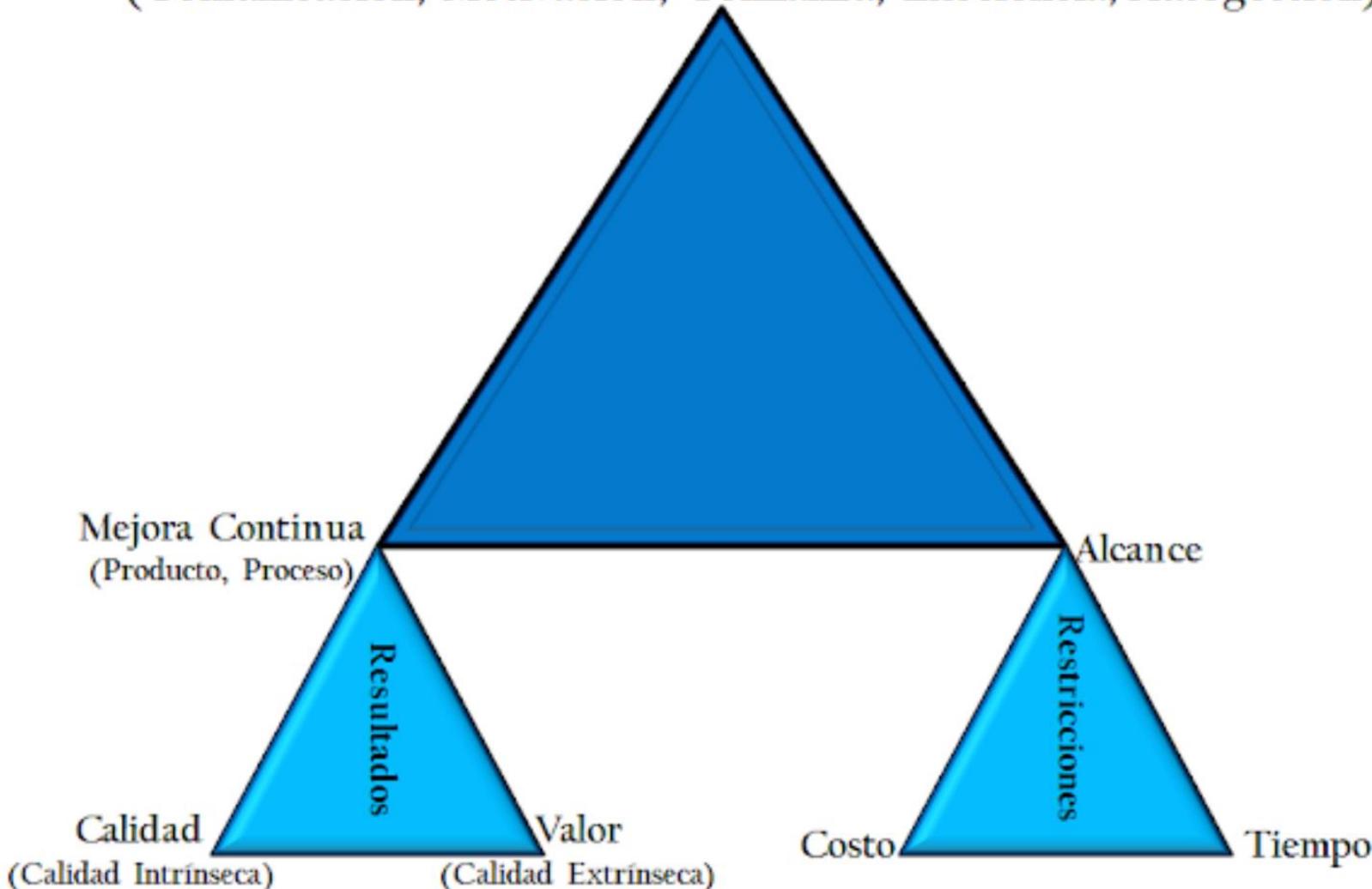
El triángulo Ágil

Highsmith

El Triángulo Ágil (Modificado)

Personas

(Comunicación, Motivación, Confianza, Excelencia, Autogestión)



Requerimientos Ágiles



PROBLEMA / NECESIDAD/ OPORTUNIDAD

[PONGA AQUÍ EL
CAMINO MÁS EFECTIVO]



SOLUCIÓN / ENTREGABLE

VALORES ÁGILES

INDIVIDUOS E
INTERACCIONES



SOBRE



PROCESOS Y
HERRAMIENTAS

SOFTWARE
FUNCIONANDO



SOBRE



DOCUMENTACION
EXTENSIVA

COLABORAR CON
EL CLIENTE



SOBRE



NEGOCIACION
CONTRACTUAL

RESPONDER
AL CAMBIO



SOBRE



SEGUIR UN PLAN

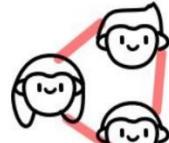


Los 12 principios del Manifiesto Ágil

1 Nuestra mayor prioridad es **satisfacer al cliente**.



4 Los responsables de negocios, diseñadores y desarrolladores deben **trabajar juntos** día a día durante el proyecto.



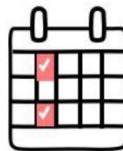
2 Aceptar que los requisitos cambien.



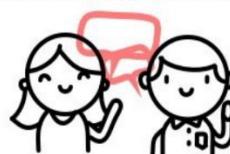
5 Desarrollamos proyectos en torno a **individuos motivados**.



3 Entregar software funcional **frecuentemente**.



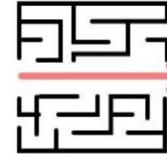
6 El método más eficiente de comunicar información es **conversaciones cara a cara**.



7 El **software funcionando** es la principal **medida de éxito**.



10 La **simplicidad** es esencial.



8 Los procesos ágiles promueven el **desarrollo sostenible**.



11 Las mejores arquitecturas, requisitos, y diseños emergen de **equipos auto-organizados**.



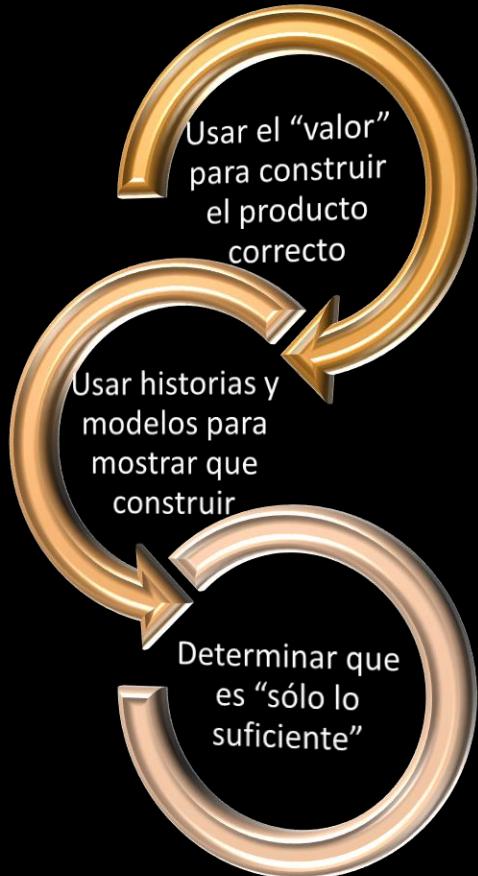
9 La **atención continua a la excelencia técnica y al buen diseño** mejor la Agilidad.



12 A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo y de acuerdo a esto **ajustan su comportamiento**.



Requerimientos en Agile



El costo del tradicional BRUF

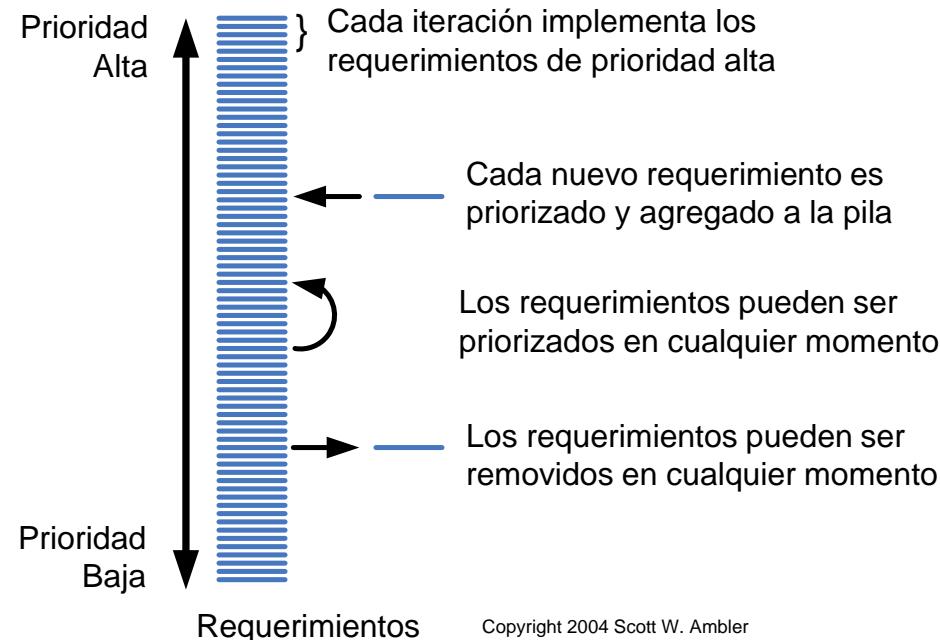
Los productos
“Exitosos”
también tienen
un desperdicio
significante



Fuente: Jim Johnson of the Standish Group, Keynote Speech XP 2002

Gestión Ágil de Requerimientos de Software

Los requisitos cambiantes son una ventaja competitiva
si puede actuar sobre ellos





just in time

Analice cuando lo
necesite, no antes

El cara-a-cara permite que fluya información vocal, subvocal, gestual con realimentación rápida.



*“Valor es la obtención de beneficio **tangible** o **intangible**”*

Masa Maeda, Serious LeAP

“El valor lo asociamos a la utilidad, beneficio o satisfacción que le ofreces a los usuarios finales por cada funcionalidad completa que le entregas”

Pablo Lischinsky, Agile Trainer & Consultant, Entrepreneur



FIJO →

REQUISITOS



ESTIMADO →

RECURSOS

TIEMPO

RECURSOS

TIEMPO

Dirigido
por
valor

ALCANCE

Tipos de Requerimientos



Requerimiento de Negocio

Disminuir un X% de tiempo invertido en procesos manuales relacionados con atención al cliente.

Requerimiento de Usuario

Realizar consultas en línea del estado de cuenta de los clientes

Requerimiento Funcional

Generar reporte de saldos de cuenta. Recibir notificaciones por mail.

Requerimiento No funcional

Formato del reporte PDF. Cumplir niveles de seguridad para credenciales de usuarios según la ley bancaria 9999XX

Requerimiento de Implementación

Servidores en la nube

**Requerimientos
de Negocio**



**Dominio
del
Problema**

**Requerimientos de
Usuario**

Requerimientos de Software

**Dominio
de la
Solución**



EN RESUMEN...



ENTENDIENDO LA
NECESIDAD Y NEGOCIO...



DESCUBRIENDO LA SOLUCIÓN
DE FORMA COLABORATIVA...

B
E
D
C



JUNTO A UN EQUIPO
MOTIVADO Y COMPETENTE...

A
N



ENTREGAMOS
FRECUENTEMENTE VALOR A
LOS STAKEHOLDERS.

Por último



Los cambios son la única constante.



Stakeholders: no son todos los que están.



Siempre se cumple eso de que: “El usuario dice lo que quiere cuando recibe lo que pidió”.



No hay técnicas ni herramientas que sirvan para todos los casos.

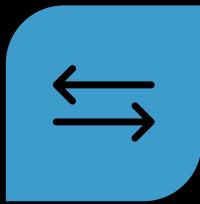


Lo importante no es entregar una salida, un requerimiento, lo importante es entregar, un resultado, una solución de “valor”.

Principios Ágiles relacionados a los Requerimientos Ágiles



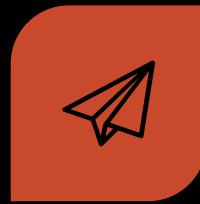
1- LA PRIORIDAD ES SATISFACER AL CLIENTE A TRAVÉS DE RELEASES TEMPRANOS Y FRECUENTES (2 SEMANAS A UN MES)



2 -RECIBIR CAMBIOS DE REQUERIMIENTOS, AUN EN ETAPAS FINALES



4 - TÉCNICOS Y NO TÉCNICOS TRABAJANDO JUNTOS TODO EL PROYECTO



6 - EL MEDIO DE COMUNICACIÓN POR EXCELENCIA ES CARA A CARA



11 - LAS MEJORES ARQUITECTURAS, DISEÑOS Y REQUERIMIENTOS EMERGEN DE EQUIPOS AUTOORGANIZADOS

User Stories

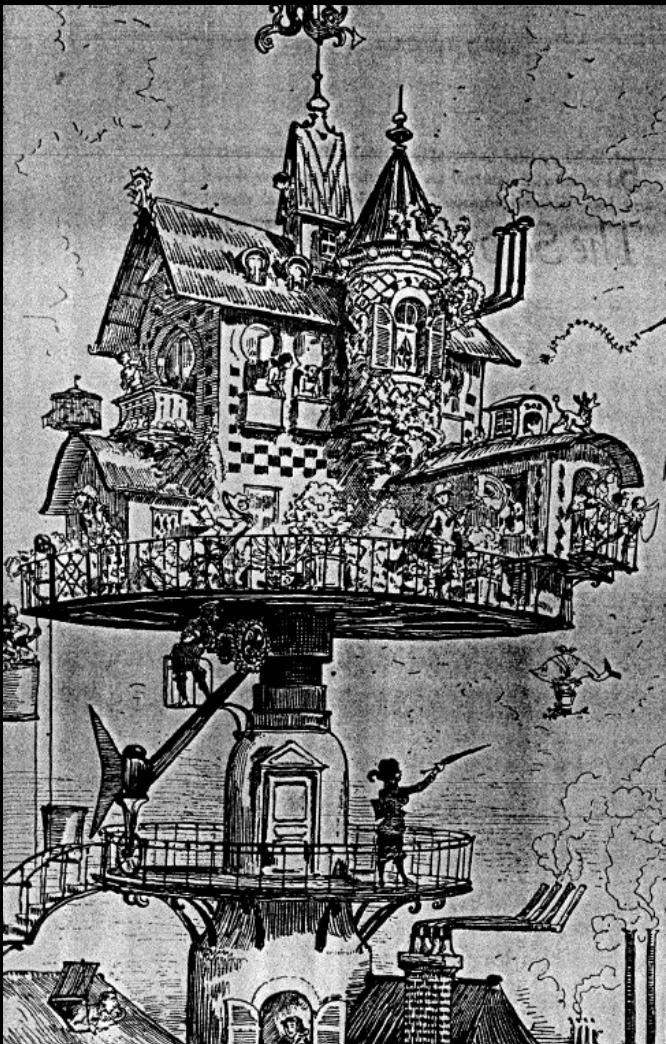
“....se las llama “stories” porque se supone que Ud. cuenta una historia. Lo que se escribe en la tarjeta no es importante, lo que Ud. habla, si!.

--- Jeff Patton, InfoQ,

create conversation.



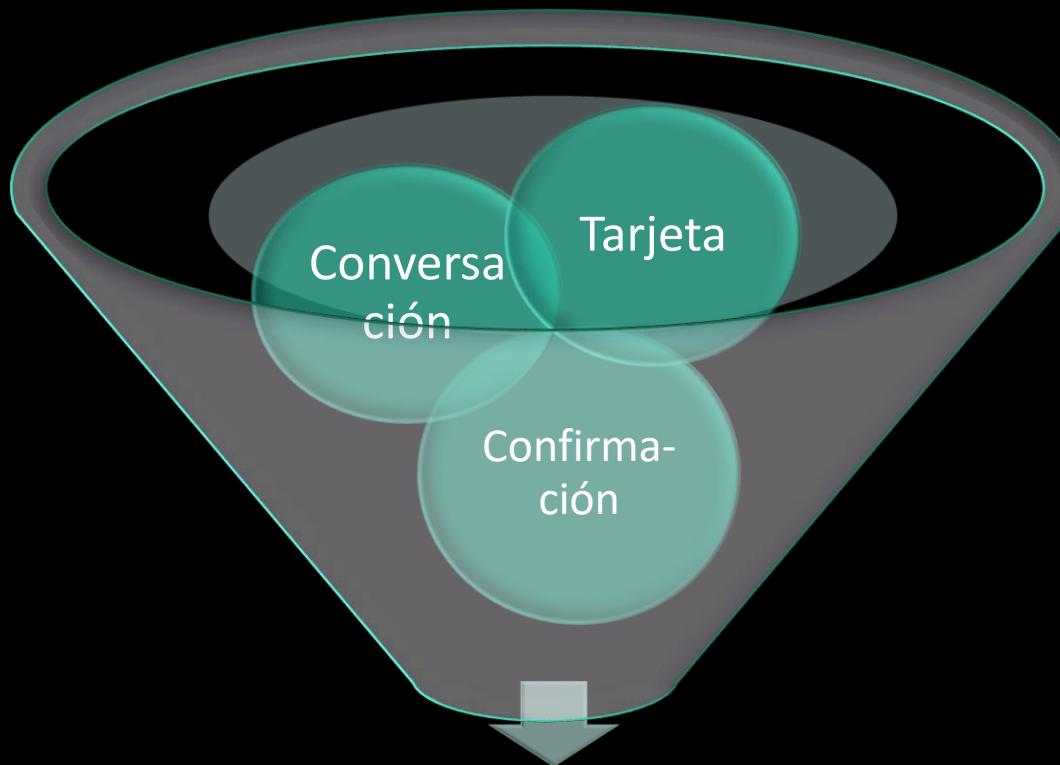
@gapingvoid



La parte más difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados... Ninguna otra parte del trabajo afecta tanto el sistema resultante si se hace incorrectamente. Ninguna otra parte es tan difícil de rectificar más adelante”

Fred Brooks - “No Silver Bullet - Essence and Accidents of Software Engineering”. IEEE Computer, Abril de 1987.

¿Cuáles son las partes de una User Story?



User Story

Forma de expresar las Historias de Usuario

Como <nombre del rol>,
yo puedo <actividad>
de forma tal que <valor
de negocio que
recibo>

Comunica porque es
necesaria la actividad



Representa quién está
realizando la acción o
quién recibe el valor
de la actividad.

Representa la
acción que realizará
el sistema

User Story: un ejemplo de tarjeta

Buscar Destino por Dirección

Frase verbal

Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

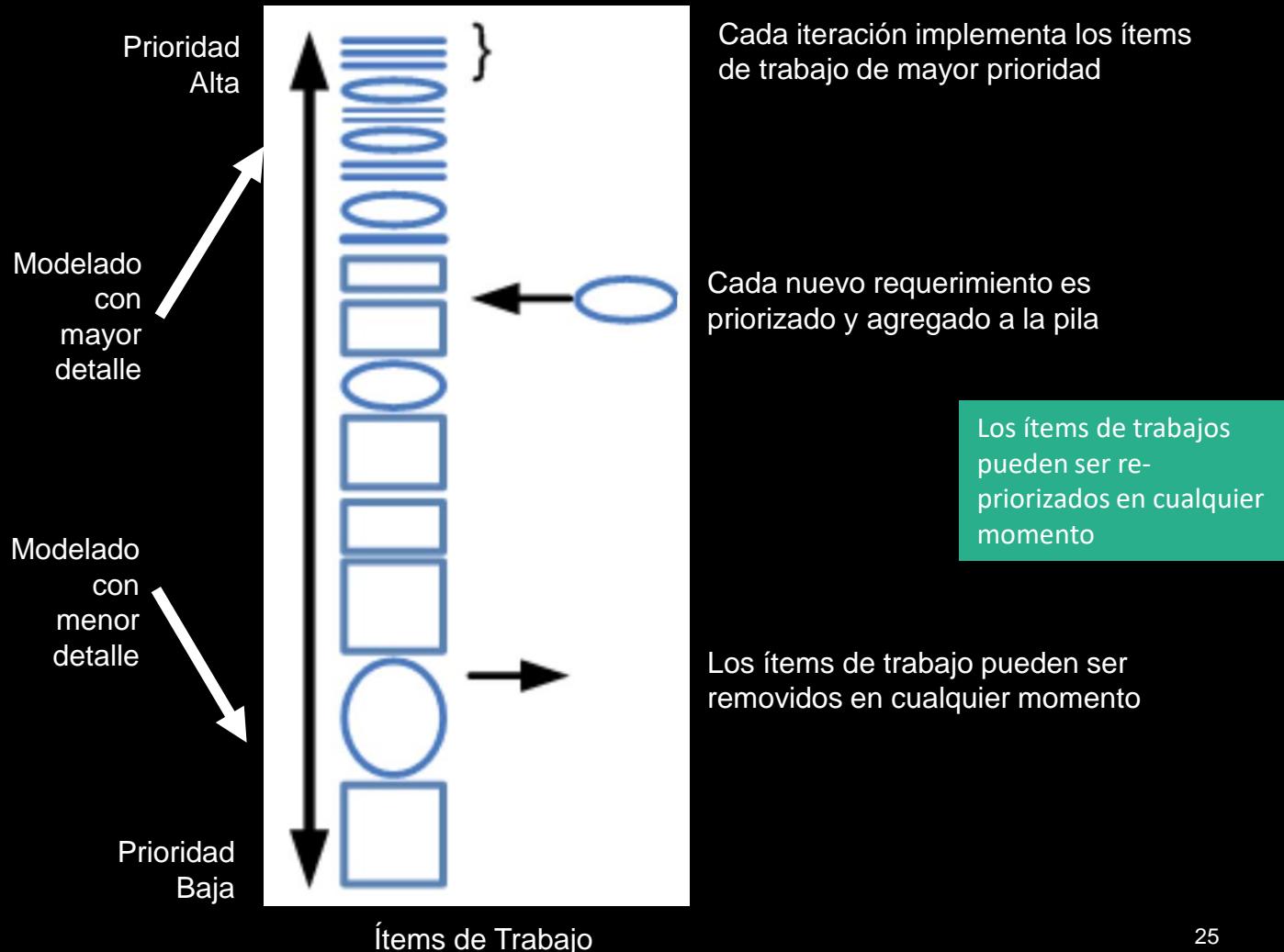
Las User Stories son Multipropósito

- Las historias son:
 - Una necesidad del usuario
 - Una descripción del producto
 - Un ítem de planificación
 - Token para una conversación
 - Mecanismo para diferir una conversación

* Kent Beck coined the term user stories in *Extreme Programming Explained 1st Edition*, 1999



El Product Owner Prioriza las historias en el Product Backlog



User stories: Porciones Verticales



Donde impacta
“fuertemente”
esto??

Story 1	Story 2	
GUI		
Business Logic		
Database		

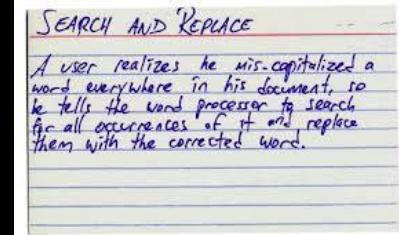
Modelado de Roles



Modelado de Roles: Tarjeta de Rol de Usuario

Rol de Usuario: Reclutador Interno

Nos es un experto en computadoras, pero bastante adepto a utilizar la Web. Utilizará el software con poca frecuencia pero muy intensamente. Leerá anuncios de otras compañías para averiguar cuál es la mejor palabra para sus anuncios. La facilidad de uso es importante, pero más importante es que lo que aprenda, lo pueda recordar meses después.



Usuarios Representantes (Proxies)

- Tipos de usuarios representantes:
 - Gerentes de Usuarios
 - Gerentes de Desarrollo
 - Alguien del grupo de marketing
 - Vendedores
 - Expertos del Dominio
 - Clientes
 - Capacitadores y personal de soporte.

*No son ideales como
los usuarios verdaderos
...EVITELOS !!!*

Criterios de Aceptación de User Stories



Criterios de Aceptación de Historias de Usuario

- Definen límites para una user story (US)
- Ayudan a que los PO respondan lo que necesitan para que la US provea valor (requerimientos funcionales mínimos)
- Ayudan a que el equipo tenga una visión compartida de la US
- Ayudan a desarrolladores y testers a derivar las pruebas.
- Ayudan a los desarrolladores a saber cuando parar de agregar funcionalidad en una US



User Story: un ejemplo de tarjeta

Buscar Destino por Dirección

Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

Criterios de Aceptación:

- La altura de la calle es un número.
- La búsqueda no puede demorar más de 30 segundos.

¿Cuáles son los Criterios de Aceptación buenos?

- Definen una intención, no una solución
 - Ej.: El usuario debe elegir al menos una cuenta para operar
- Son independientes de la implementación
- Relativamente de alto nivel, no es necesario que se escriba cada detalle



¿Y los detalles? ¿Dónde van?



- Detalles como:
 - El encabezado de la columna se nombra “Saldo”
 - El formato del saldo es 999.999.999,99
 - Debería usarse una lista desplegable en lugar de un Check box.
- Estos detalles que son el resultado de las conversaciones con el PO y el equipo puede capturarlos en dos lugares:
 - Documentación interna de los equipos
 - Pruebas de aceptación automatizadas

Pruebas de Aceptación de User Stories

Front of Card

IB

As a student I want to purchase
a parking pass so that I can
drive to school

Priority: ~~High~~ Should
Estimate: 4

Back of Card

Confirmations:

The student must pay the correct amount.
One pass for one month is issued at a time.
The student will not receive a pass if the payment
isn't sufficient.

The person buying the pass must be a currently
enrolled student.

The student may only buy one pass per month.

Pruebas de Aceptación de Historias de Usuario

Expresan detalles resultantes de la conversación

Complementan la User Story

Proceso de dos pasos:

1. Identificarlas al dorso de la US.
2. Diseñar las pruebas completas



User Story: Tarjeta y Pruebas de Aceptación

Buscar Destino por Dirección

Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

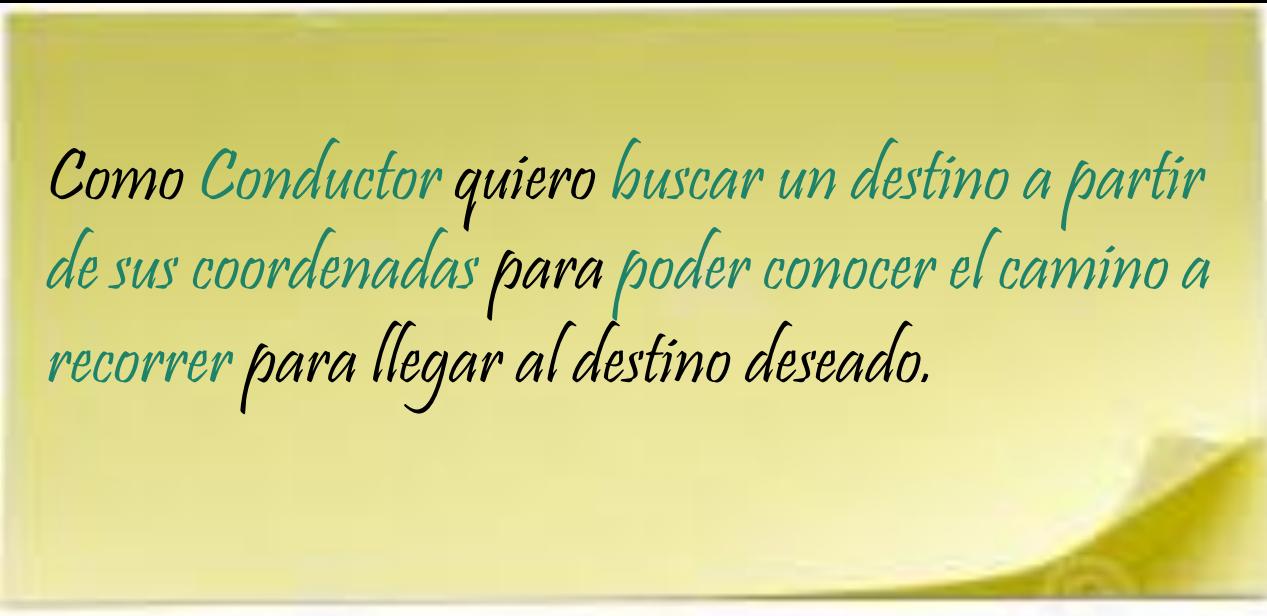
Criterios de Aceptación:

- La altura de la calle es un número.
- La búsqueda no puede demorar más de 30 segundos.

Pruebas de Usuario

- Probar buscar un destino en un país y ciudad existentes, de una calle existente y la altura existente (pasa).
- Probar buscar un destino en un país y ciudad existentes, de una calle inexistente (falla).
- Probar buscar un destino en un país y ciudad existentes, de una calle existente y la altura inexistente (falla).
- Probar buscar un destino en un país inexistente (falla).
- Probar buscar un destino en País existente, ciudad inexistente (falla).
- Probar buscar un destino en un país y ciudad existentes, de una calle existente y demora más de 30 segundos (falla).

Ejemplo: para un software de un GPS



Como Conductor quiero buscar un destino a partir de sus coordenadas para poder conocer el camino a recorrer para llegar al destino deseado.

Ejemplo:

Como *Conductor* quiero buscar un destino a partir de sus coordenadas para poder conocer el camino a recorrer para llegar al destino deseado.

Criterios de Aceptación: Las coordenadas se representan con tres números que indican longitud y tres números que indican latitud. Cada número representa los grados, minutos y segundos respectivamente. Además se debe indicar la orientación (norte, sur, este, oeste).

- Probar buscar un destino en un país y ciudad existentes, de dos coordenadas existentes (pasa).
- Probar buscar un destino en un país y ciudad existentes, de una coordenada inexistente (falla).
- Probar buscar un destino en un país y ciudad existentes, de dos coordenadas existentes sin indicar la orientación (falla).
- Probar ingresar coordenadas de latitud y longitud válidas (pasa).
- Probar ingresar coordenadas de latitud y longitud inválidas (falla).

Ejemplo: User Stories / Casos de Prueba

Como compañía quiero pagar por una búsqueda de puestos con una tarjeta de crédito, así resuelvo mi necesidad en forma más eficiente.

Criterio de Aceptación:

- Se acepta Visa, MasterCard y American Express
- En compras mayores de \$100 se piden el número del dorso de la tarjeta

Probar con Visa (pasa)

Probar con MasterCard (pasa)

Probar con American Express (pasa)

Probar con Dinner's Club (falla)

Probar con números de tarjeta buenos

Probar con números de tarjeta malos

Probar con números de tarjeta faltantes

Probar con tarjetas vencidas

Probar con montos menores de \$100

Probar con montos mayores de \$100

Definición de listo – Definition of Ready

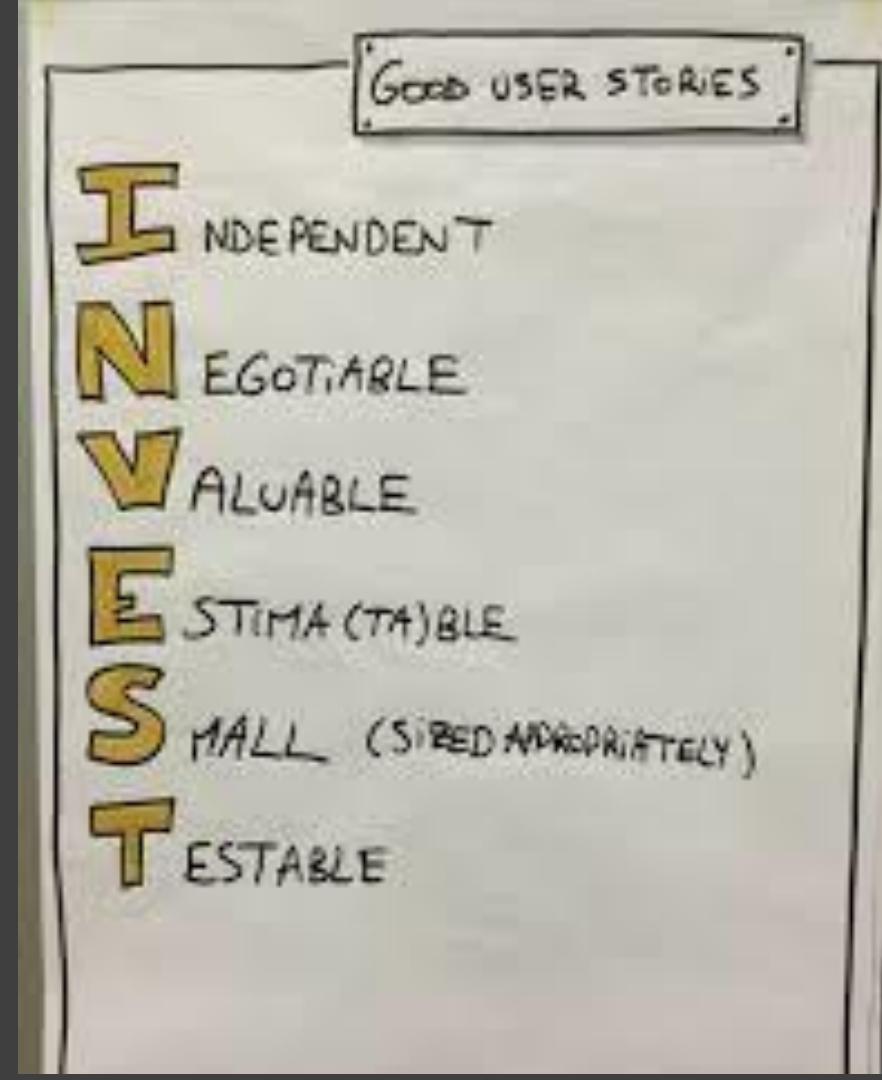


Definición de Hecho – Definition of Done



*INVEST Model

- **Independent** – calendarizables e implementables en cualquier orden
- **Negotiable** – el “qué” no el “cómo”
- **Valuable** – debe tener valor para el cliente
- **Estimatable** – para ayudar al cliente a armar un ranking basado en costos
- **Small** – deben ser “consumidas” en una iteración
- **Testable** – demostrar que fueron implementadas

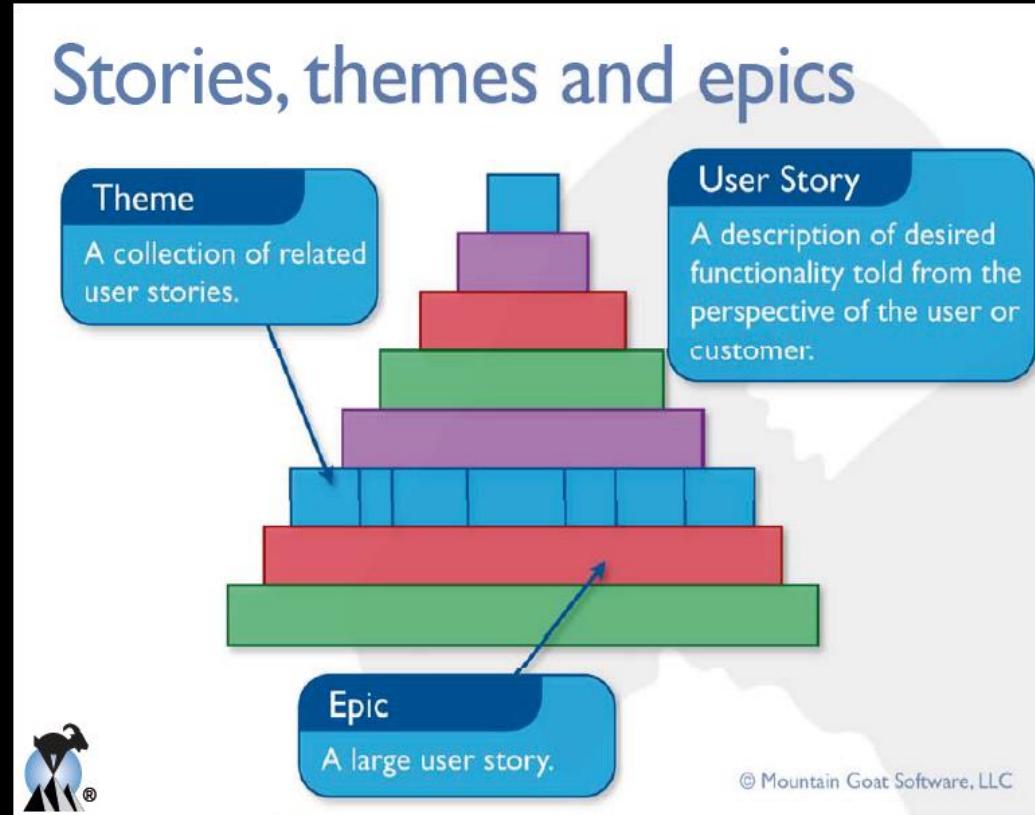


* “INVEST in Stories” – Bill Wake

Algo más sobre las User Stories...

- No son especificaciones detalladas de requerimientos (como los casos de uso)
- Son expresiones de intención, “es necesario que haga algo como esto...”
- No están detallados al principio del proyecto, elaborados evitando especificaciones anticipadas, demoras en el desarrollo, inventario de requerimientos y una definición limitada de la solución.
- Necesita poco o nulo mantenimiento y puede descartarse después de la implementación.
- Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

Diferentes niveles de abstracción



Idea

Problema

Necesidad

Cambios en el Negocio

Ideas / Cambios en el Software

Detalle de Implementación

Impacto en el Cliente

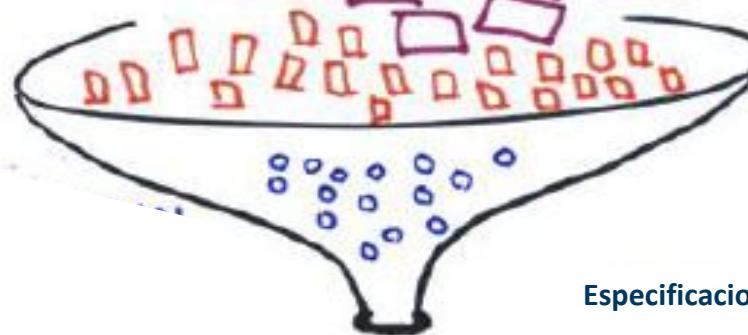
"EPICS"

"STORIES"

Especificaciones

Objetivo

Objetivo



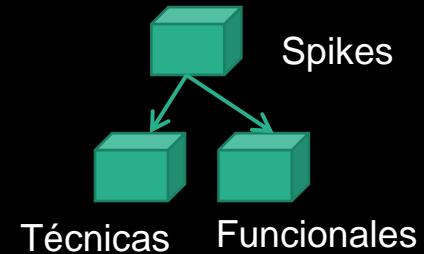
Qué no son las user stories....



Spikes

- Tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una User Story u otra faceta del proyecto.
- Se clasifican en : técnicas y funcionales.
- Pueden utilizarse para:
 - Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.
 - Analizar un comportamiento de una historia compleja y poder así dividirla en piezas manejables.
 - Ganar confianza frente a riesgos tecnológicos, investigando o prototipando para ganar confianza.
 - Frente a riesgos funcionales, donde no está claro como el sistema debe resolverla interacción con el usuario para alcanzar el beneficio esperado.

Spikes

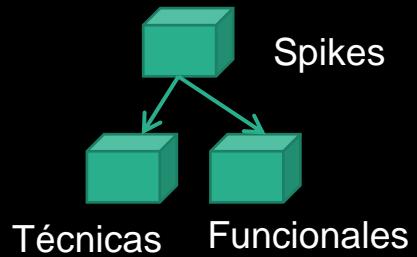


Técnicas

- Utilizadas para investigar enfoques técnicos en el dominio de la solución.
 - Evaluar performance potencial
 - Decisión hacer o comprar
 - Evaluar la implementación de cierta tecnología.
- Cualquier situación en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad en un tiempo fijo.

Funcionales

- Utilizadas cuando hay cierta incertidumbre respecto de cómo el usuario interactuará con el sistema.
- Usualmente son mejor evaluadas con prototipos para obtener realimentación de los usuarios o involucrados.



Spikes

- Algunas User Stories requieren de ambos tipos de spikes. Por ejemplo:
 - Como un cliente, quiero ver mi uso diario de energía en un histograma, para poder comprender rápidamente mi consumo de energía pasado, presente y proyectado.
- En este caso un equipo puede crear dos spikes:
 - Spike Técnico:
 - Investigar cuánto tiempo requiere actualizar un display de un cliente al uso actual, determinando requerimientos de comunicación, ancho de banda y si los datos se actualizan en formato push o pull.
 - Spike Funcional:
 - Crear un prototipo de histograma en el portal web y obtener la retroalimentación de algunos usuarios respecto del tamaño, el estilo de la presentación y los atributos gráficos.

Lineamientos para Spikes

Estimables, demostrables, y aceptables

La excepción, no la regla

- Toda historia tiene incertidumbre y riesgos.
- El objetivo del equipo es aprender a aceptar y resolver cierta incertidumbre en cada iteración.
- Los spikes deben dejarse para incógnitas mas críticas y grandes.
- Utilizar spikes como última opción.

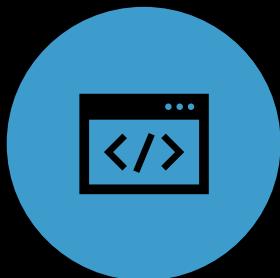
Implementar la spike en una iteración separada de las historias resultantes

- Salvo que el spike sea pequeño y sencillo y sea probable encontrar una solución rápida en cuyo caso, spike e historia pueden incluirse en la misma iteración.

Algunas cosas para dejar en claro



DIFERIR EL ANÁLISIS
DETALLADO TAN TARDE
COMO SEA POSIBLE, LO QUE
ES JUSTO ANTES DE QUE EL
TRABAJO COMIENCE.



HASTA ENTONCES, SE
CAPTURAN
REQUERIMIENTOS EN LA
FORMA DE “USER STORIES”.



LAS USER STORIES NO SON
REQUERIMIENTOS DE
SOFTWARE, NO NECESITAN
SER DESCRIPCIONES
EXHAUSTIVAS DE LA
FUNCIONALIDAD DEL
SISTEMA.

Tips para que las user stories sean útiles para el equipo



Un paso a la vez (evitar la palabra “Y”)



Usar palabras claras en los criterios de aceptación



No olvides la parte invisible: la conversación



Las user stories se escriben desde la perspectiva del usuario

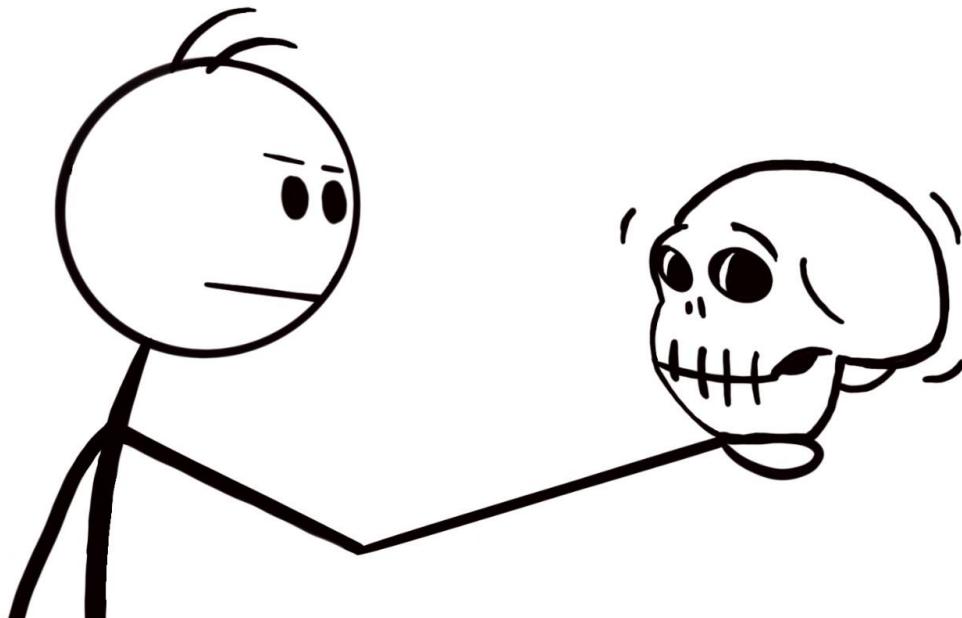


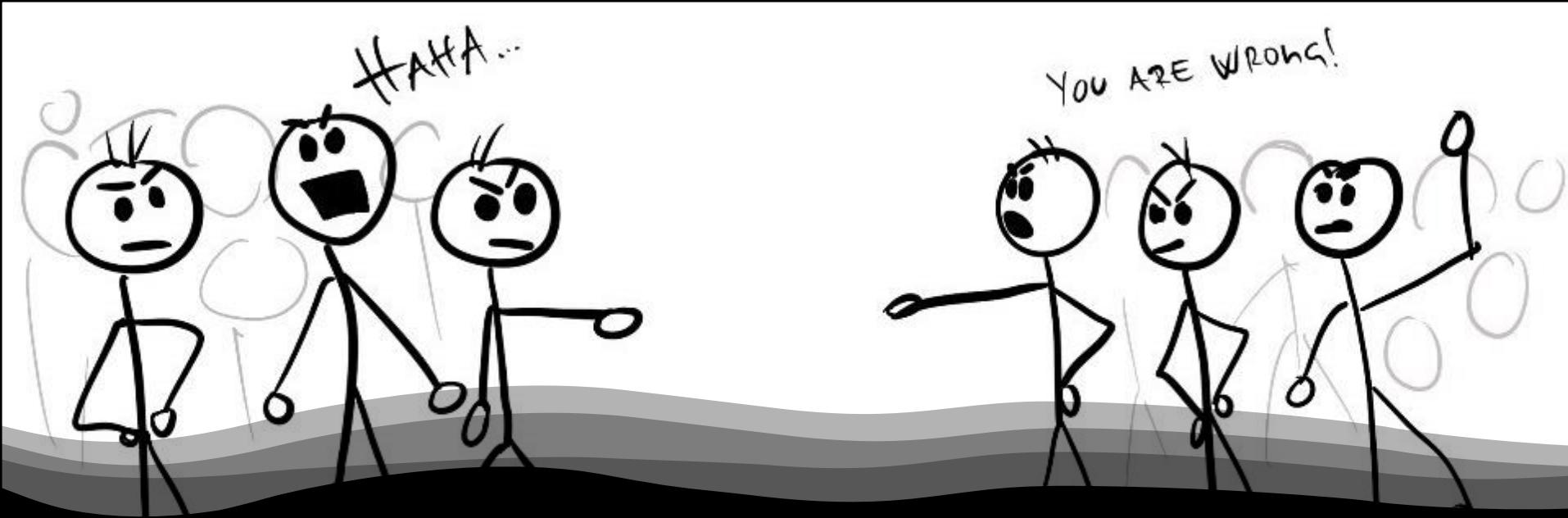
No forzar todo para escribirlo como user stories

Material Bibliográfico de Referencia

- Libro:
 - Cohn, Mike - USER STORIES APPLIED – Editorial Addison Wesley 2004- Capítulos 1, 2 y 6
- Papers
 - Dean Leffingwell and Pete Behrens – A user story primer (2009)
- Link
 - <http://www.mountaingoatsoftware.com/>

Estimate OR #noEstimate?





No ser dogmático sobre nada.

Ser pragmático es la clave del éxito en casi todo.

Tips respecto de las estimaciones...



Si las estimaciones se utilizan como compromisos son muy peligrosas y perjudiciales para cualquier organización.



Lo más beneficioso en las estimaciones es el “proceso de hacerlas”.



La estimación podría servir como una gran respuesta temprana sobre si el trabajo planificado es factible o no.



La estimación puede servir como una gran protección para el equipo.

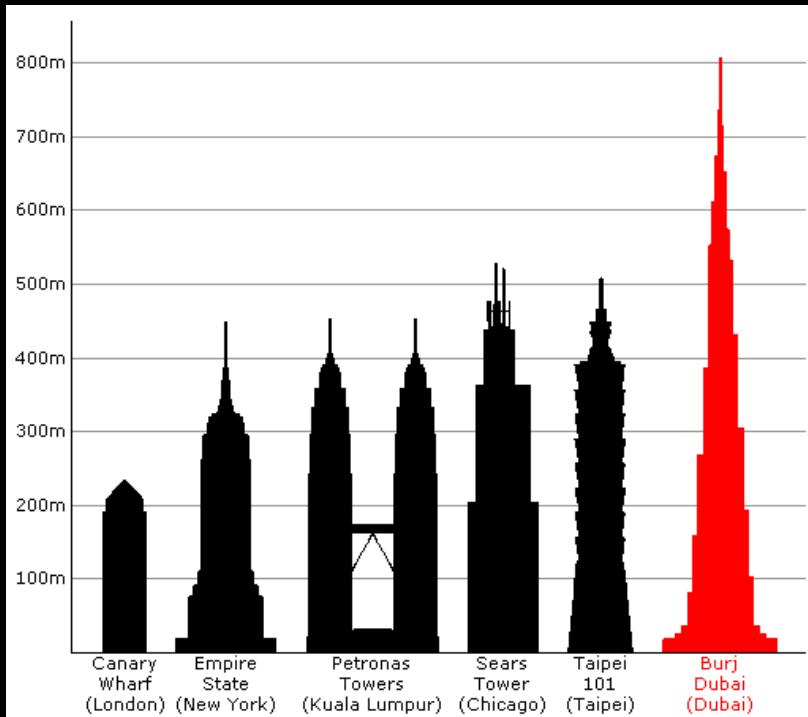
Estimaciones Ágiles

- Las features/stories son estimadas usando una medida de tamaño relativo conocida como story points (SP)
- Las medidas relativas no son absolutas.
- Story Points no es una medida basada en tiempo



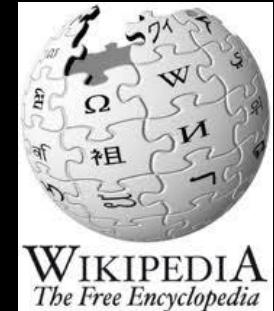
Estimación Relativa

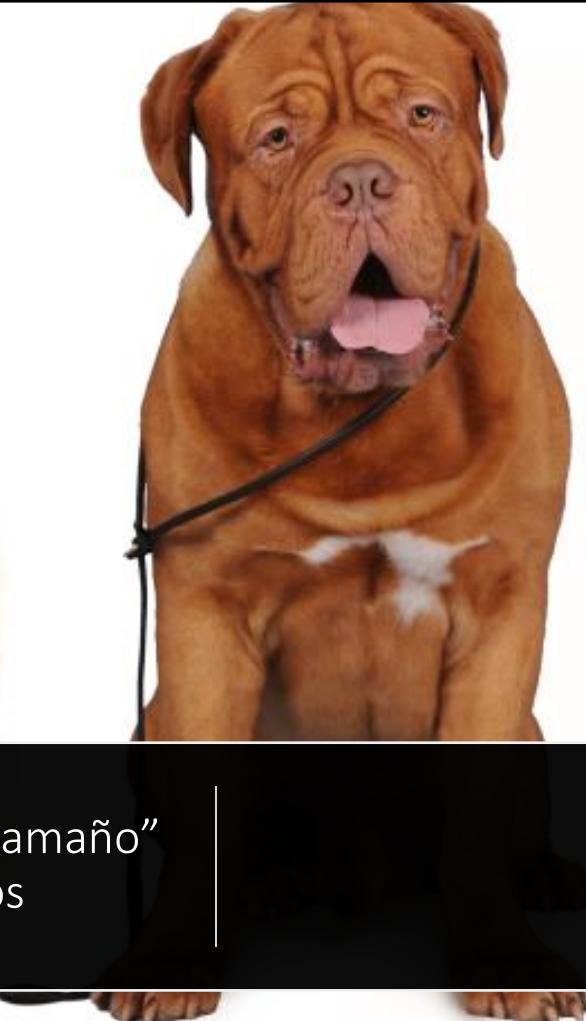
- Las personas no saben estimar en términos absolutos
- Somos buenos comparando cosas
- Comparar es generalmente más rápido.
- Se obtiene una mejor dinámica grupal y pensamiento de equipo más que individual
- Se emplea mejor el tiempo de análisis de las storys



y...pero, el tamaño?

- “La palabra tamaño refiere a cuan grande o pequeño es algo”
- El tamaño es una medida de la cantidad de trabajo necesaria para producir una feature/story.
- El tamaño indica:
 - Cuán compleja es una feature/story
 - Cuánto trabajo es requerido para hacer o completar una feature/story
 - Y cuán grande es una feature/story





Por favor, “dé tamaño”
de estos perros

Descripción del Trabajo vs. Esfuerzo

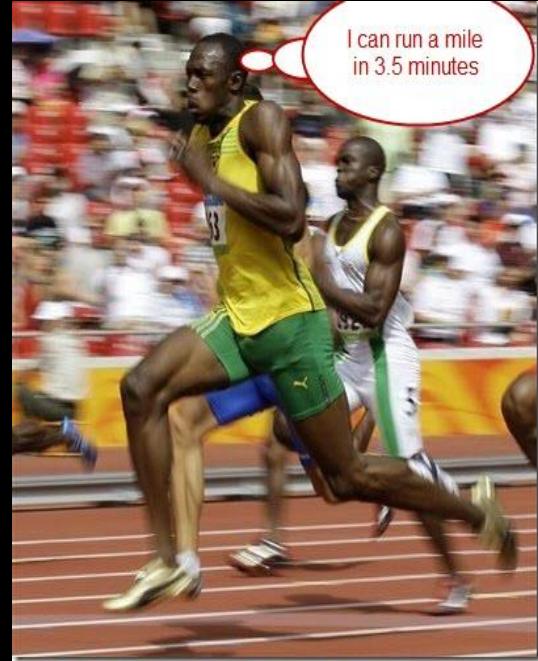


Tamaño Vs. Esfuerzo



Las estimaciones basadas en tiempo son más propensas a errores debido a varias razones.

- Habilidades
- Conocimiento
- Asunciones
- Experiencia
- Familiaridad con los dominios de aplicación/negocio



Tamaño NO ES esfuerzo

Y qué hacemos con el tamaño!?????

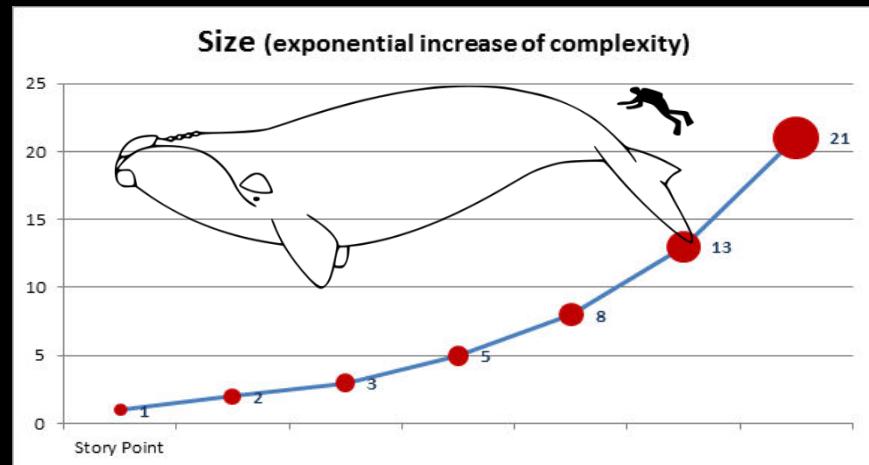
- Tamaño por números: 1 a 10
- Talles de remeras: S, M, L, XL, XXL
- Serie 2^n : 1, 2, 4, 8, 16, 32, 64, etc.
- Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.

Una vez elegida la escala no se cambia! Si se cambia cambiamos el metro patrón



Y qué es un Story Point? (una de muchas definiciones)

- Es una unidad de medida específica (del equipo) de, complejidad, riesgo y esfuerzo, es lo que “el kilo” a la unidad de nuestro sistema de medición de peso
- Story point da idea del “peso” de cada story y decide cuan grande (compleja) es
- La complejidad de una
 - feature/story tiende a
 - incrementarse exponencialmente.



3 Stories que queremos estimar

Story 1



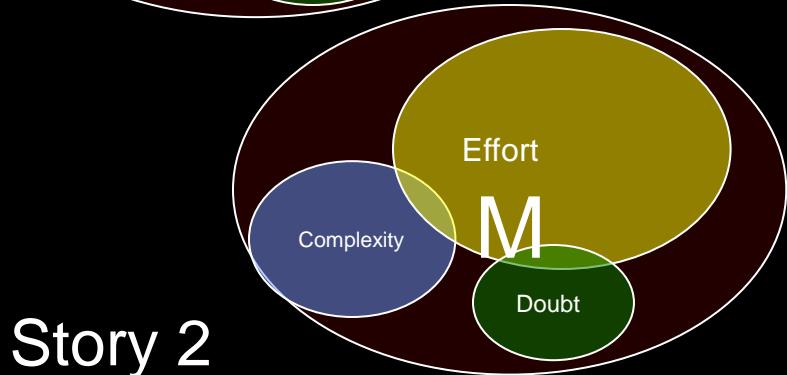
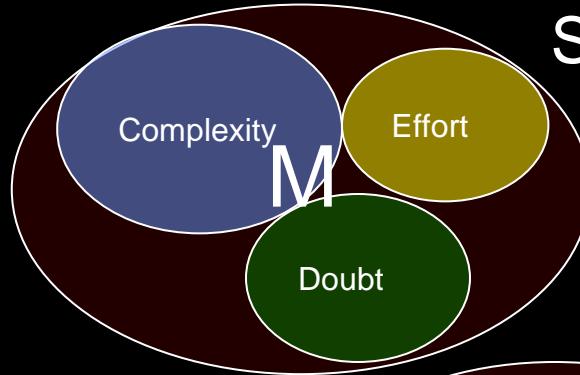
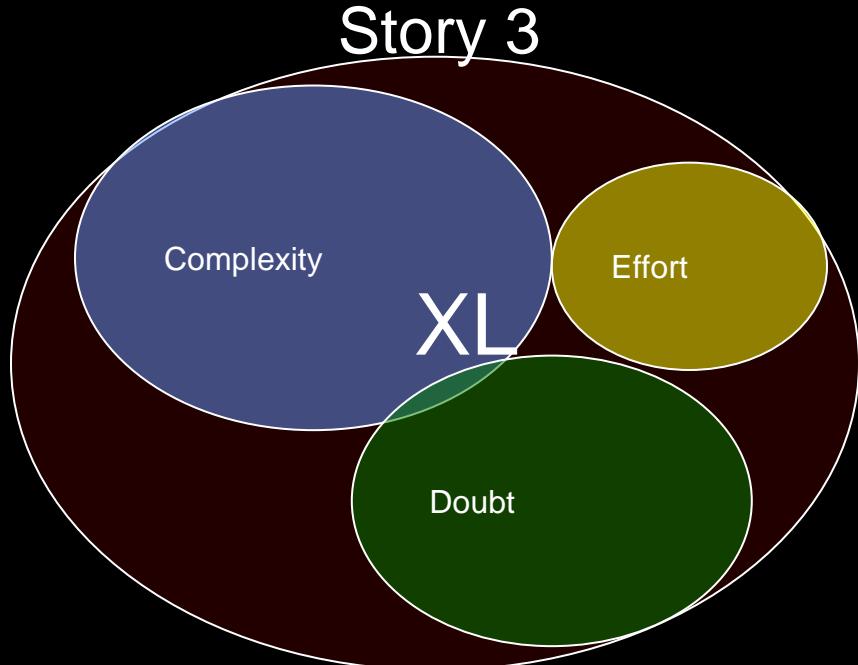
Story 2



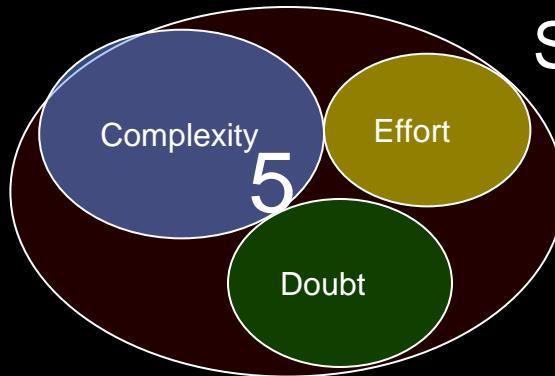
Story 3



“tamaño” de las Stories

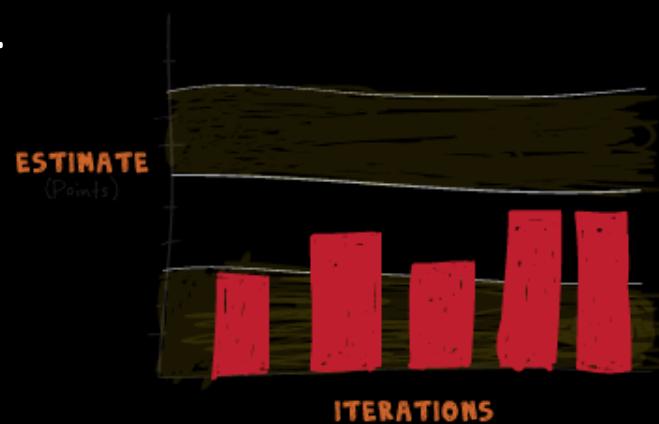


Y si usamos números?



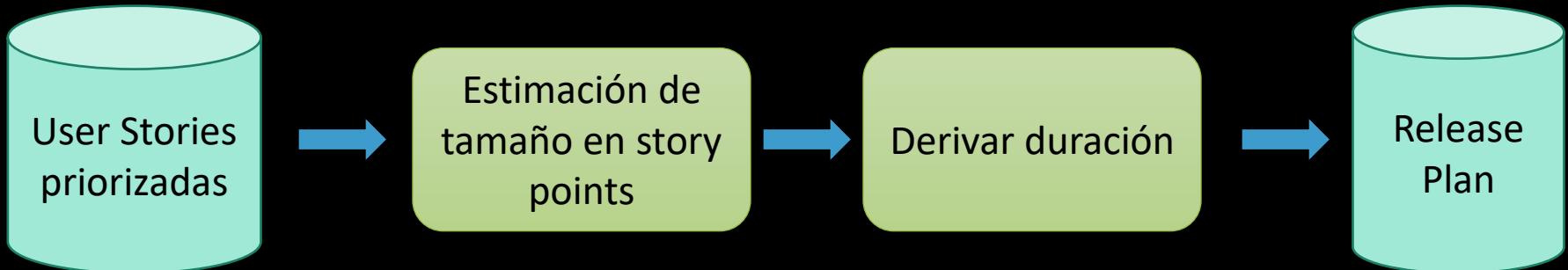
Velocidad (Velocity)

- **Velocidad/ Velocity** es una medida (métrica) del progreso de un equipo. Se calcula sumando el número de story points (asignados a cada user story) que el equipo **completa** durante la **iteración**.
- Se cuentan los **story poins** de las **Users Storys** que están **completas**, no parcialmente completas.
- La Velocidad corrige los errores de estimación.



¿Y cómo hago con un proyecto?

- Si estimo User Storys cómo hago para estimar un proyecto?
 - La duración de un proyecto no se “estima”, se deriva.... tomando el número total de story points de sus user storys y dividiéndolo por la velocidad del equipo.



- La velocidad nos ayuda a determinar un horizonte de planificación apropiando
- La estimación en story points separa completamente la estimación de esfuerzo de la estimación de la duración.

Una Propuesta de método de estimación



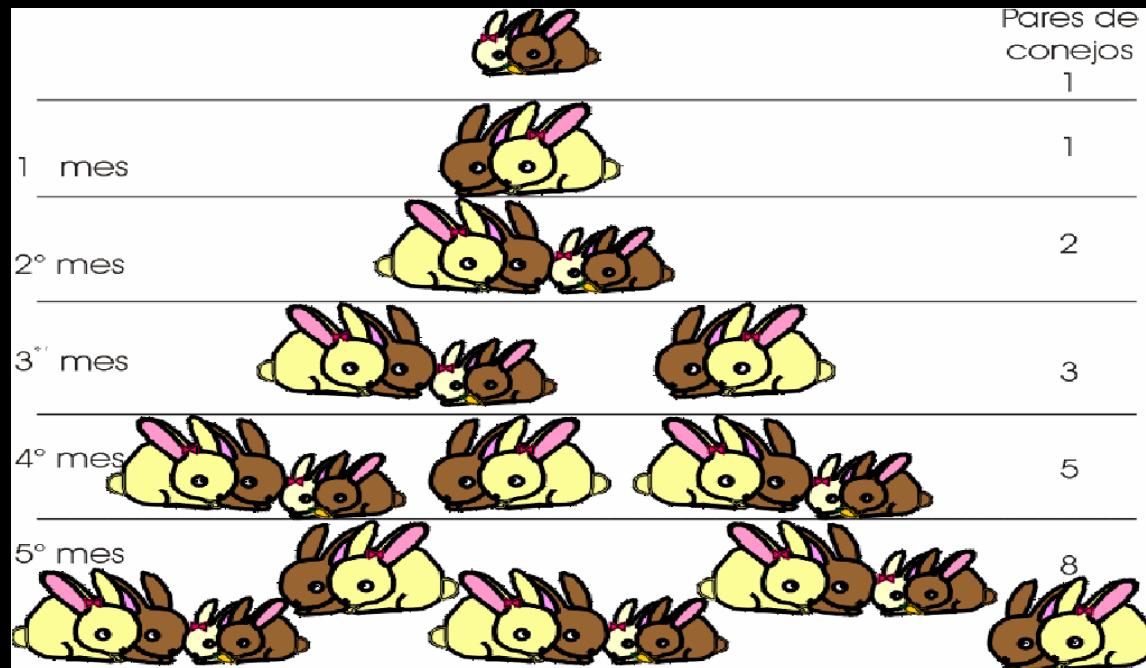
Poker estimation

- Popular entre los Agile practicioners, publicado por Mike Cohn
- Combina opinión de experto, analogía y desegregación.
- Participantes en “planning poker” son desarrolladores
 - “Las personas más competentes en resolver una tarea deben ser quienes las estiman”

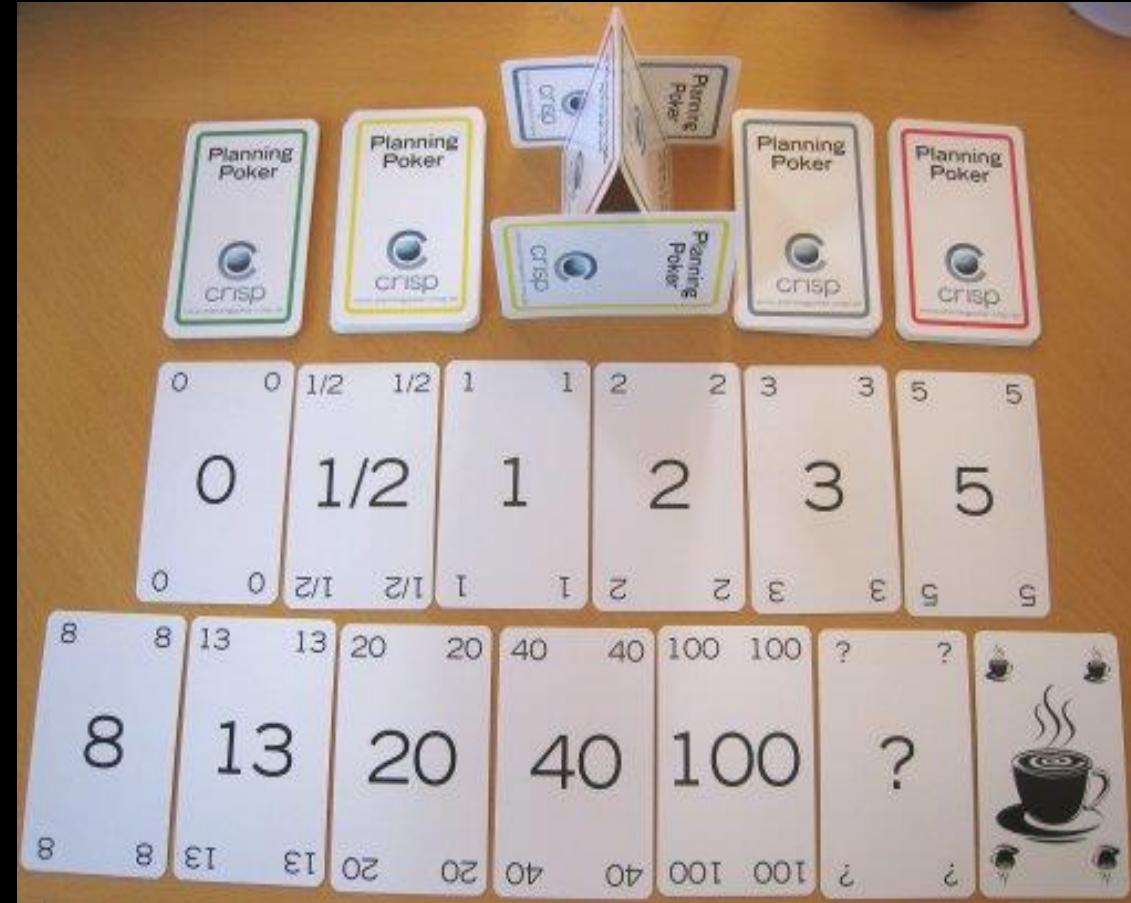


Fibonacci (se acuerdan?)

- La secuencia empieza en 1 y cada numero subsecuente es la suma de los dos precedentes. (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144....)

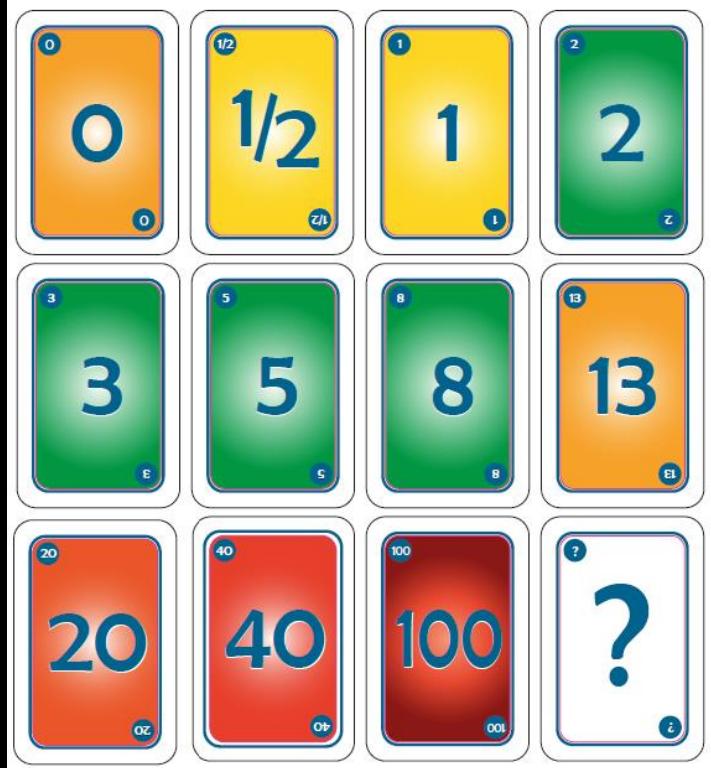


Poker Planning



<http://www.planningpoker.com/detail.html>

¿Cómo “decodificar” las estimaciones?



- **0**: Quizás ud. no tenga idea de su producto o funcionalidad en este punto.
- **$\frac{1}{2}$, 1**: funcionalidad pequeña (usualmente cosmética).
- **2-3**: funcionalidad pequeña a mediana. Es lo que queremos. ☺
- **5**: Funcionalidad media. Es lo que queremos ☺
- **8**: Funcionalidad grande, de todas formas lo podemos hacer, pero hay que preguntarse sino se puede partir o dividir en algo más pequeño. No es lo mejor, pero todavía ☺
- **13**: Alguien puede explicar por que no lo podemos dividir?
- **20**: Cuál es la razón de negocio que justifica semejante story y más fuerte aún, por qué no se puede dividir?.
- **40**: no hay forma de hacer esto en un sprint.
- **100**: confirmación de que está algo muy mal. Mejor ni arrancar.

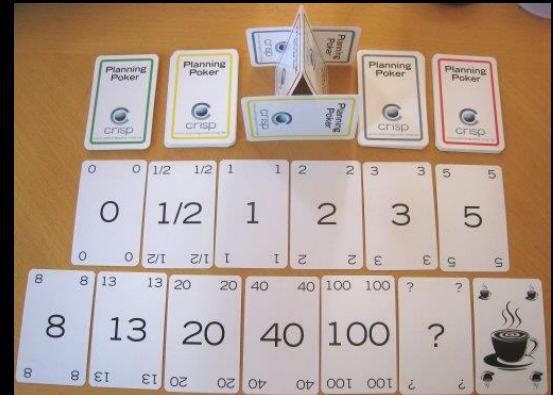
Poker Planning

Prerrequisitos:

- Lista de features/stories a ser estimadas
- Cada estimador tiene un mazo de cartas.

Pasos:

1. Determine la **base story** (la canónica) que será usada para comprar con las otras stories. Digamos, Story Z.
 - 1.1 La story a ser estimada se lee a todo el equipo.
 - 1.2 Los estimadores discuten la story, haciendo preguntas al product owner (las que se necesiten).
 - 1.3 Cada estimador selecciona una carta y pone la carta boca abajo en la mesa.
 - 1.4 Cuando todos pusieron las cartas, las mismas se exponen al mismo tiempo.
 - 1.5 Si todos los estimadores selecciona el mismo valor, ese es el estimado. Sino, los estimadores discuten sus resultados, poniendo especial atención en los más altos y los más bajos. Después de la charla, GOTO to 1.3.
2. Se toma la próxima story, se discute con el product owner.
3. Cada estimador asigna a la story un valor por comparación contra la base story. “Cuan grande/pequeña, compleja, riesgosa es esta story comparada con Story Z?”. GOTO to 1.3

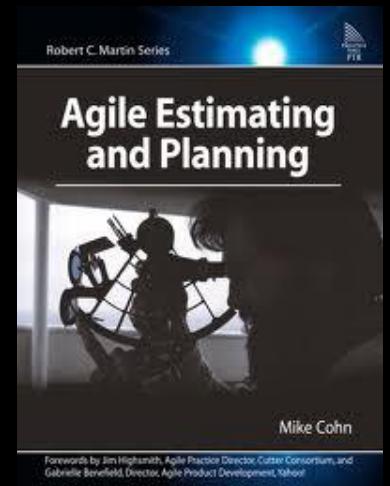
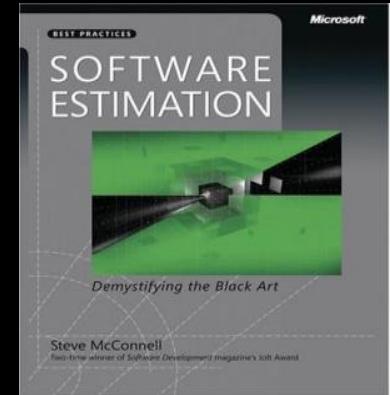


Agile es empírico,
Inspeccionar y adaptar
es mandatorio!



Bibliografía

- Software Estimation: Demystifying the Black Art
by Steve McConnell
Microsoft Press 2006 ISBN:0735605351
- Agile Estimating and Planning, Mike Cohn,
Pearson Education, ISBN: 9780137126347
- <http://www.planningpoker.com/detail.html>





Universidad Tecnológica Nacional

Facultad Regional Córdoba

Cátedra de Ingeniería de Software

Docentes: Judith Meles – Laura Covaro

ESTIMACIONES DE SOFTWARE

“PREDICTION IS VERY DIFFICULT, ESPECIALLY
ABOUT THE FUTURE.”

*La predicción es muy difícil, especialmente
acerca del futuro.*

—NIELS BOHR,

Algunas consideraciones

Por definición una estimación no es precisa.

Estimar no es planear y planear no es estimar.

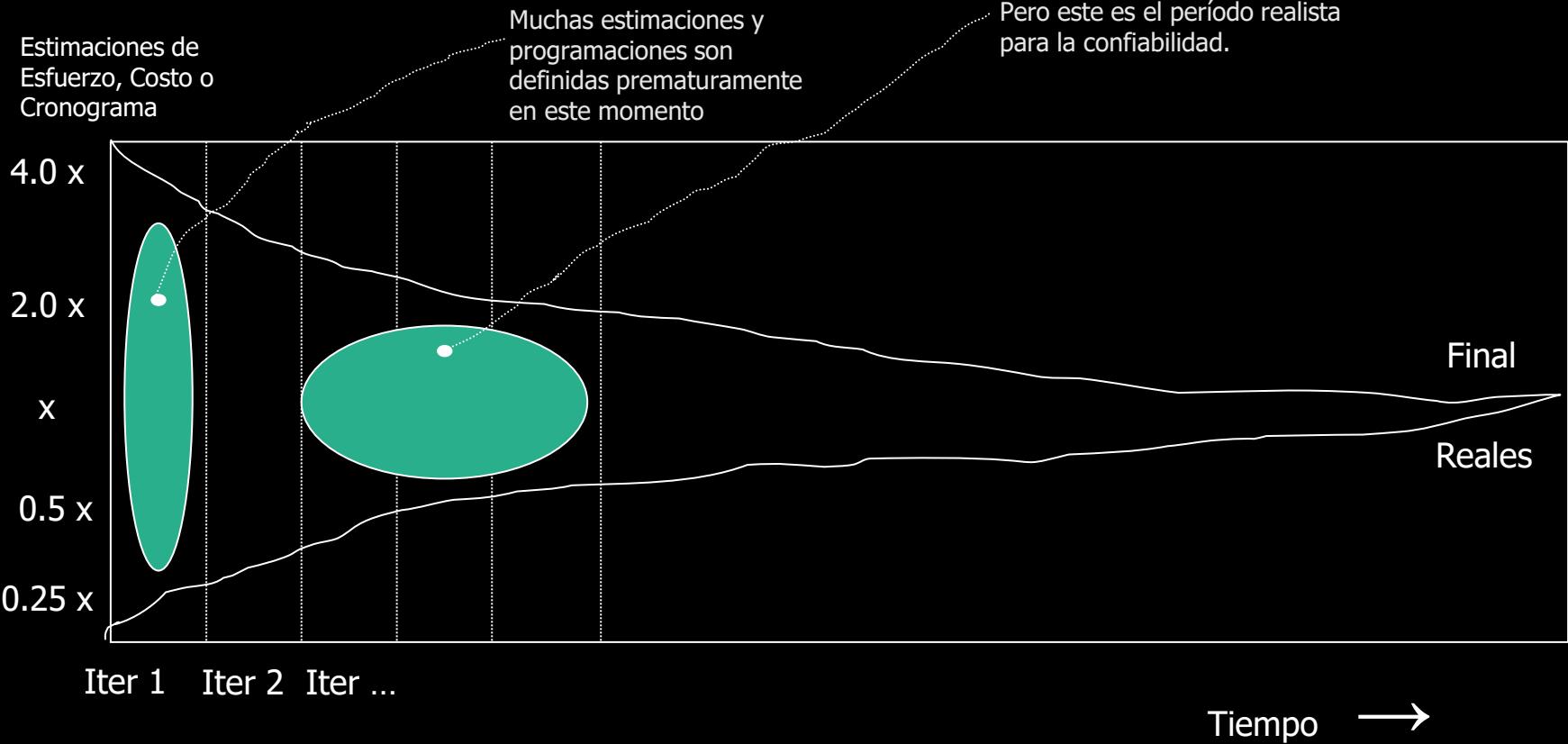
Las estimaciones son la base de los planes, pero los planes no tienen que ser lo mismo que lo estimado.

A mayor diferencia entre lo estimado y lo planeado mayor riesgo.

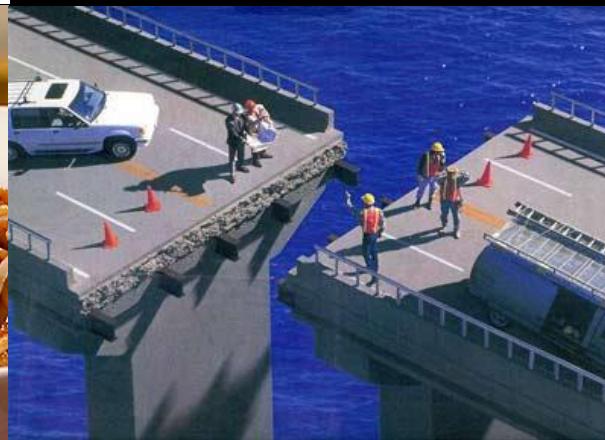
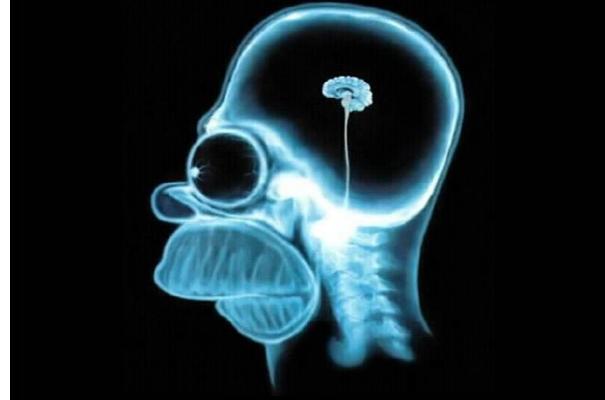
Las estimaciones no son compromisos.

¿Para qué estimamos?

4

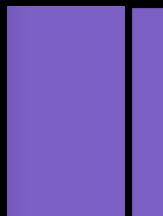


¿De dónde vienen los errores de estimación?





Técnicas
fundamentales
de estimación
- Contar

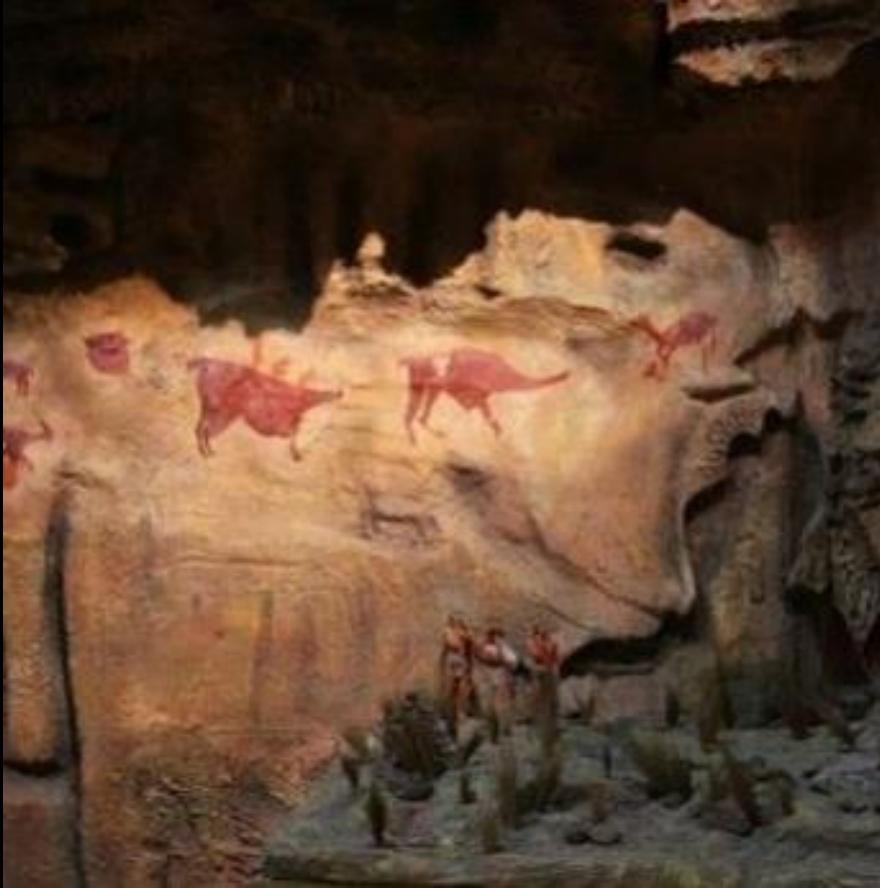


Métodos utilizados

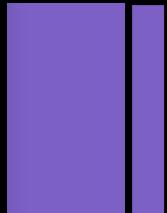
- Basados en la experiencia.
- Basados exclusivamente en los recursos.
- Método basado exclusivamente en el mercado.
- Basados en los componentes del producto o en el proceso de desarrollo.
- Métodos algorítmicos

Métodos
basados en la
experiencia:

- Datos Históricos
- Juicio experto
 - Puro,
 - Delphi
- Analogía



Datos
históricos





01	January	04	April	02	July	10	October
M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
02	February	05	May	08	August	11	November
M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
03	March	06	June	09	September	12	December
M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	M T W T F S S	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30



¿Qué datos históricos necesito?

Juicio experto: Puro

- Un experto estudia las especificaciones y hace su estimación.
- Se basa fundamentalmente en los conocimientos del experto.
- Si desaparece el experto, la empresa deja de estimar





Juicio de Experto

- Es el enfoque de estimaciones más utilizado en la práctica.
- Acerca del 75% de organizaciones de software usan principalmente "juicio de experto"
- Experto en qué?

Estructurando el Juicio de Experto

- Tenga tareas una granularidad aceptable.
- Use el método de “optimista, pesimista y habitual” y su formula = ($o + 4h + p)/6$
- Use un checklist y un criterio definido para asegurar cobertura.



Mucho cuidado
con todo esto, es
un buen
comienzo, pero un
péssimo final

Juicio experto: Wideband Delphi

- Un grupo de personas son informadas y tratan de adivinar lo que costará el desarrollo tanto en esfuerzo, como en duración.
- Las estimaciones en grupo suelen ser mejores que las individuales.



Wideband Delphi

Se dan las especificaciones a un grupo de expertos.

Se les reúne para que discutan tanto el producto como la estimación.

Remiten sus estimaciones individuales al coordinador.

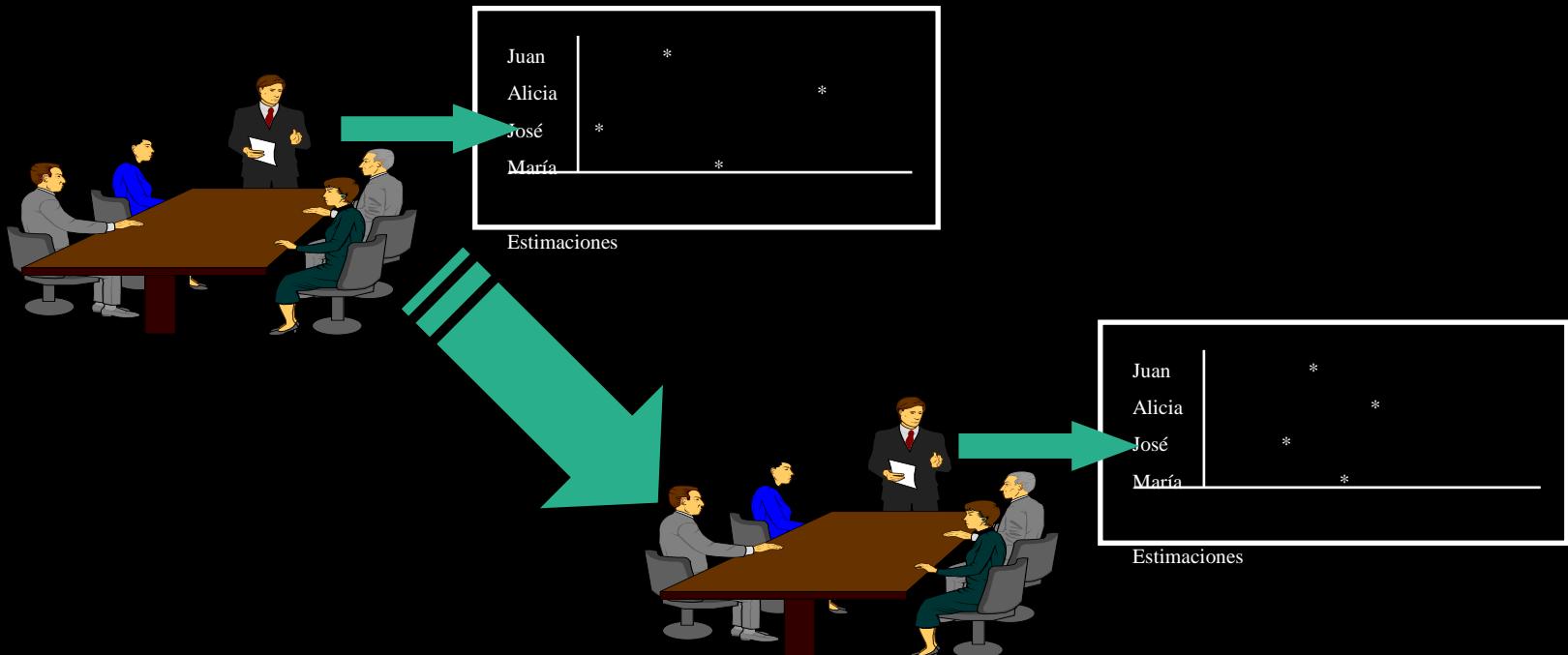
Cada estimador recibe información sobre su estimación, y las ajenas pero de forma anónima.

Se reúnen de nuevo para discutir las estimaciones.

Cada uno revisa su propia estimación y la envía al coordinador.

Se repite el proceso hasta que la estimación converge de forma razonable.

Método de trabajo del Wideband Delphi



Actividades Omitidas

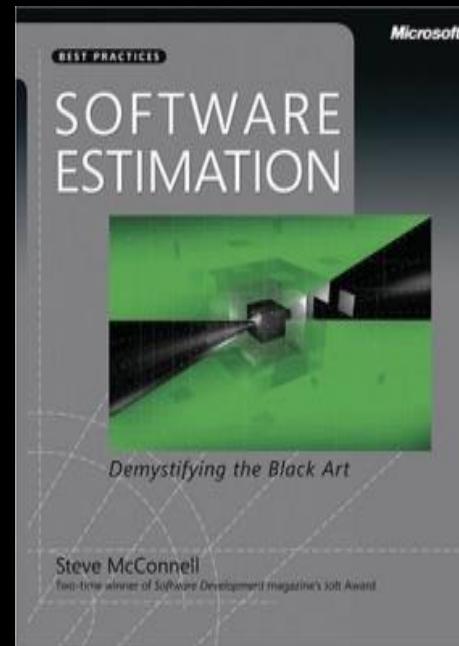
- Una de la fuentes de error mas común en las estimaciones es omitir actividades necesarias para las estimación del proyecto.
 - Requerimientos faltantes.
 - Actividades de desarrollo faltantes (documentación técnica, participación en revisiones, creación de datos para el testing, mantenimiento de producto en previas versiones)
 - Actividades generales. (días por enfermedad, licencias, cursos, reuniones de compañía).
- Uso de buffers

“Nunca tenga temor que estimaciones creadas por desarrolladores sean demasiado pesimistas, dado que los desarrolladores siempre generan cronogramas demasiado optimistas”.

Chris Peters, Microsoft VP

Bibliografía

- Software Estimation:
Demystifying the Black Art
by Steve McConnell
Microsoft Press 2006
ISBN:0735605351

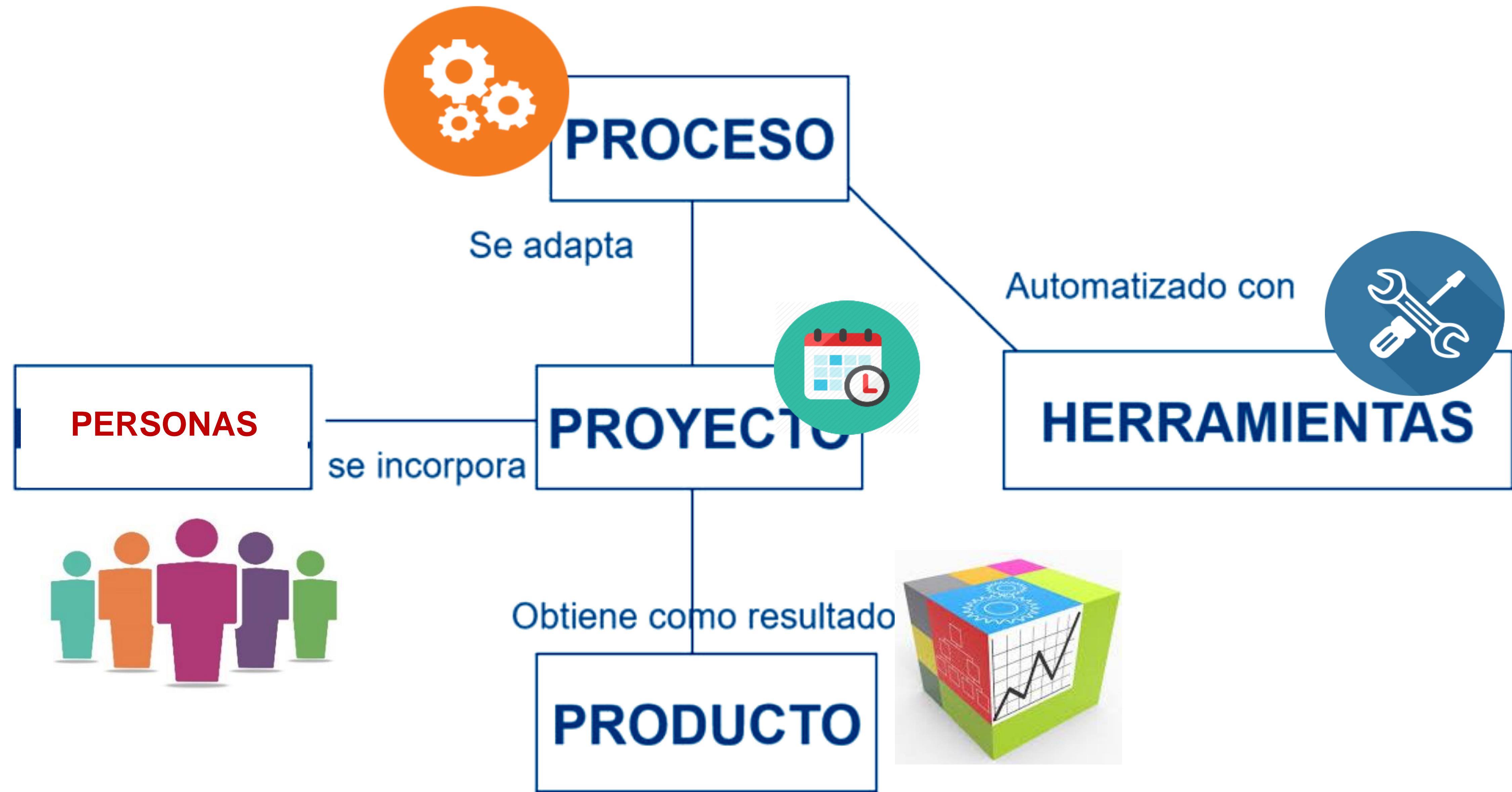


Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra de Ingeniería de Software
Docentes: Judith Meles- Laura Covaro

Software Configuration Management (SCM)

. (o más allá del Commit, Update)

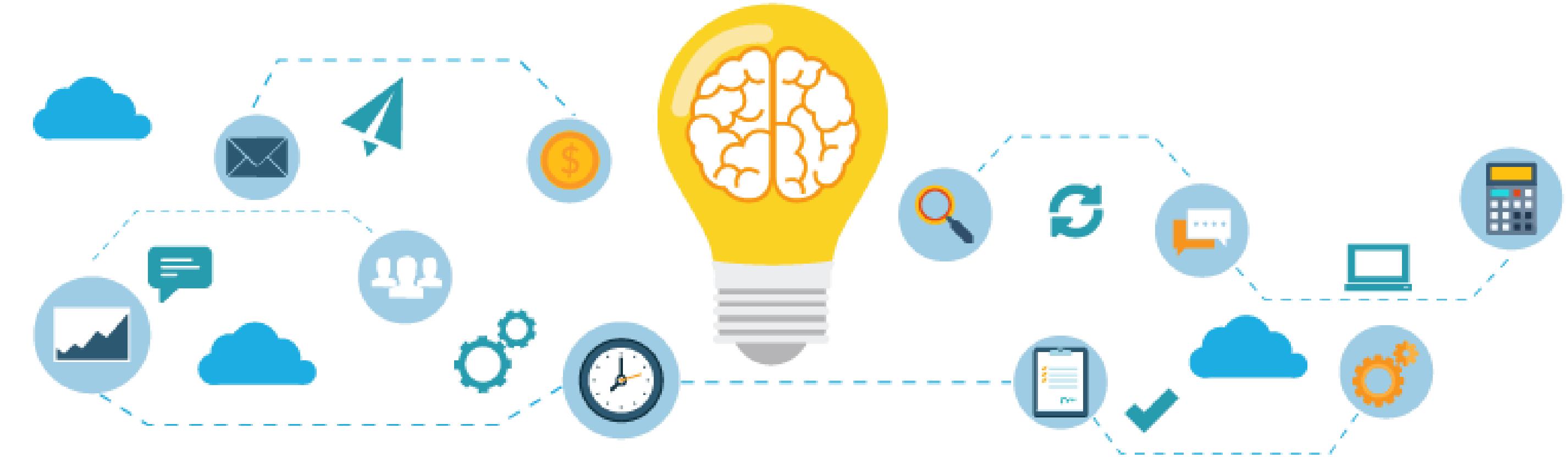
Software en contexto



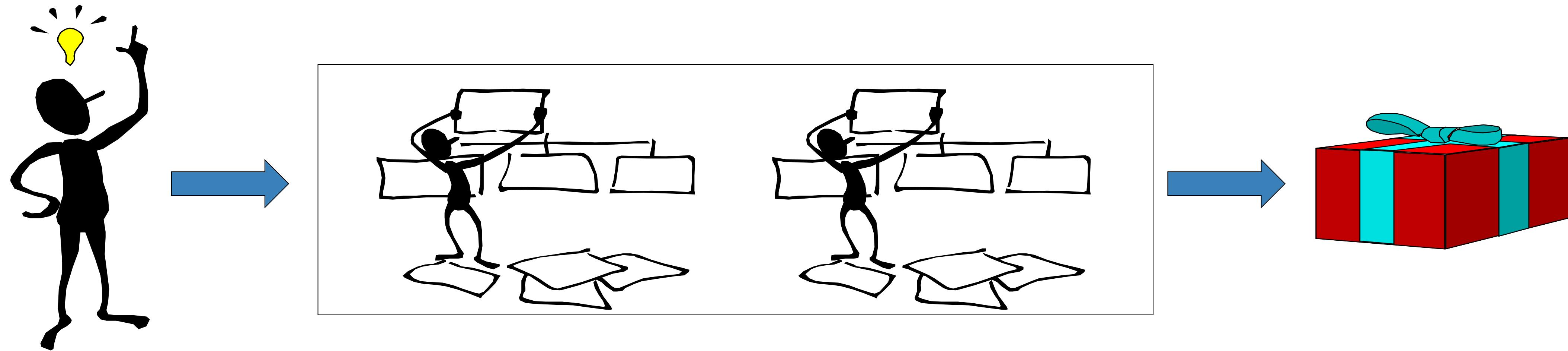
¿Cuándo pensamos en Software... en qué pensamos?

Conjunto de:

- *Programas*
- *Procedimientos*
- *Reglas*
- *Documentación*
- *Datos*



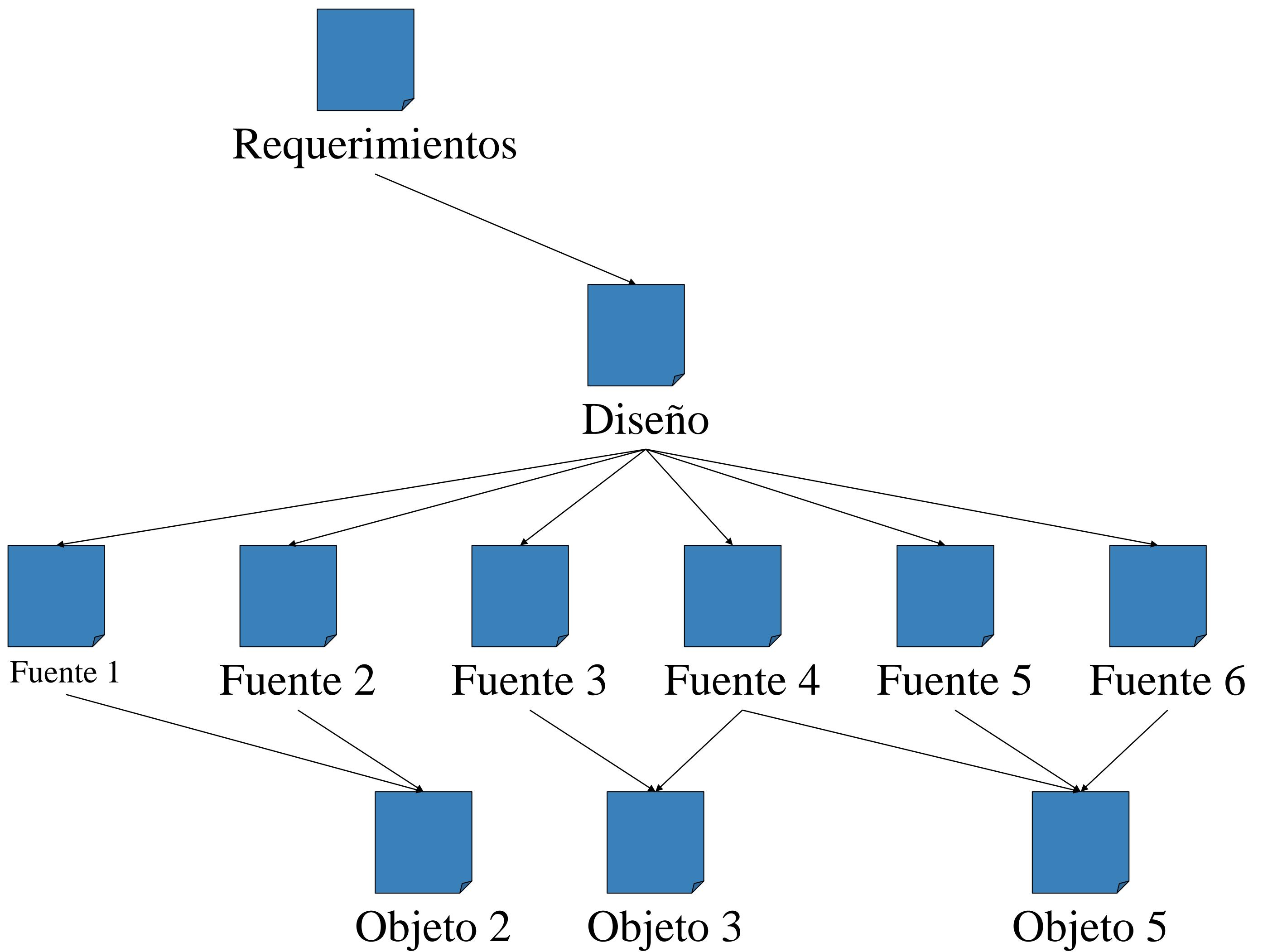
El Software



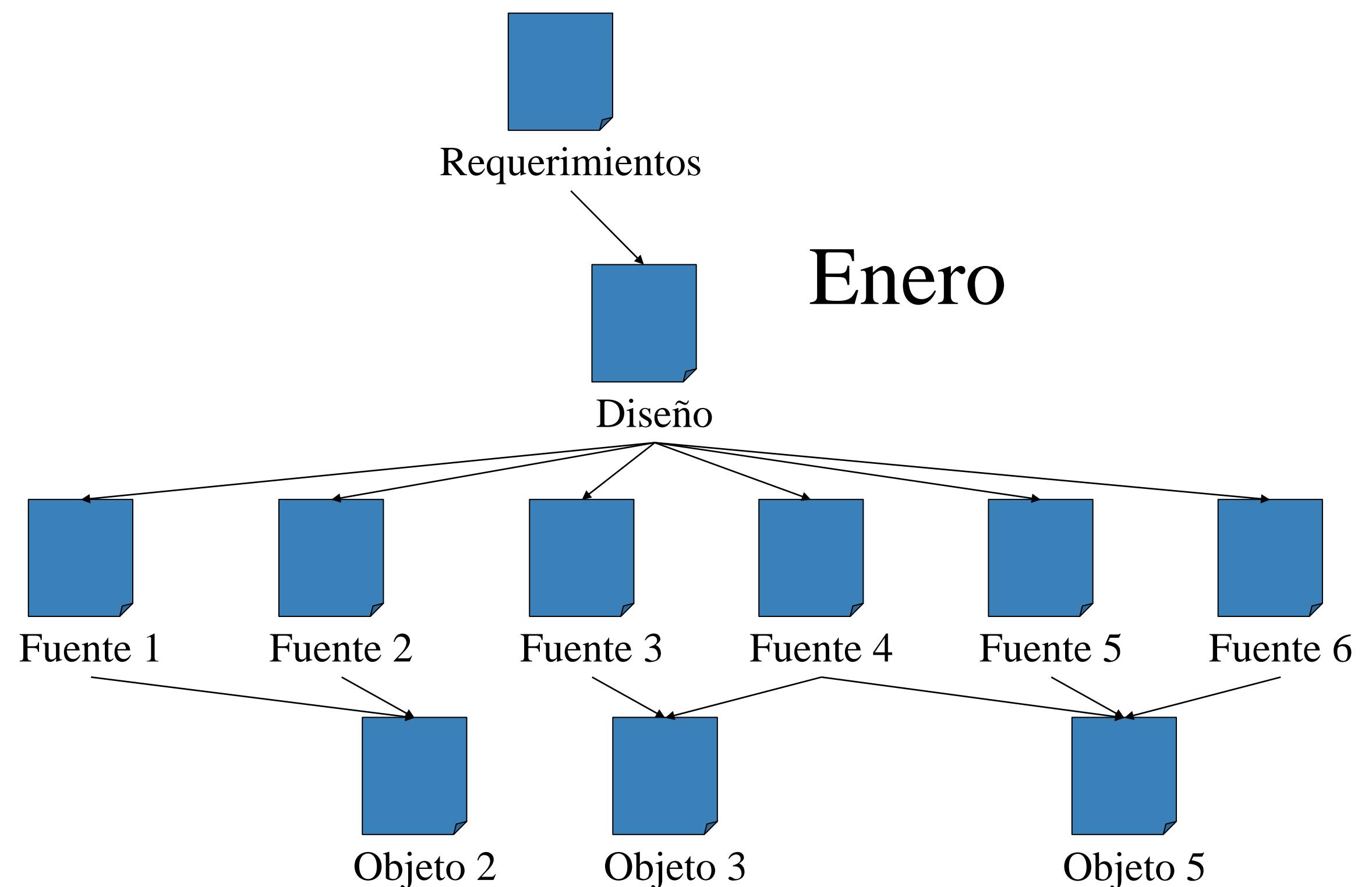
- Información:
 - estructurada con propiedades lógicas y funcionales.
 - creada y mantenida en varias formas y representaciones.
 - confeccionada para ser procesada por computadora en su estado más desarrollado

El software

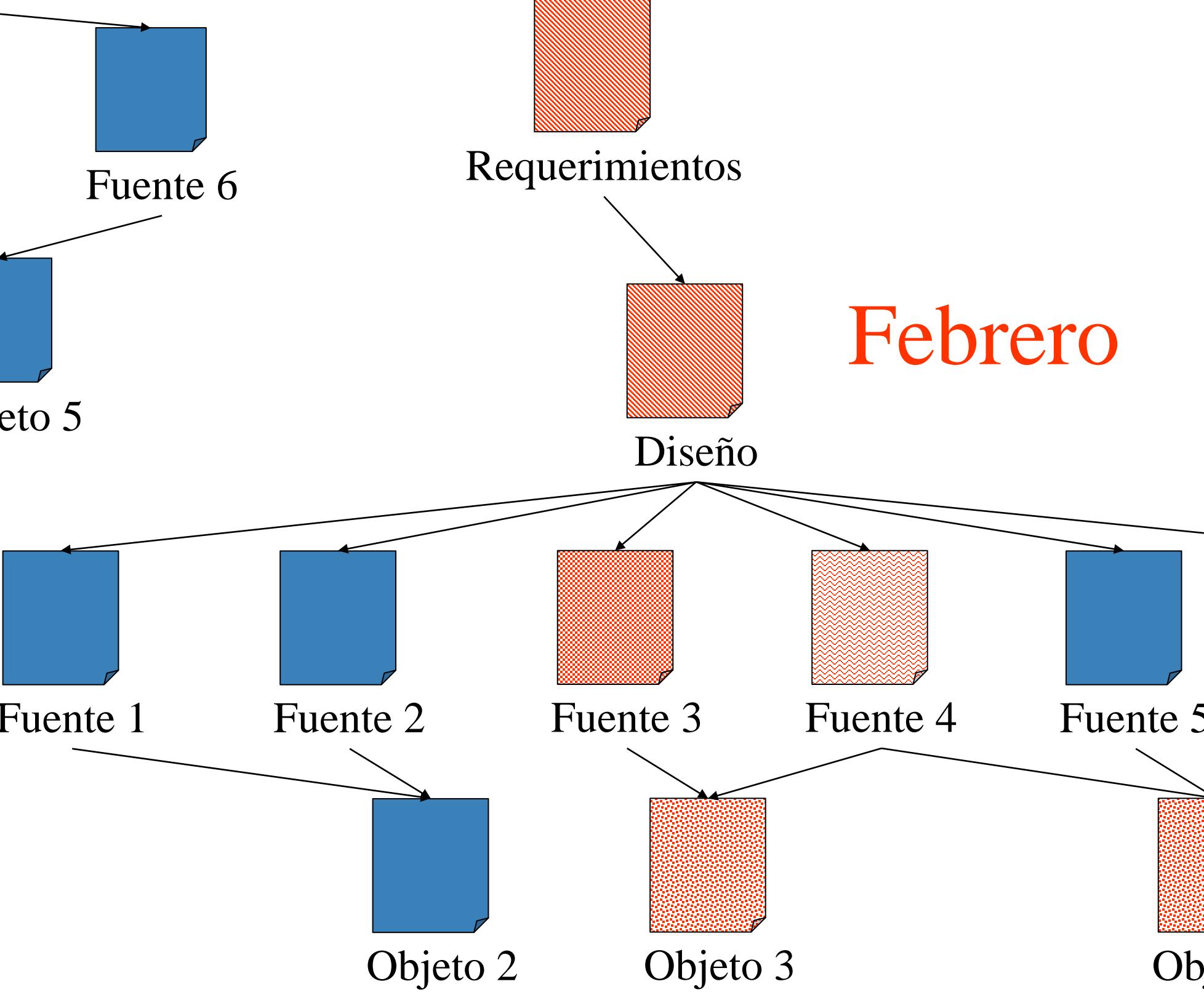
Enero



La evolución del software

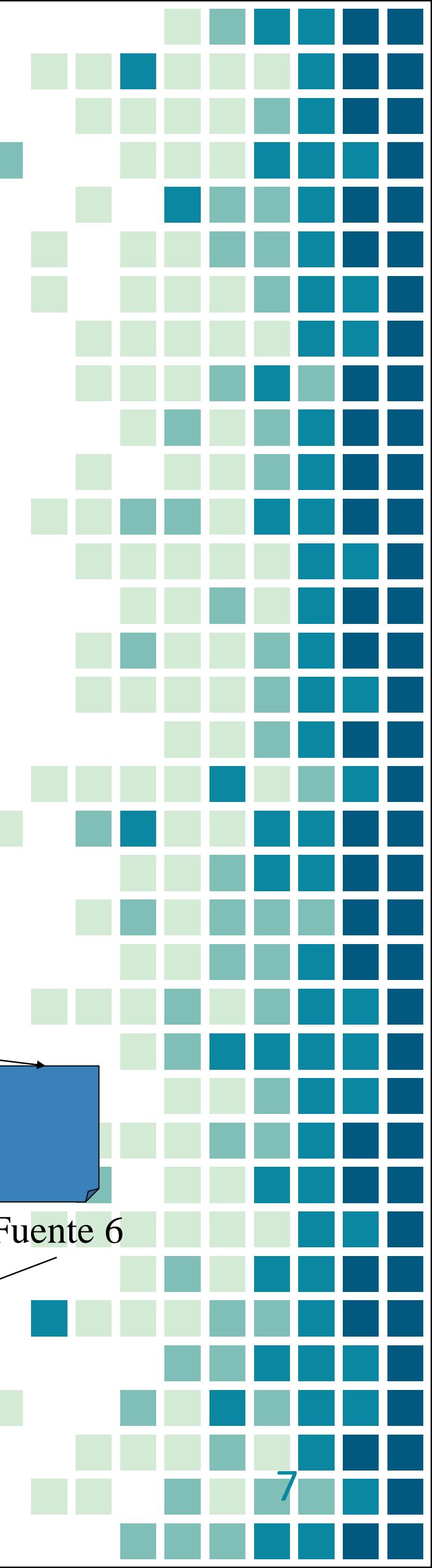
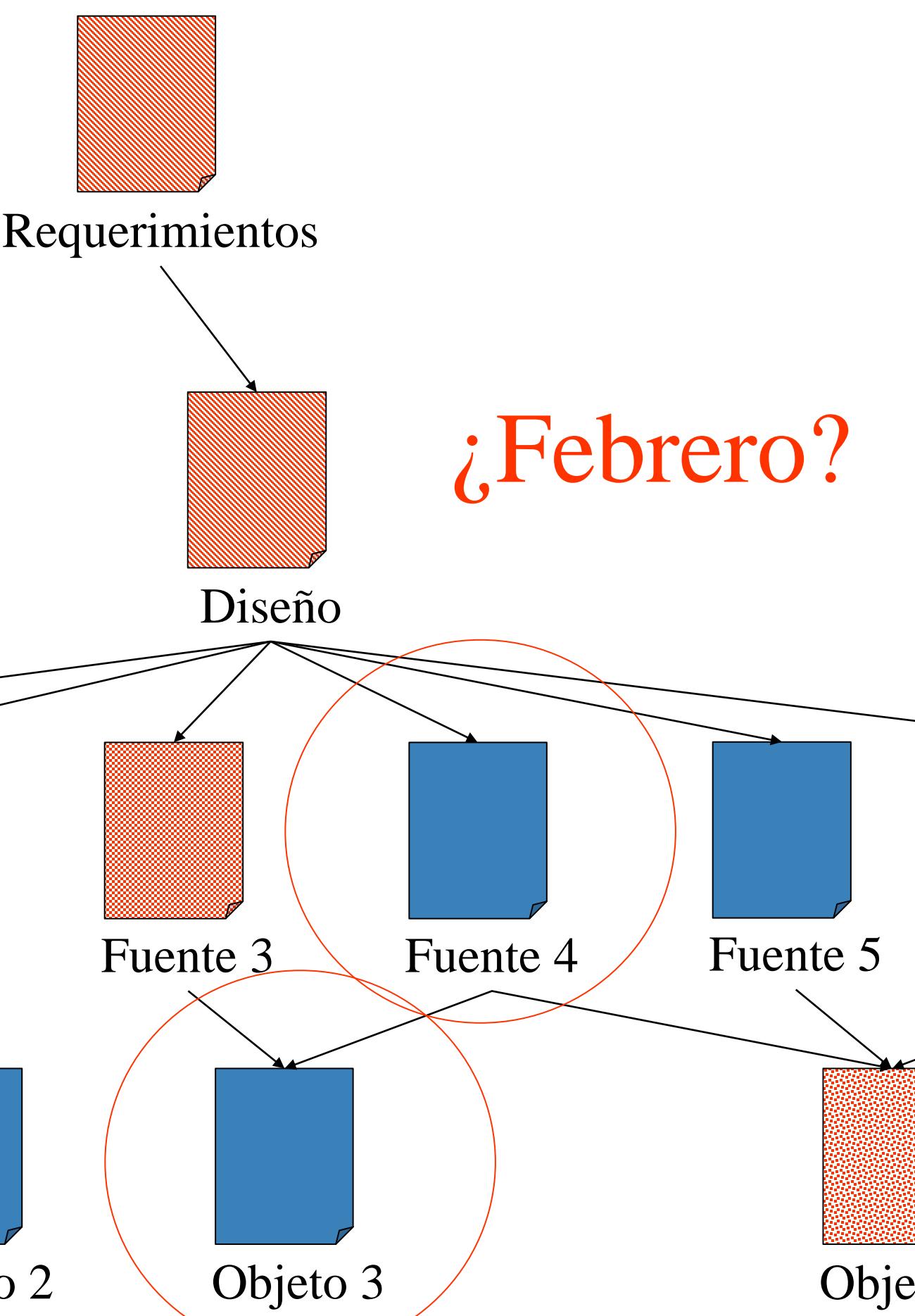
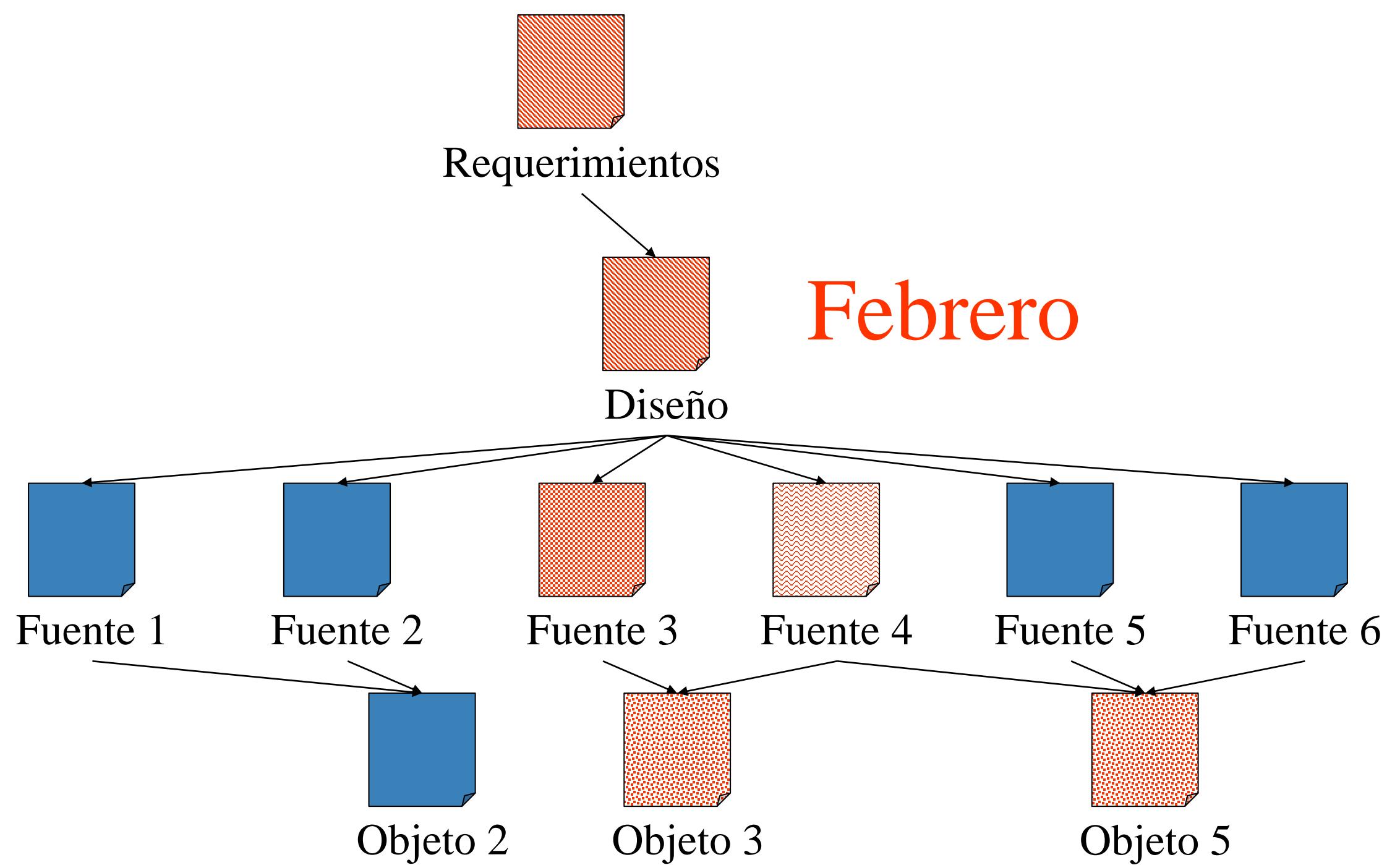


Enero



Febrero

Problemas



Cambios en el Software

Tienen su origen en:

- ❖ Cambios del negocio y nuevos requerimientos
- ❖ Soporte de cambios de productos asociados
- ❖ Reorganización de las prioridades de la empresa por crecimiento
- ❖ Cambios en el presupuesto
- ❖ Defectos encontrados a corregir
- ❖ Oportunidades de mejora

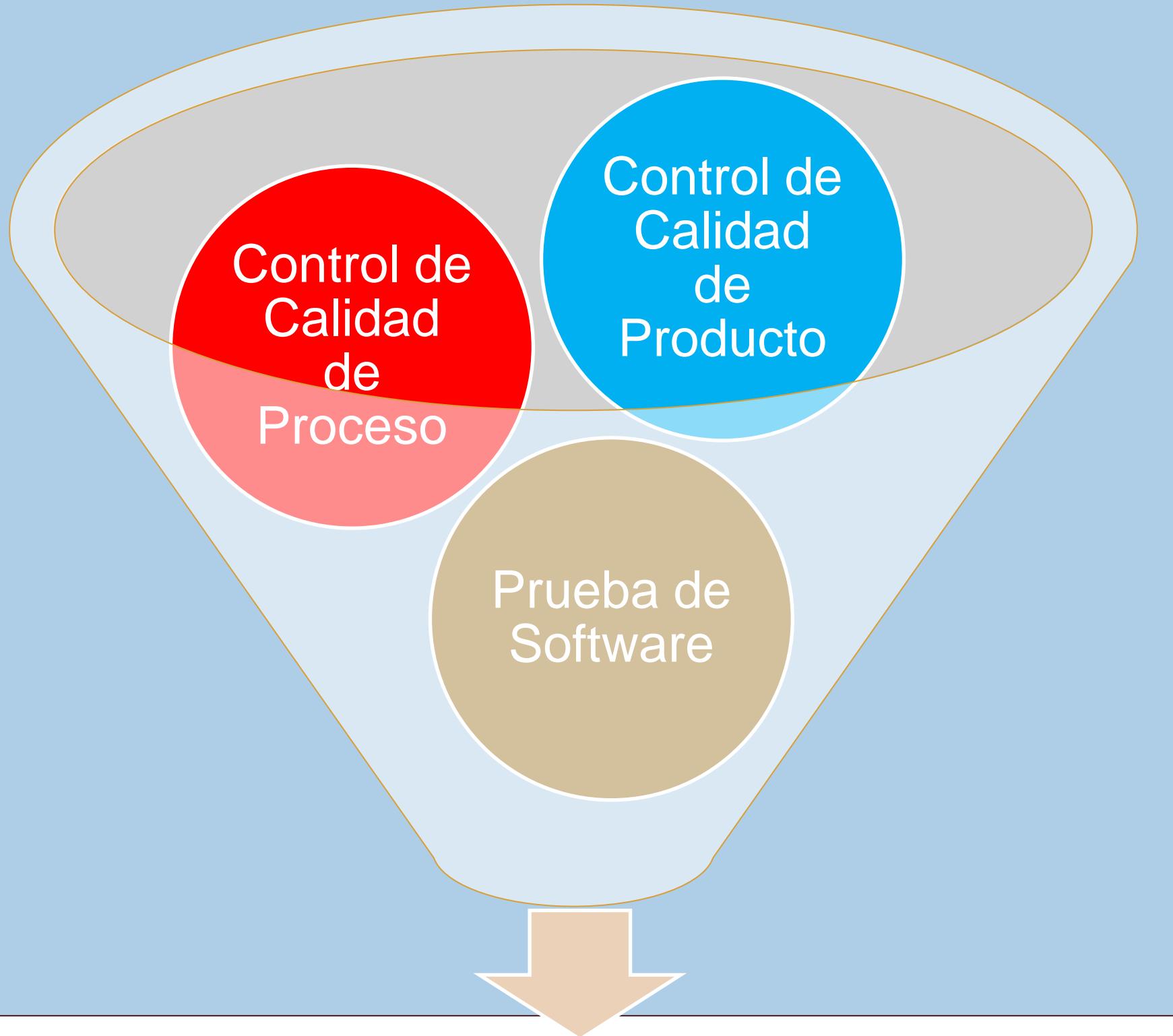
SCM como disciplina de soporte

Es una actividad “paragüas”, transversal a todo el proyecto con aplicación en las diferentes disciplinas.



Disciplinas de soporte del Software

Administración de
Configuración de Software



Aseguramiento de
Calidad de Software

Un poco de Historia



Tiene su origen a mediados de 1950s, cuando CM (por Configuration Management) originalmente utilizado para desarrollo de hardware y control de producción, fue utilizado en el desarrollo de software.

Definición

Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos

(ANSI/IEEE 828, 1990)

¿Por qué deberíamos gestionar la configuración?

Su propósito es establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida.

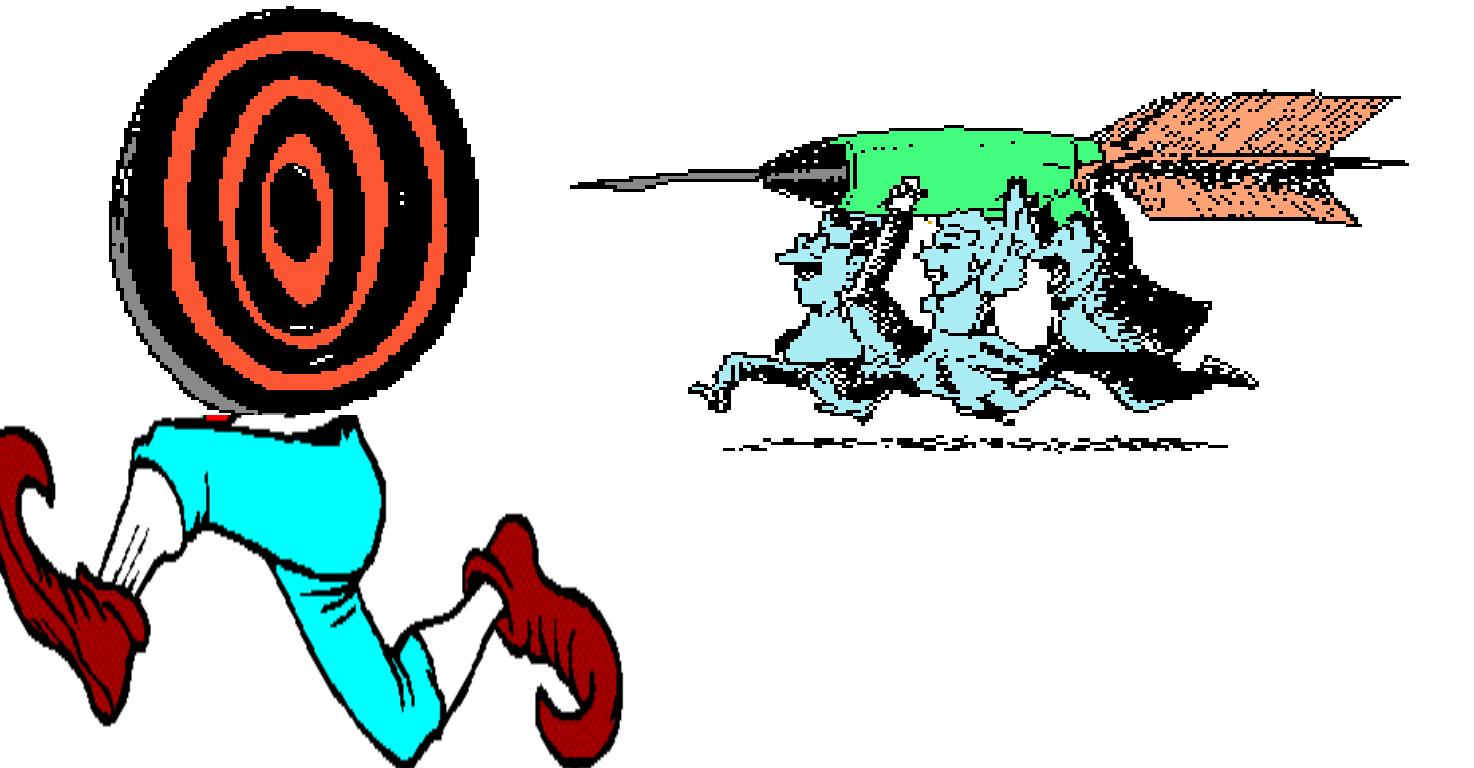
Involucra para la configuración:

- ❖ Identificarla en un momento dado
- ❖ Controlar sistemáticamente sus cambios
- ❖ Mantener su integridad y origen

Integridad del Producto

- satisface las necesidades del usuario
- puede ser fácil y completamente rastreado durante su ciclo de vida
- satisface criterios de performance
- cumple con sus expectativas de costo

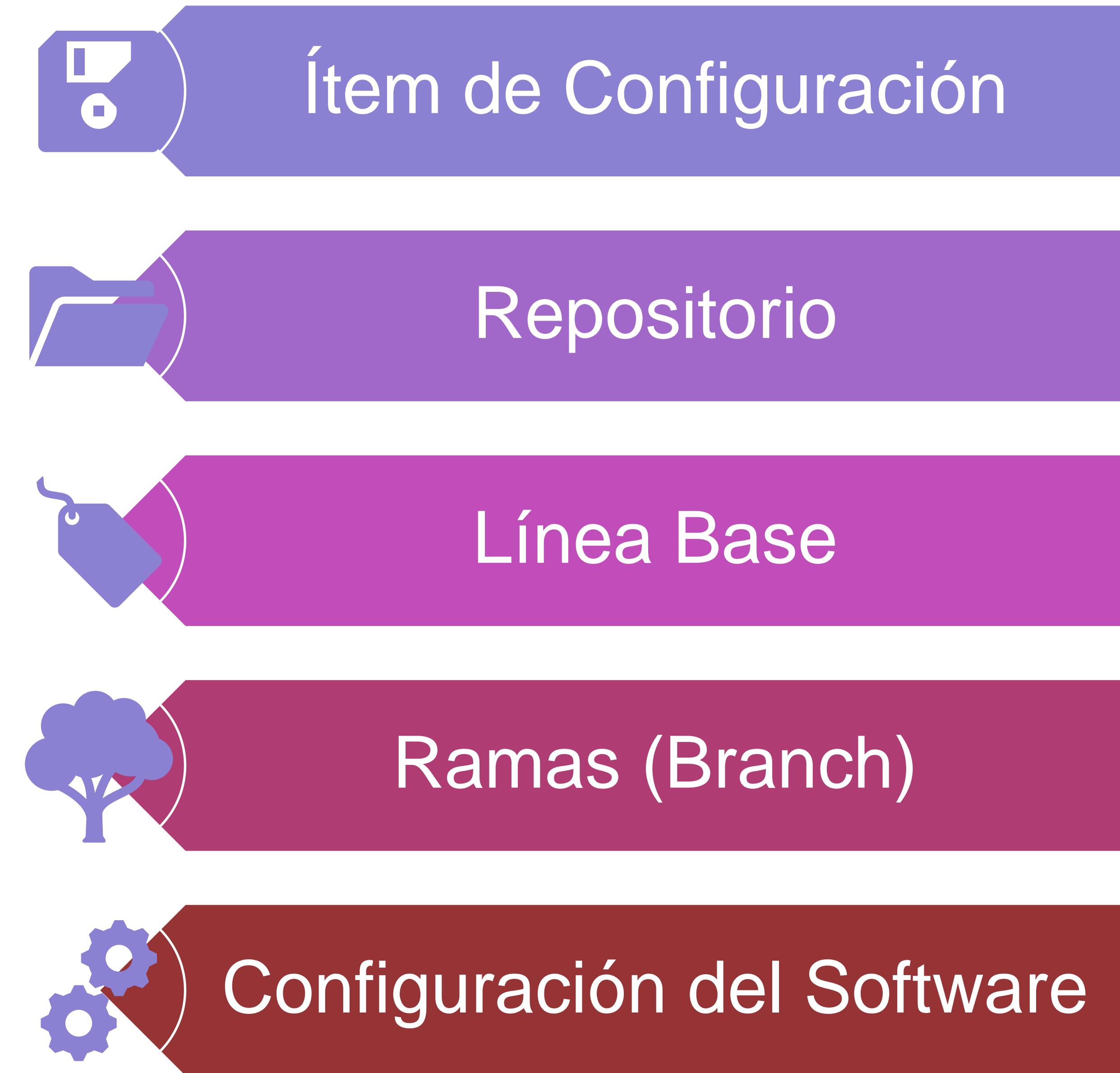
El software: un blanco móvil



Problemas en el manejo de componentes

- ❖ Pérdida de un componente
- ❖ Pérdida de cambios (el componente que tengo no es el último)
- ❖ Sincronía fuente - objeto – ejecutable
- ❖ Regresión de fallas
- ❖ Doble mantenimiento
- ❖ Superposición de cambios
- ❖ Cambios no validados

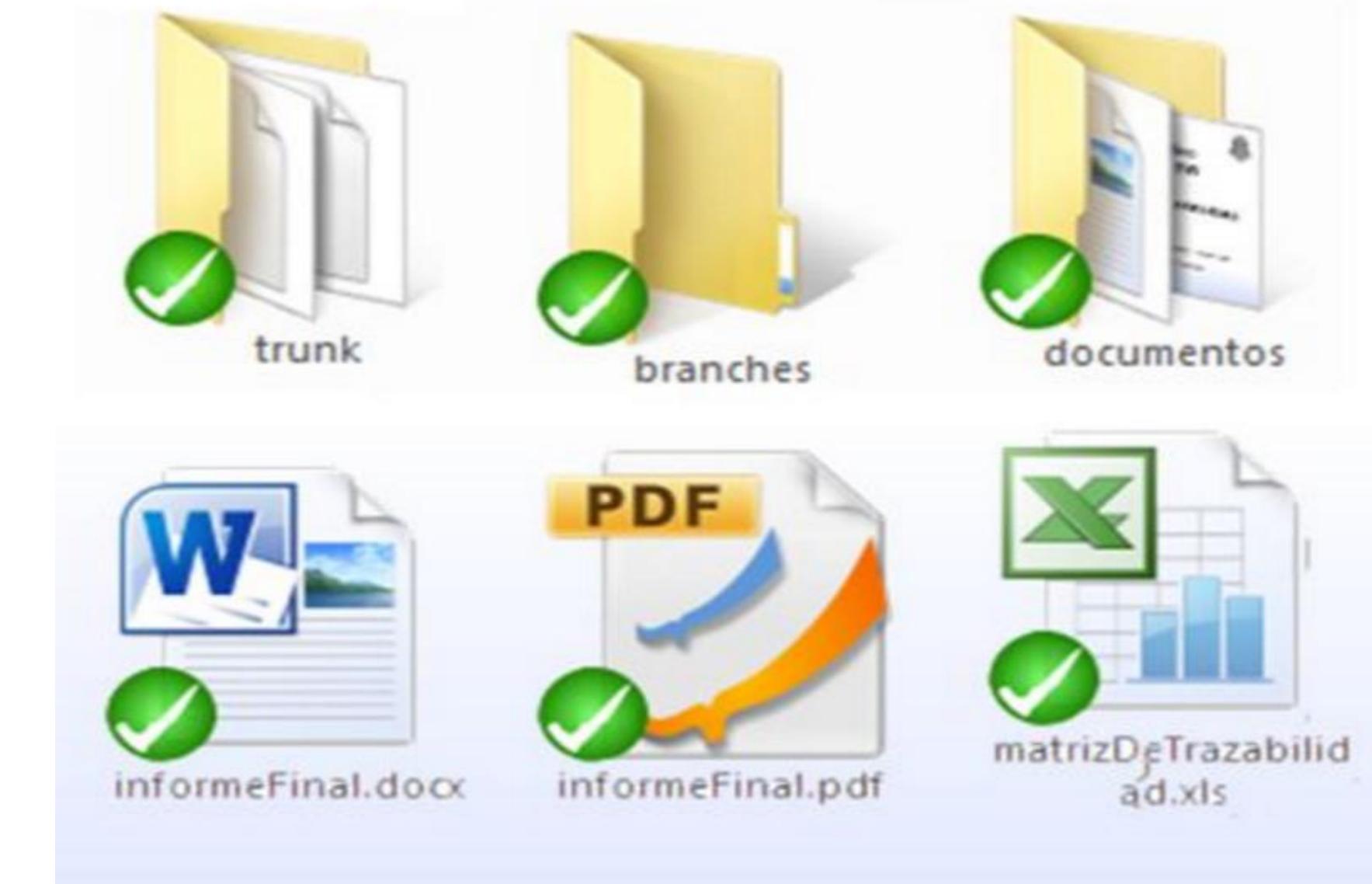
Algunos Conceptos Clave para la Gestión de Configuración de Software



Ítem de Configuración de Software (SCI)

- ✓ Documentos de diseño, código fuente, código ejecutable, etc.

Se llama **ítem de configuración** (IC) a todos y cada uno de los artefactos que forman parte del producto o del proyecto, que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución.

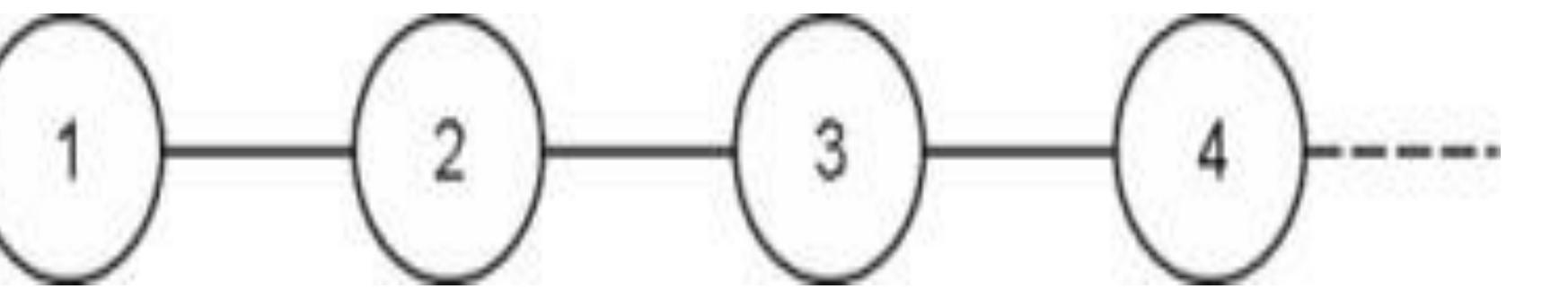


Algunos ejemplos de Ítems de Configuración

- ❖ Plan de CM
- ❖ Propuestas de Cambio
- ❖ Visión
- ❖ Riesgos
- ❖ Plan de desarrollo
- ❖ Prototipo de Interfaz
- ❖ Guía de Estilo de IHM
- ❖ Manual de Usuario
- ❖ Requerimientos
- ❖ Plan de Calidad
- ❖ Arquitectura del Software
- ❖ Plan de Integración
- ❖ Planes de Iteración
- ❖ Estándares de codificación
- ❖ Casos de prueba
- ❖ Código fuente
- ❖ Gráficos, iconos, ...
- ❖ Instructivo de ensamble
- ❖ Programa de instalación
- ❖ Documento de despliegue
- ❖ Lista de Control de entrega
- ❖ Formulario de aceptación
- ❖ Registro del proyecto

Versión

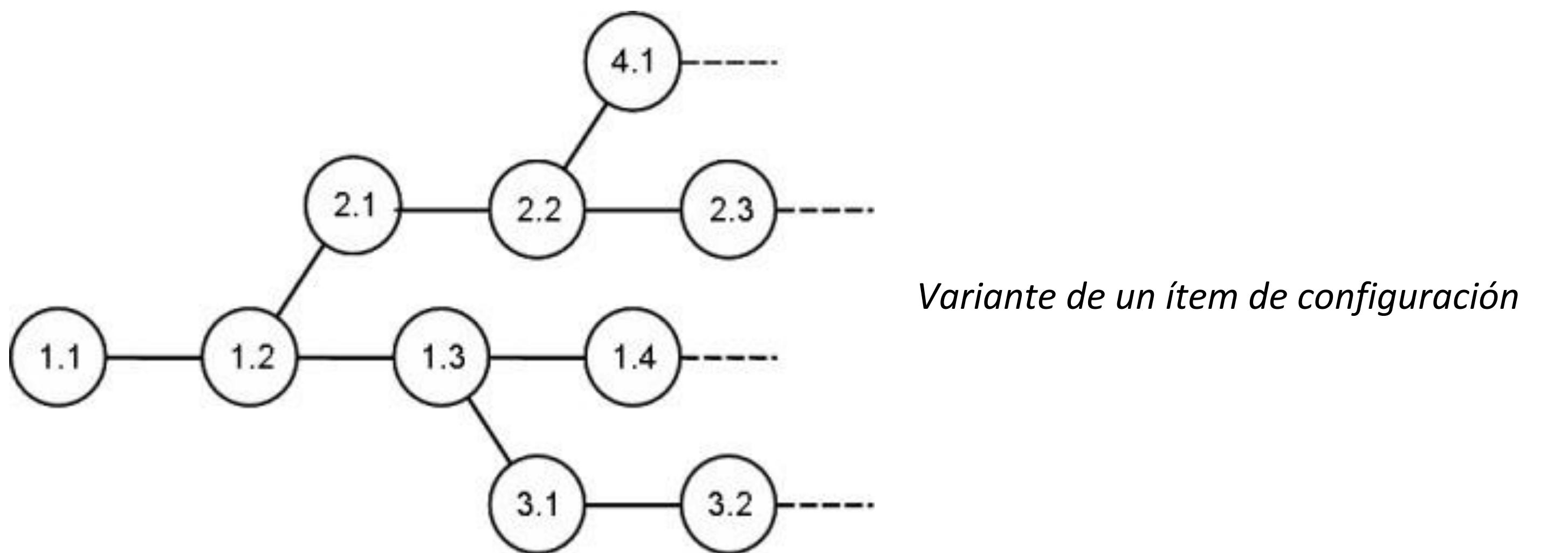
- Una versión se define, desde el punto de vista de la evolución, como la forma particular de un artefacto en un instante o contexto dado.
- El control de versiones se refiere a la evolución de un único ítem de configuración (IC), o de cada IC por separado.
- La evolución puede representarse gráficamente en forma de grafo.



Evolución lineal de un ítem de configuración

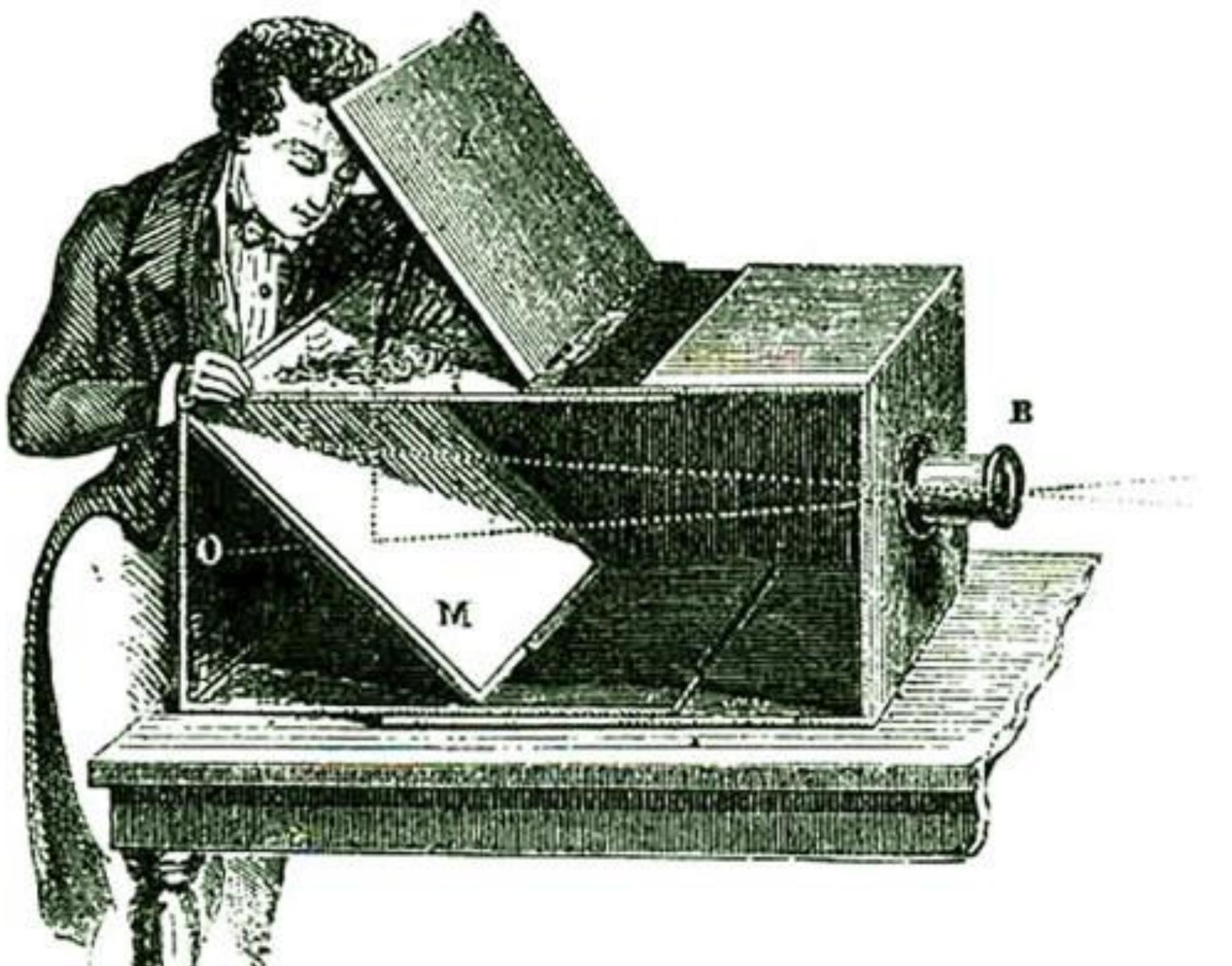
Variante

- ❖ Una variante es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado.
- ❖ Las variantes representan configuraciones alternativas.
- ❖ Un producto de software puede adoptar distintas formas (configuraciones) dependiendo del lugar donde se instale.
- ❖ Por ejemplo, dependiendo de la plataforma (máquina + S.O.) que la soporta, o de las funciones opcionales que haya de realizar o no.



La Configuración del Software

Un conjunto de ítems de configuración con su correspondiente versión en un momento determinado



¿Qué es un Repositorio?



- ❖ Un repositorio de información conteniendo los ítems de configuración (ICs)
- ❖ Mantiene la historia de cada IC con sus atributos y relaciones.
- ❖ Usado para hacer evaluaciones de impacto de los cambios propuestos.
- ❖ Pueden ser una o varias bases de datos

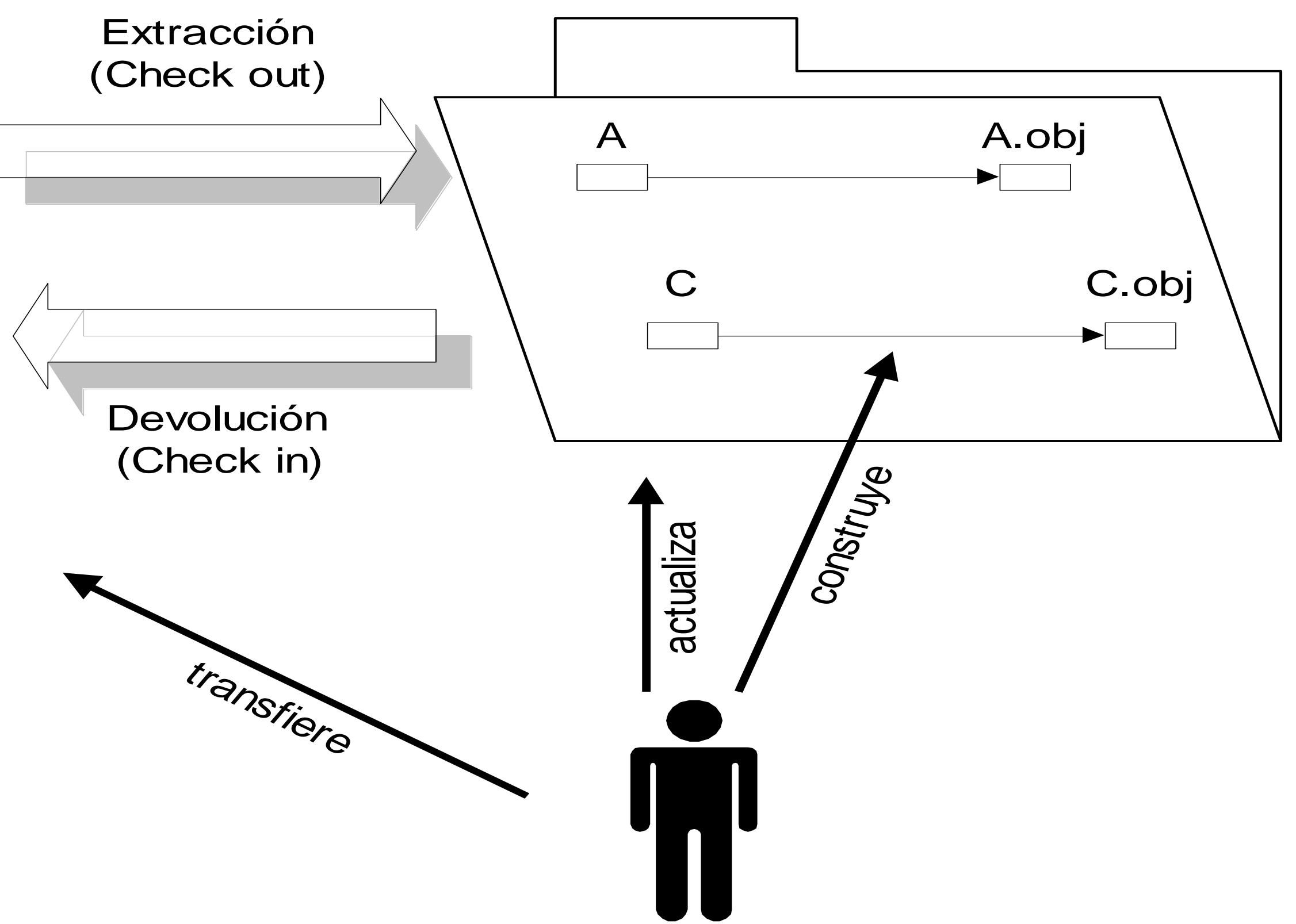
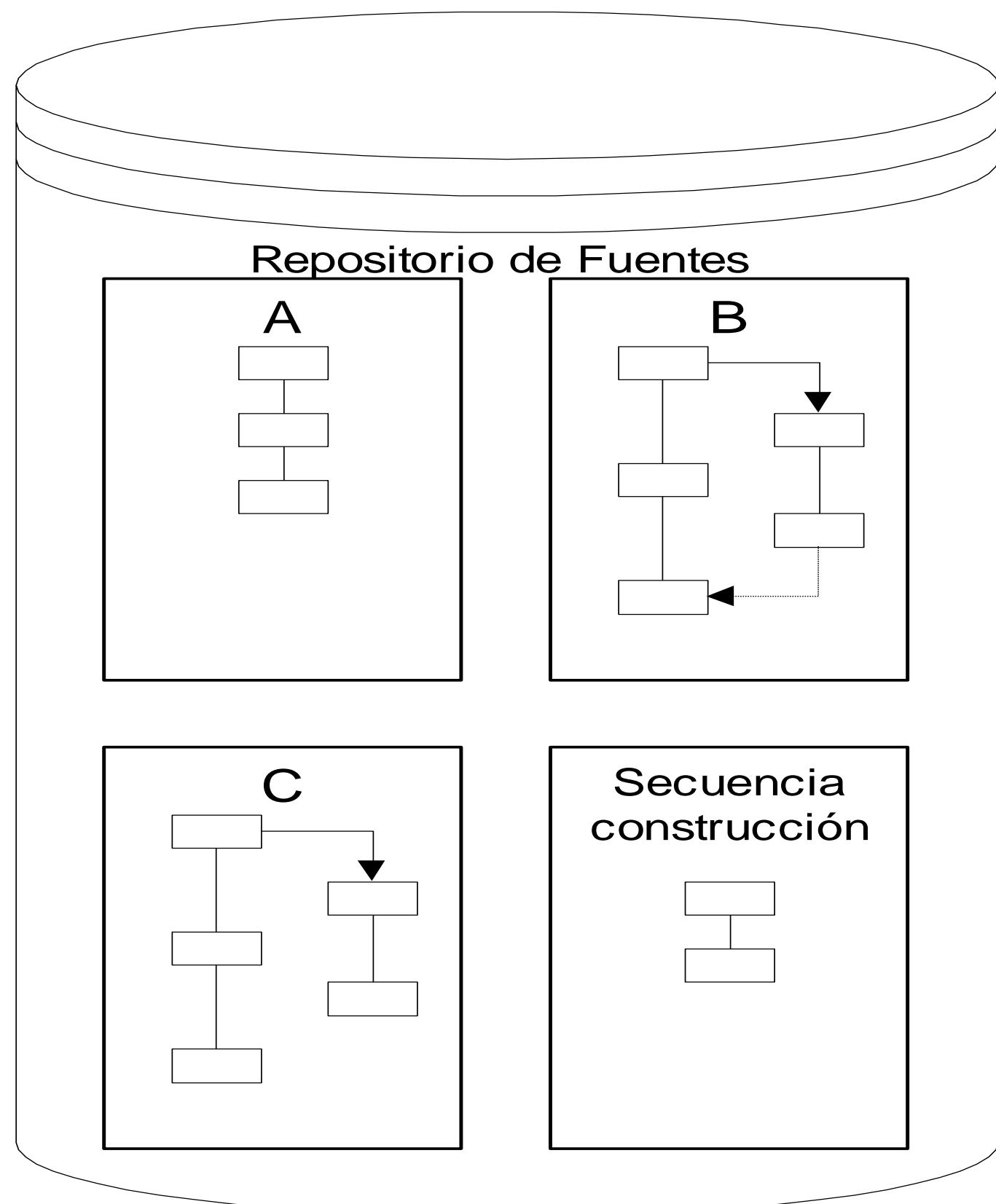


Ejemplo de Repositorio...

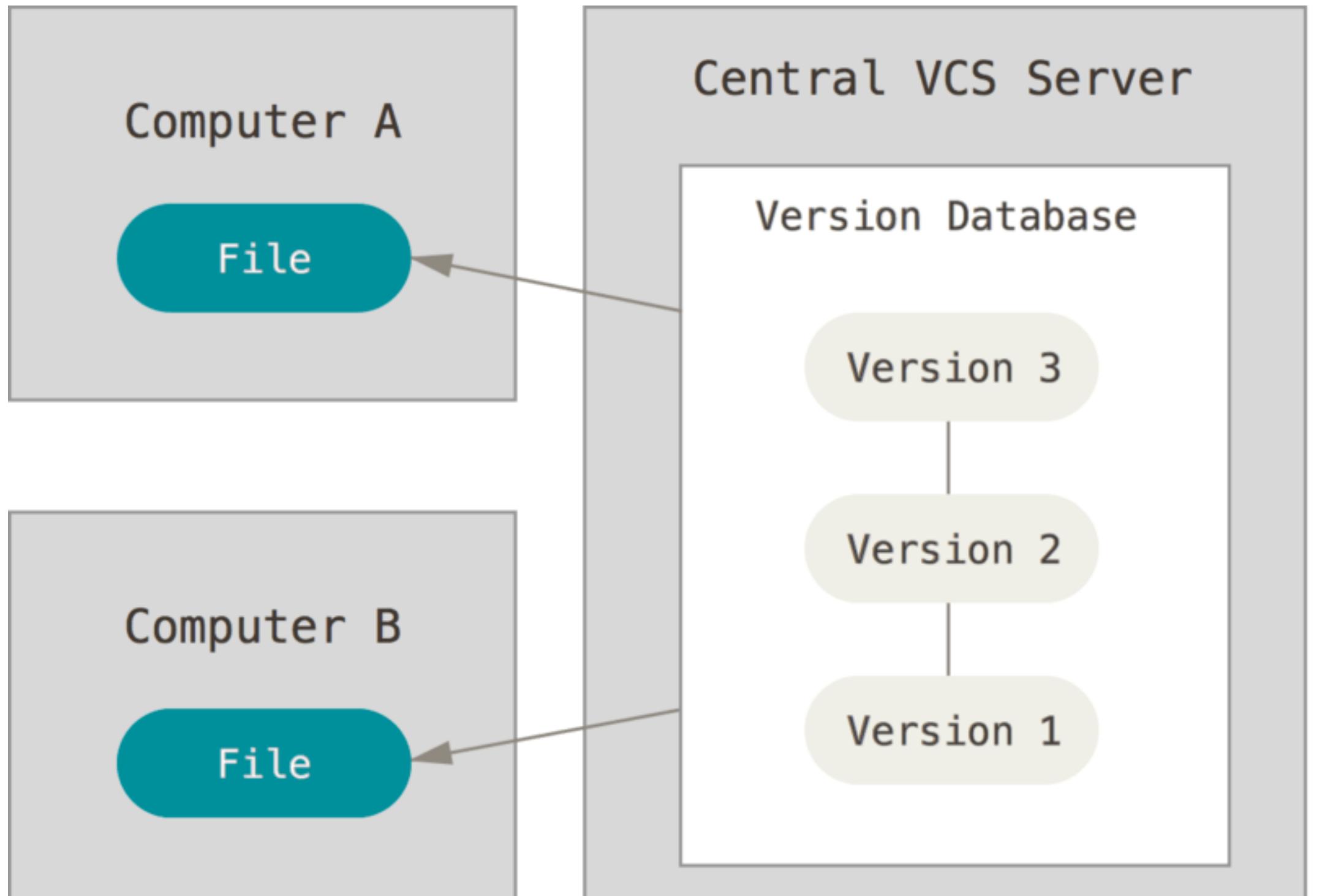
The screenshot shows a Windows file explorer window displaying a directory structure. The path in the address bar is: Judith > Documentos > EJERCICIOS > project21 > Ejercicios >. The left sidebar shows 'Vínculos favoritos' (Documentos, Imágenes, Música) and 'Carpetas' (Personal Trainer, project21, .svn, 2008, 2009, 2010, Apuntes y Material, Ejercicios). The main pane lists files and folders with columns for Nombre, Fecha modificación, Tipo, Tamaño, and Etiquetas. The 'Ejercicios' folder is selected. The taskbar at the bottom shows several open applications: Ejercicios, Bandeja de entrada ..., Curso de SCM(1) [M...], R&D SCM 05 2010 I..., R&D SCM 05 2010, and R&D SCM 05 2010. The system tray shows icons for network, battery, volume, and time (11:08 a.m.).

Nombre	Fecha modificación	Tipo
.svn	27/05/2010 12:24 a...	Carpeta de archivos
Complejo de Cine	26/04/2010 10:48 ...	Carpeta de archivos
Escuela Musical	20/04/2010 12:22 a...	Carpeta de archivos
Estación de Servicios	27/05/2010 12:24 a...	Carpeta de archivos
Festival de Folklore	11/05/2010 01:02 a...	Carpeta de archivos
Guardia Documental	27/05/2010 12:22 a...	Carpeta de archivos
Liga de Fútbol	04/05/2010 04:27 a...	Carpeta de archivos
Medidores de Agua	27/05/2010 12:23 a...	Carpeta de archivos
Notificación Compras	27/05/2010 12:23 a...	Carpeta de archivos
Seguimiento de Defecto...	27/03/2010 02:02 ...	Carpeta de archivos
Simulador de Compras	27/05/2010 12:27 a...	Carpeta de archivos
Solicitud de Préstamo	27/05/2010 12:23 a...	Carpeta de archivos
Tours	03/05/2010 10:29 ...	Carpeta de archivos
Venta Digital	14/03/2010 07:26 ...	Carpeta de archivos
Voto Electrónico	20/04/2010 12:23 a...	Carpeta de archivos

Funcionamiento del Repositorio

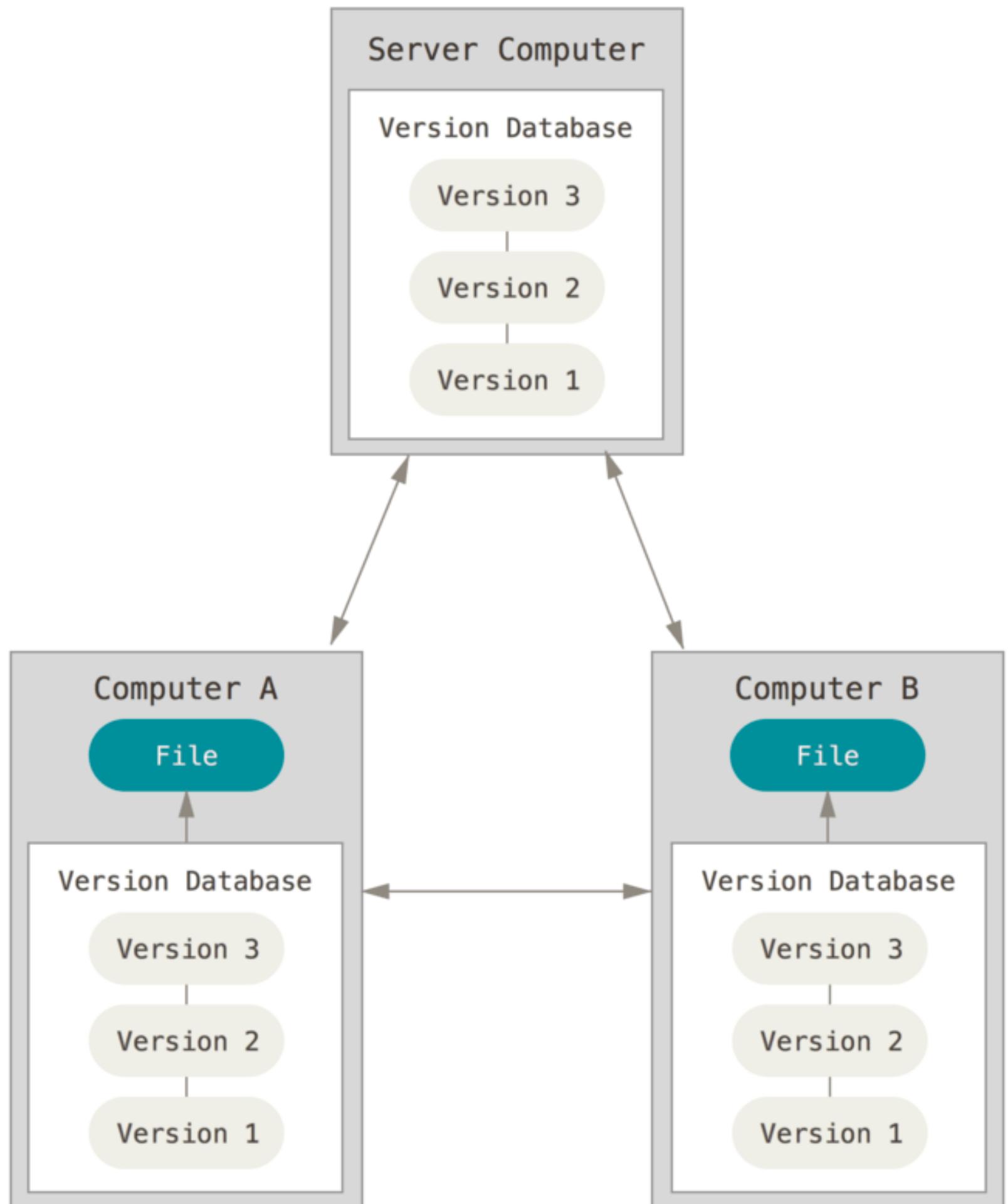


Repositorios Centralizados



- ❖ Un servidor contiene todos los archivos con sus versiones.
- ❖ Los administradores tiene mayor control sobre el repositorio.
- ❖ Falla el servidor y "estamos al horno".

Repositorios Descentralizados



- ❖ Cada cliente tiene una copia **exactamente igual** del repositorio completo.
- ❖ Si un servidor falla sólo es cuestión de “copiar y pegar”.
- ❖ Posibilita otros workflows no disponibles en el modelo centralizado.

Identificación de la Línea Base

- ❖ Se utilizan etiquetas para “marcar” las baseline
- ❖ No confundir con la versión del Producto

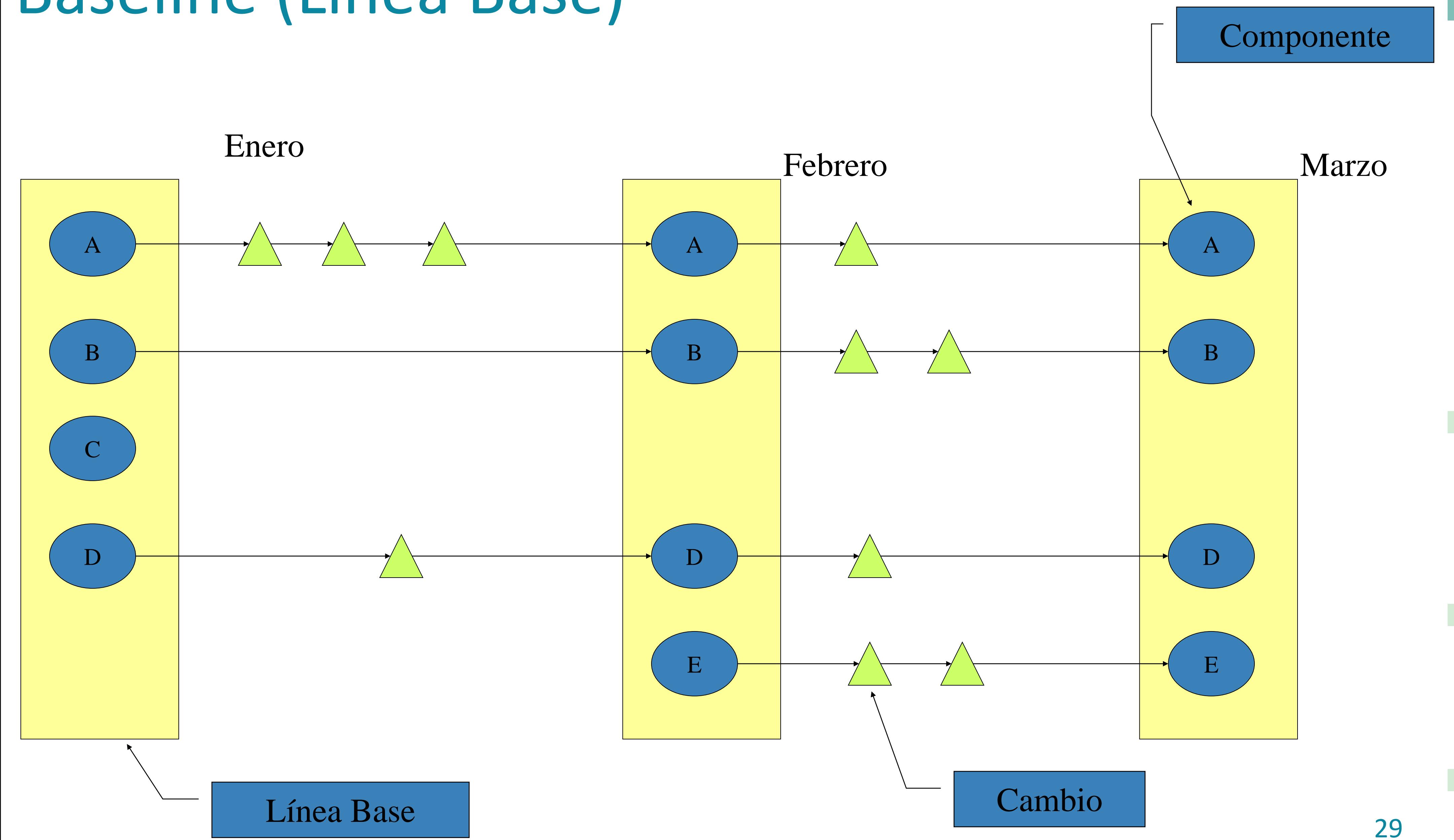


Líneas Base



- ❖ Una configuración que ha sido revisada formalmente y sobre la que se ha llegado a un acuerdo
- ❖ Sirve como base para desarrollos posteriores y puede cambiarse sólo a través de un procedimiento formal de control de cambios
- ❖ Permiten ir atrás en el tiempo y reproducir el entorno de desarrollo en un momento dado del proyecto

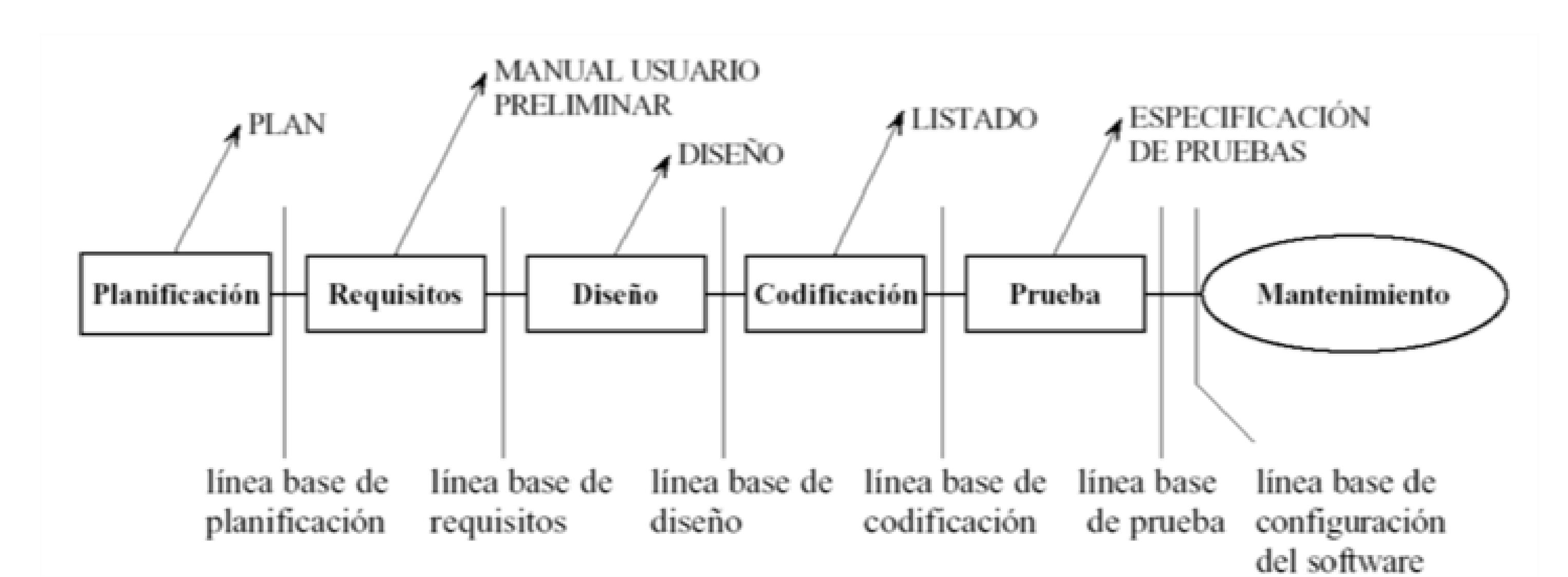
Baseline (Línea Base)



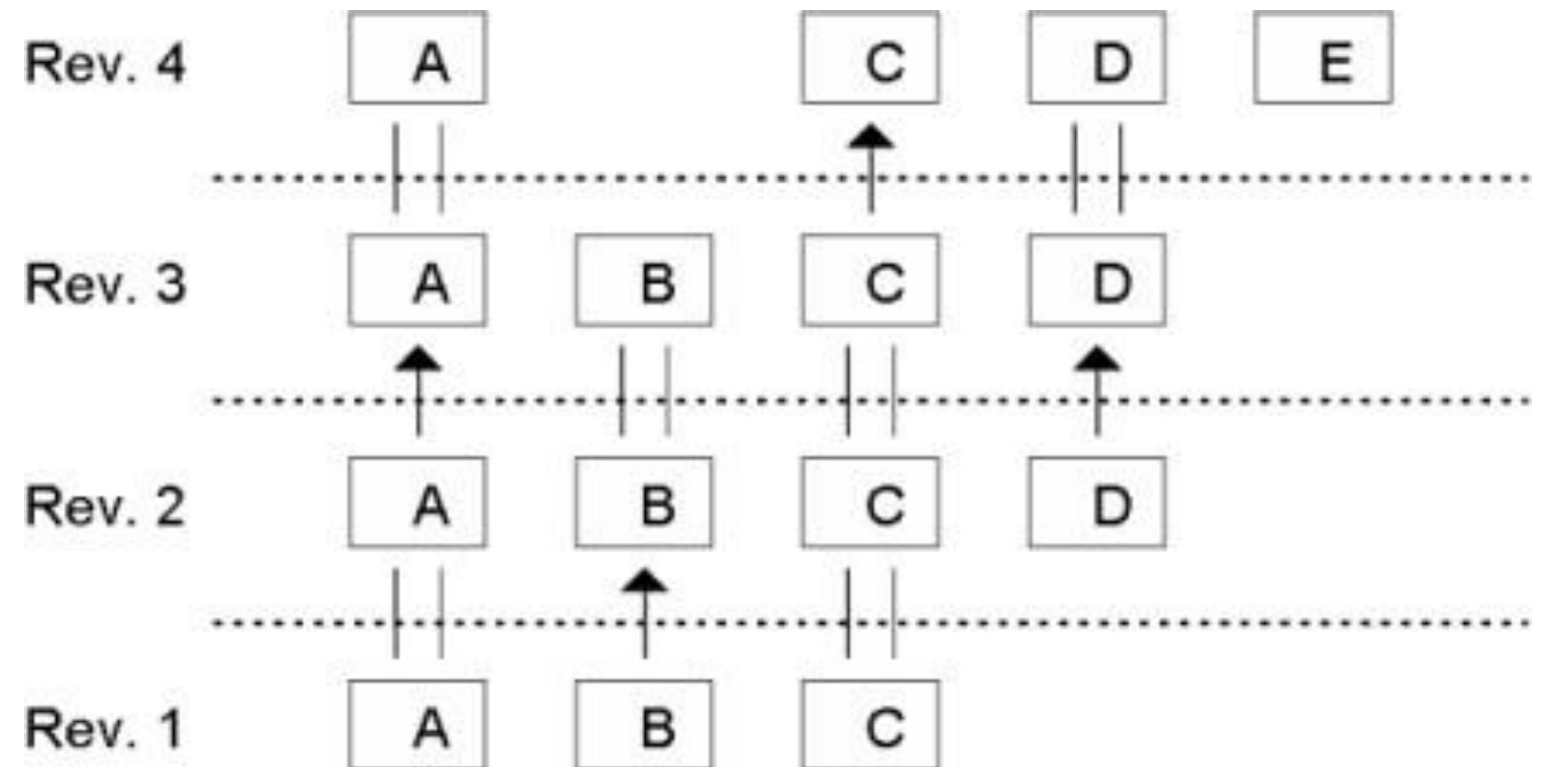
Representación de Líneas Base

Pueden ser:

- De especificación (Requerimientos, Diseño)
- De productos que han pasado por un control de calidad definido previamente

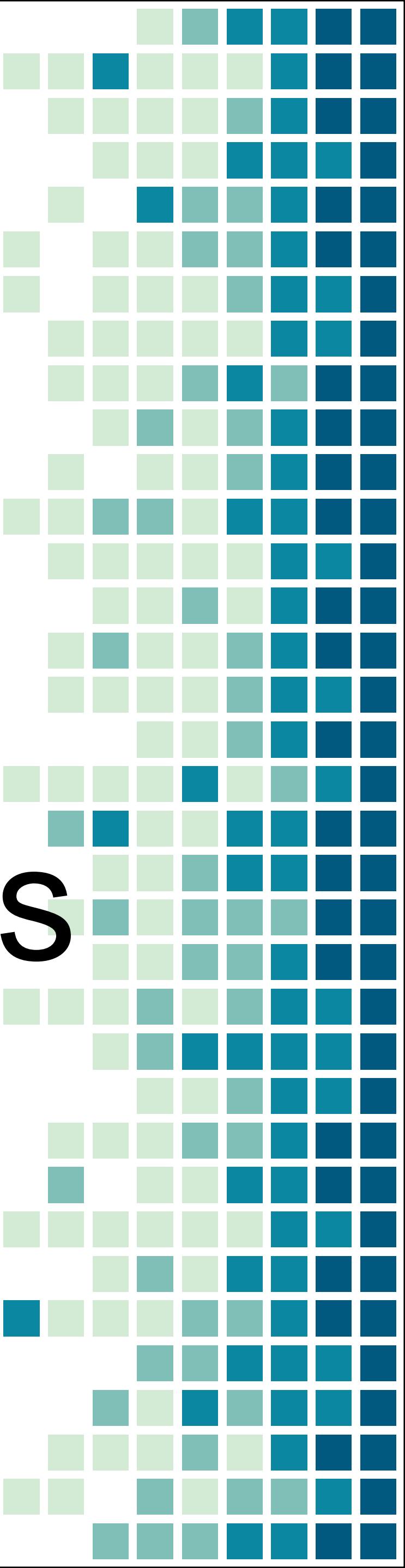


Evolución de una configuración



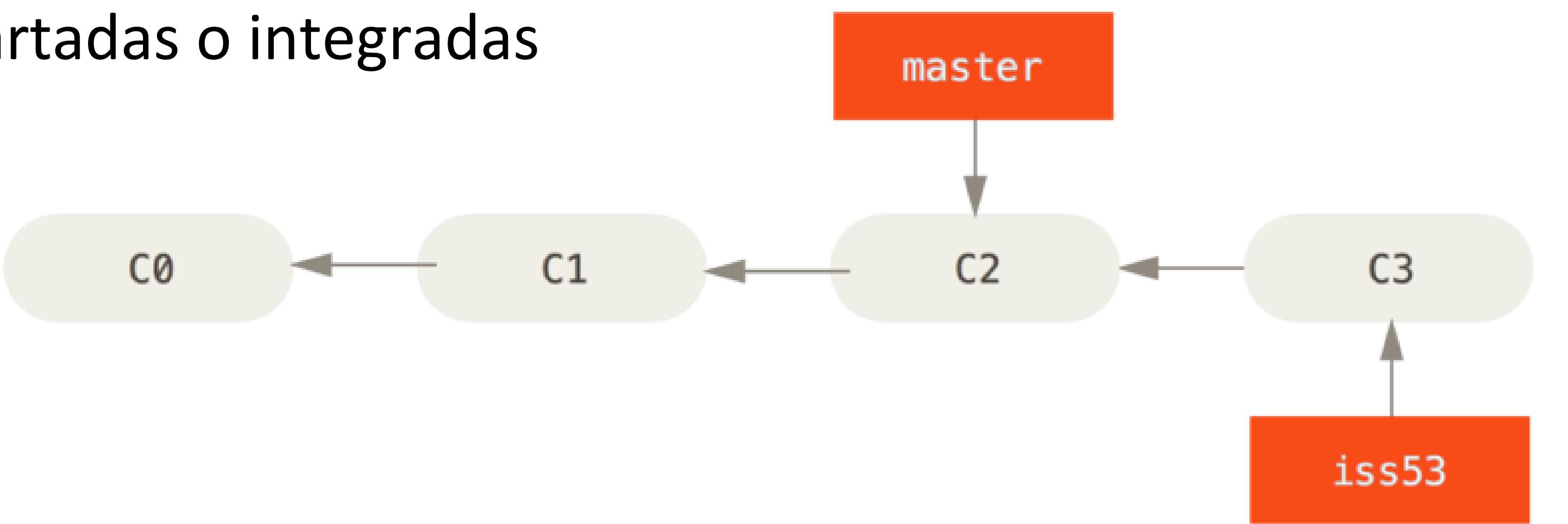


Ramas



Creación de ramas

- ❖ Existe una rama principal (trunk, master)
- ❖ Sirven para bifurcar el desarrollo
- ❖ Pueden tener razones de creación con semántica
- ❖ Permiten la experimentación
- ❖ Pueden ser descartadas o integradas



Integración de ramas



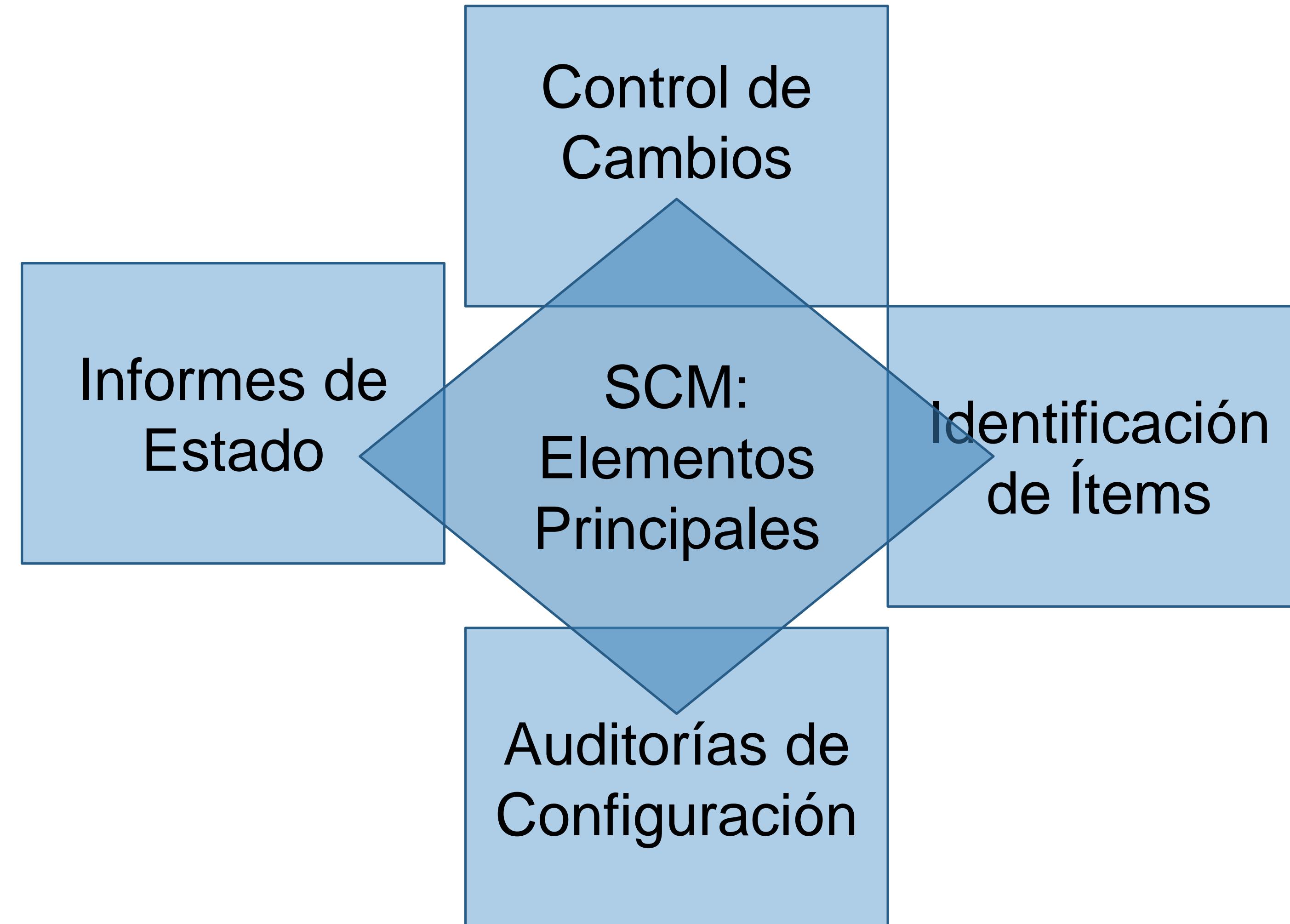
- ❖ La operación se llama merge
- ❖ Lleva los cambios a la rama principal
- ❖ Pueden surgir conflictos (resolvemos con diff)
- ❖ Todas las ramas deberían eventualmente integrarse a la principal o ser descartadas

Definición

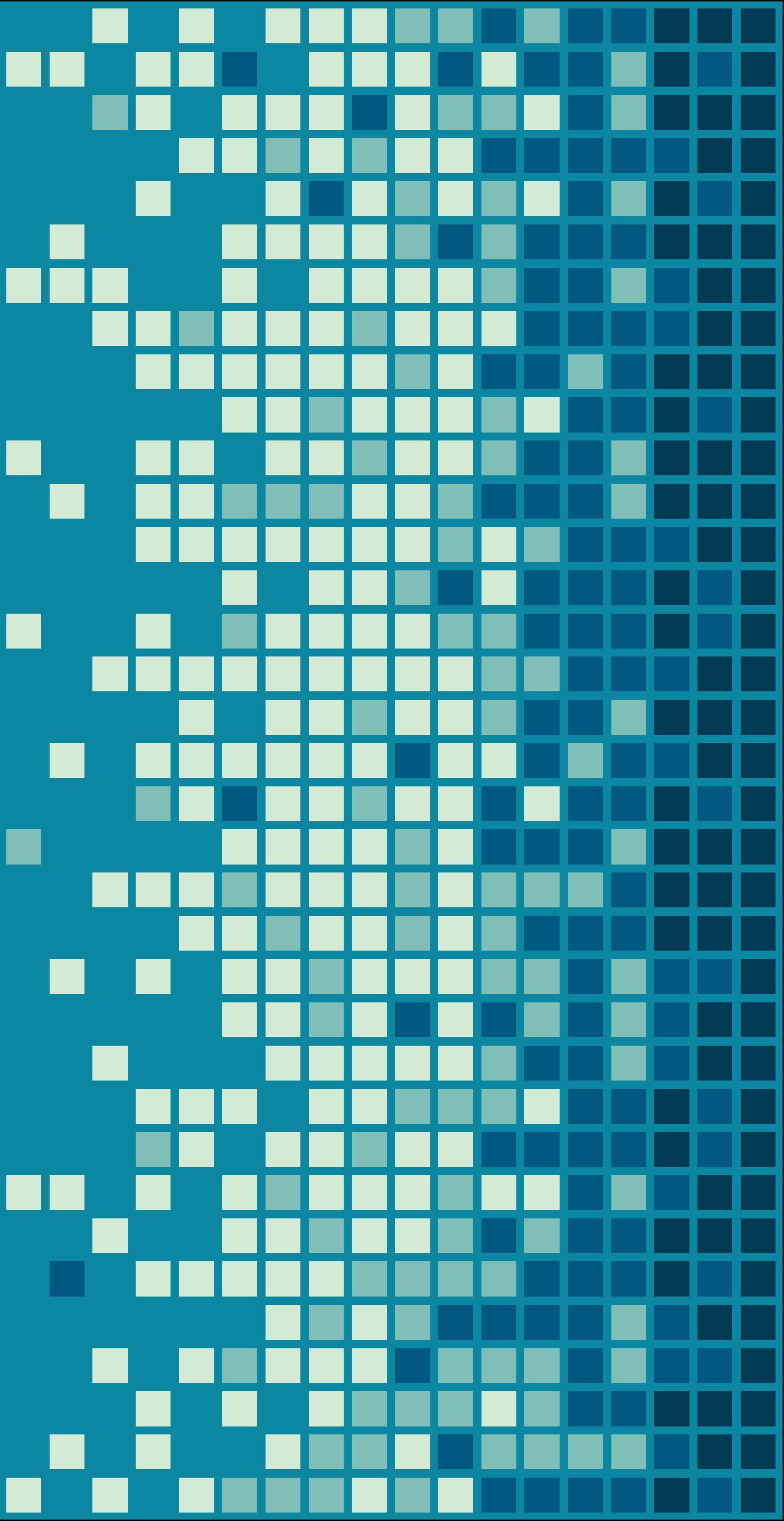
Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos

(ANSI/IEEE 828, 1990)

Actividades Fundamentales de la Administración de Configuración de Software



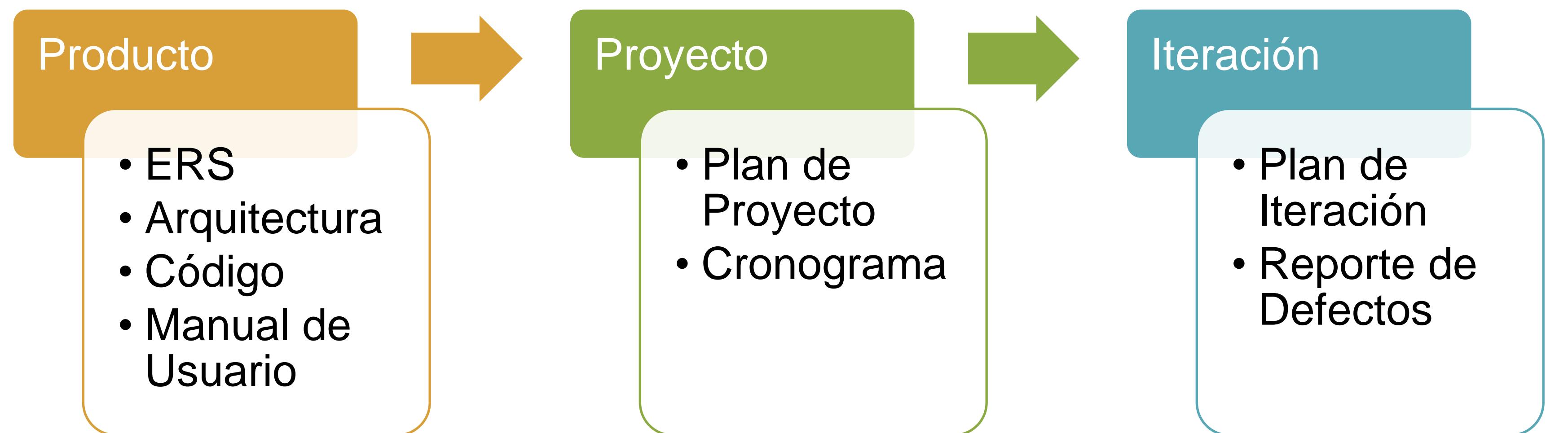
II. Identificación de ítems de configuración



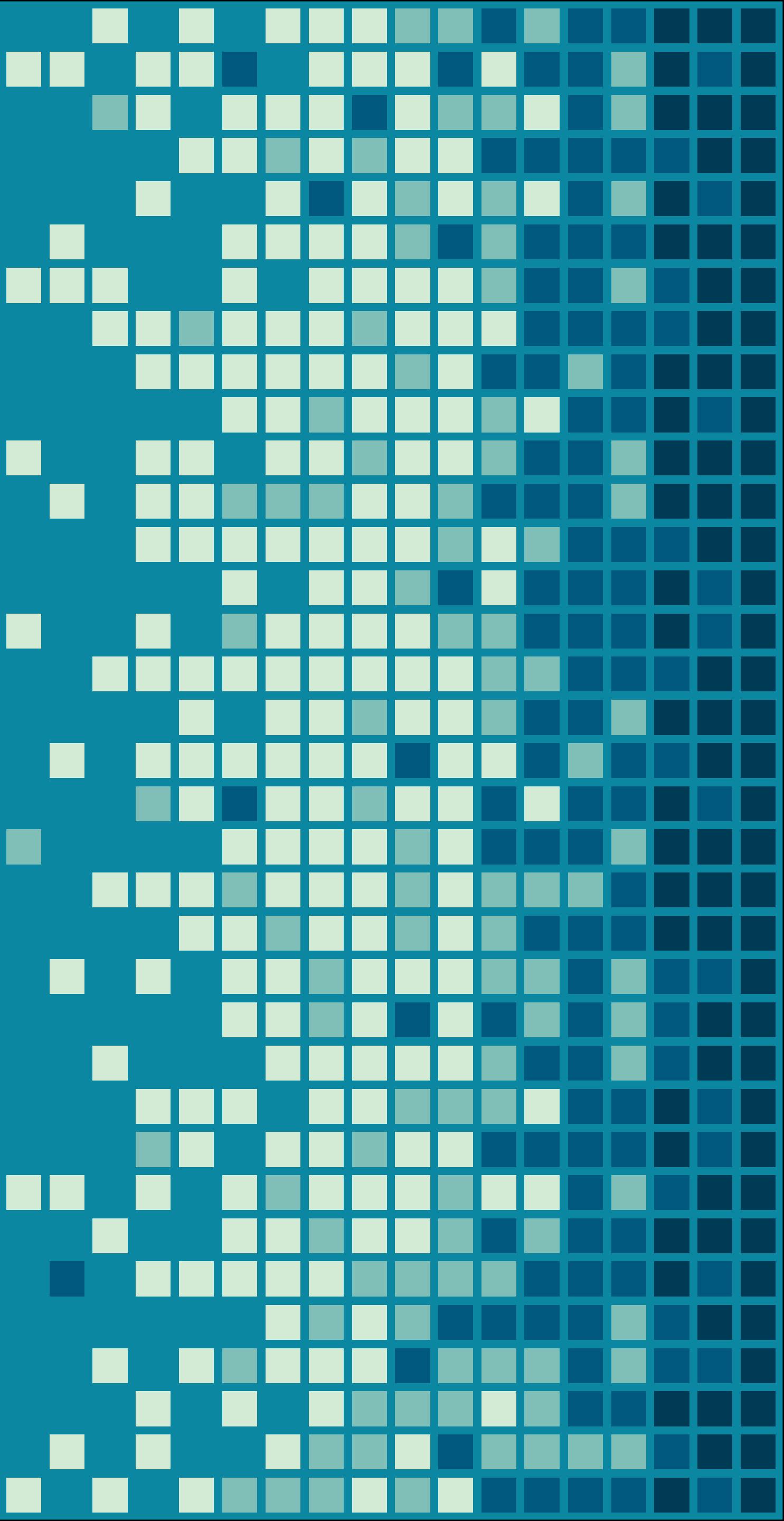
Identificación de ítems de configuración

- ❖ Identificación unívoca de cada ítem de configuración
- ❖ Convenciones y reglas de nombrado
- ❖ Definición de la Estructura del Repositorio
- ❖ Ubicación dentro de la estructura del repositorio

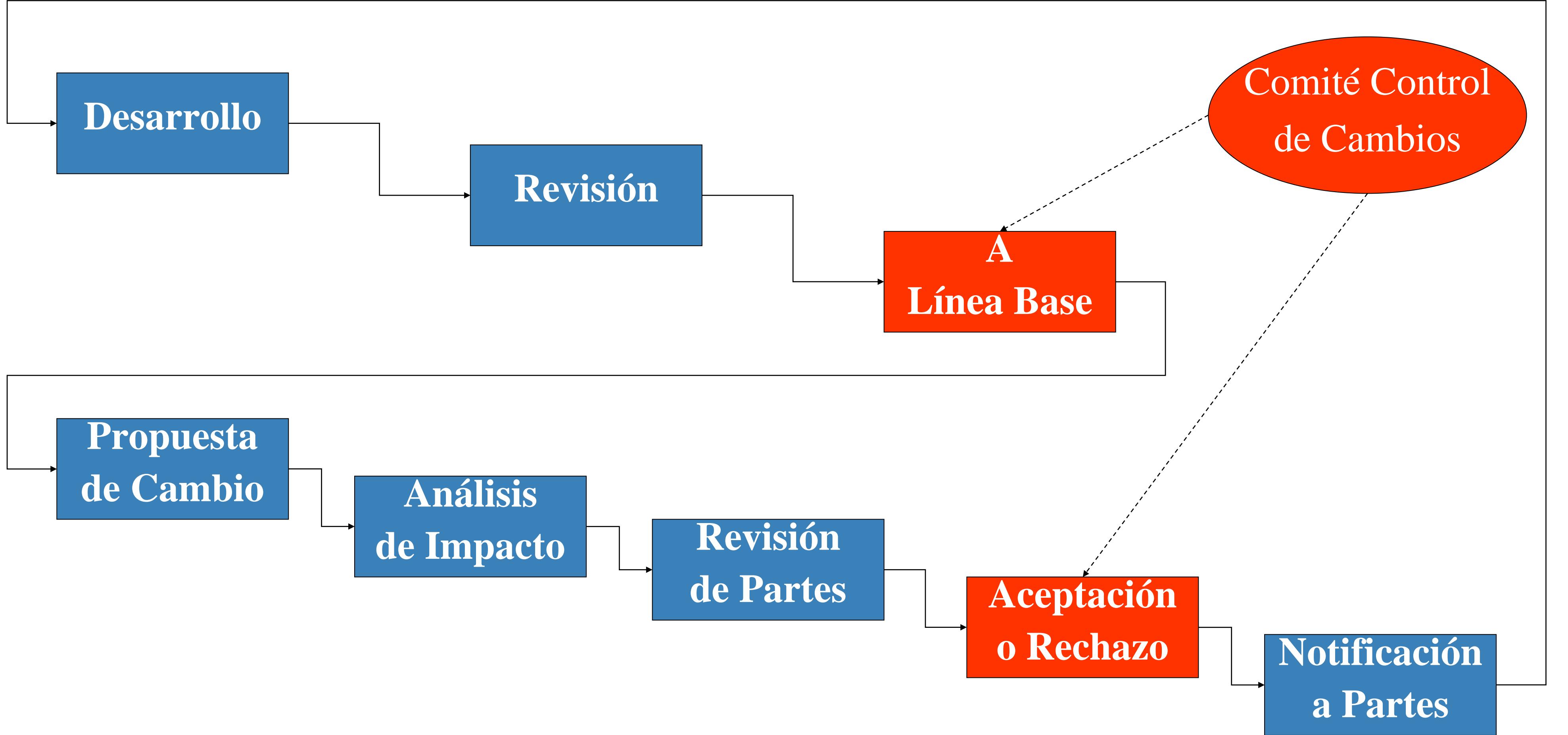
Ítems de Configuración para un proyecto de desarrollo de software



“. Control de Cambios

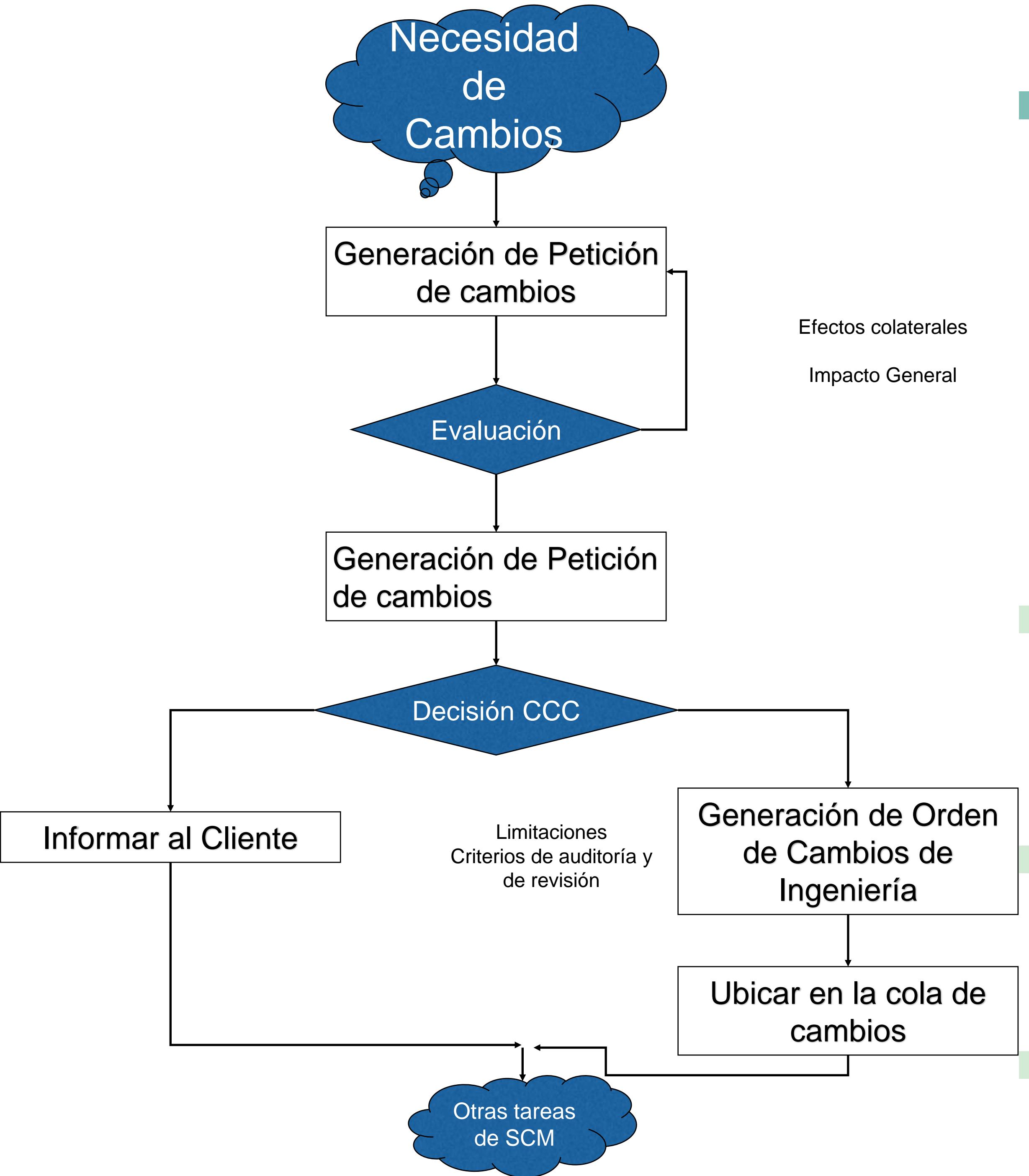


Proceso de Control de Cambios

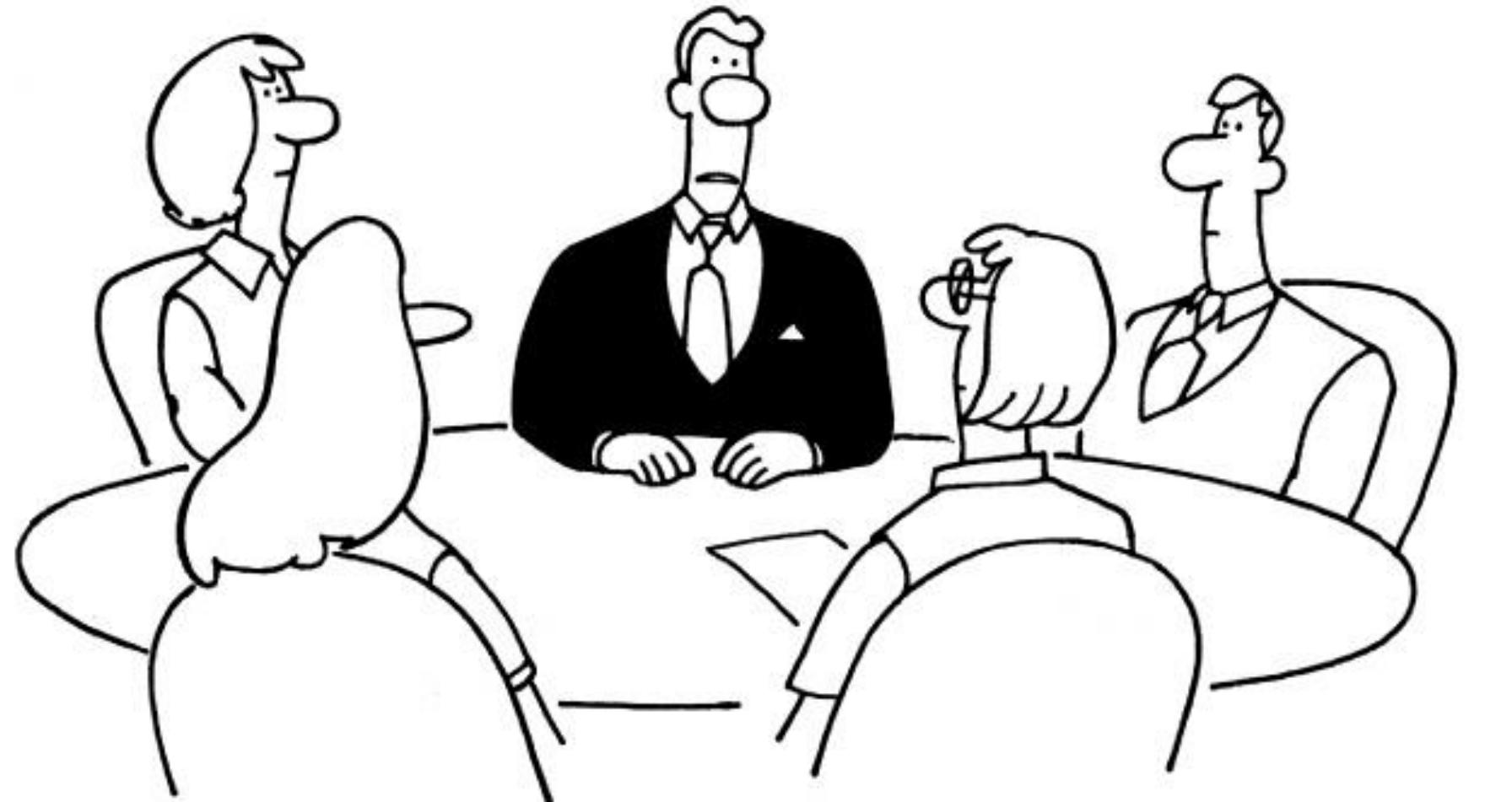


Control de Cambios

- ❖ Tiene su origen en un Requerimiento de Cambio a uno o varios ítems de configuración que se encuentran en una **línea base**.
- ❖ Es un Procedimiento formal que involucra diferentes actores y una evaluación del **impacto** del cambio



El Comité de Control de Cambios

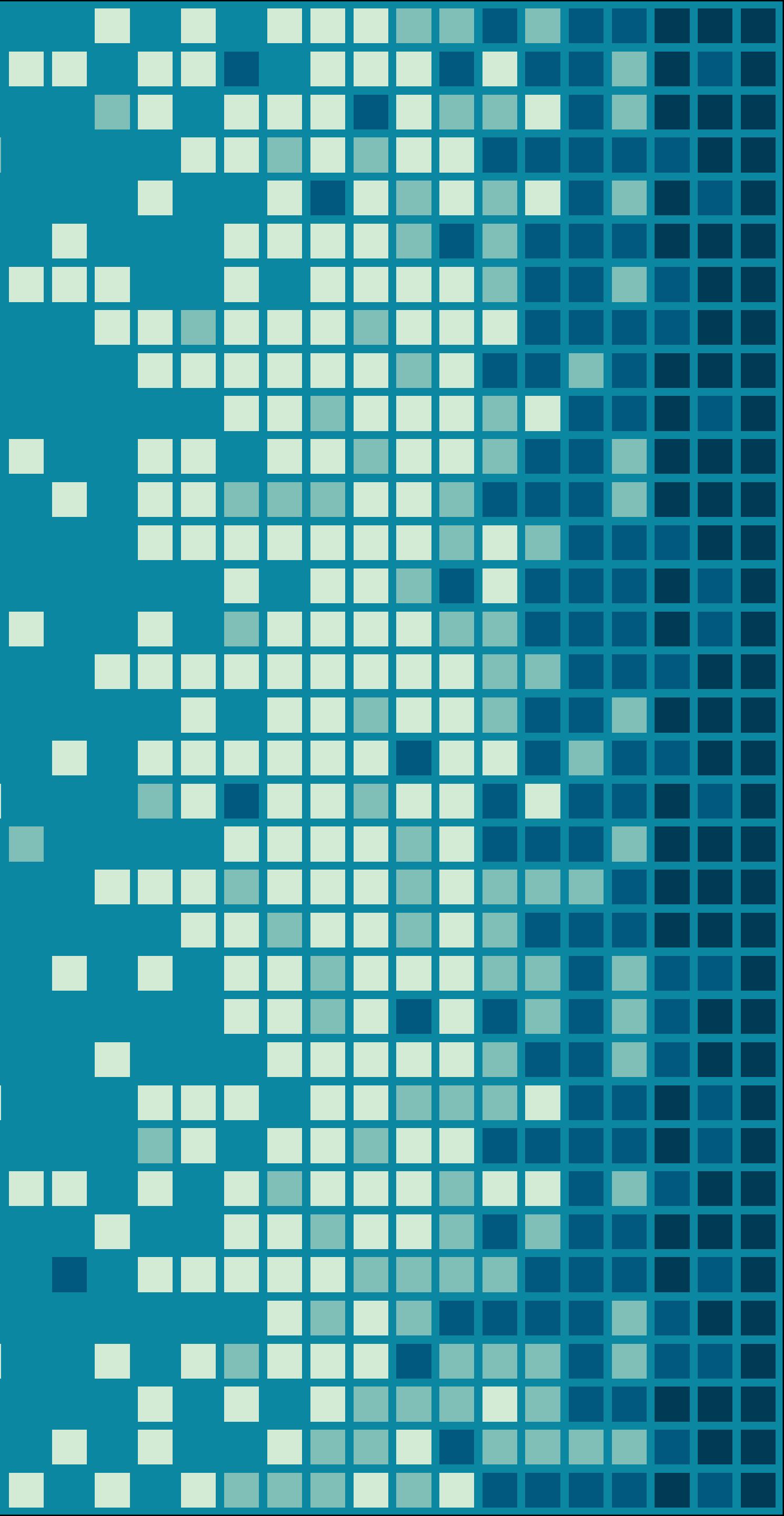


*"Whew! That was close!
We almost decided something!"*

Está formado por representantes de todas las áreas involucradas en el desarrollo:

- ❖ Análisis, Diseño
- ❖ Implementación
- ❖ Testing
- ❖ Otros interesados

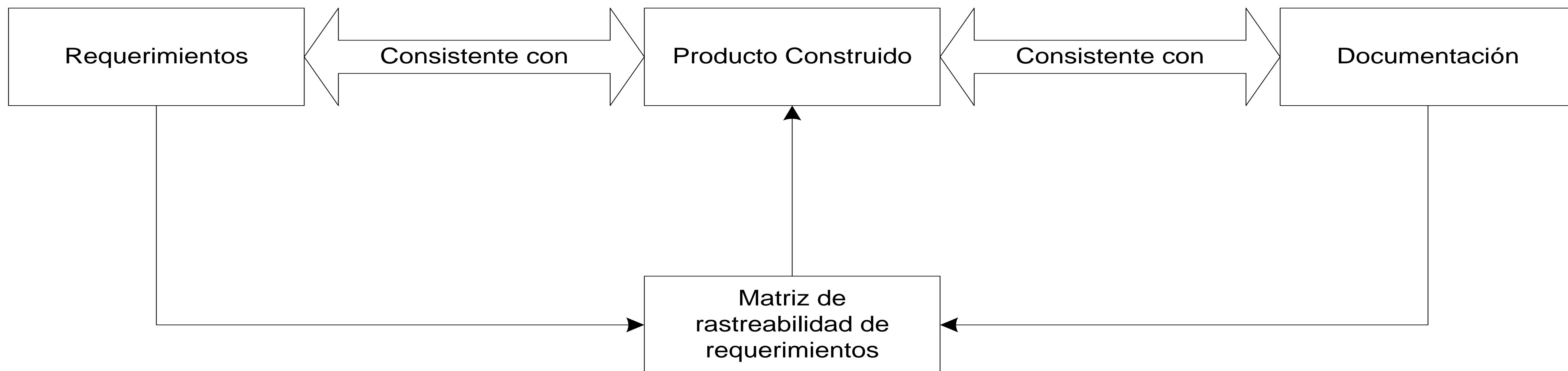
“ Auditorías de Configuración de Software



Auditoría de Gestión de Configuración

Auditoría Funcional de Configuración

Auditoría Física de Configuración



Auditorías de Configuración

- ❖ **Auditoría física de configuración (PCA)**

Asegura que lo que está indicado para cada ICS en la línea base o en la actualización se ha alcanzado realmente.

- ❖ **Auditoría funcional de configuración (FCA)**

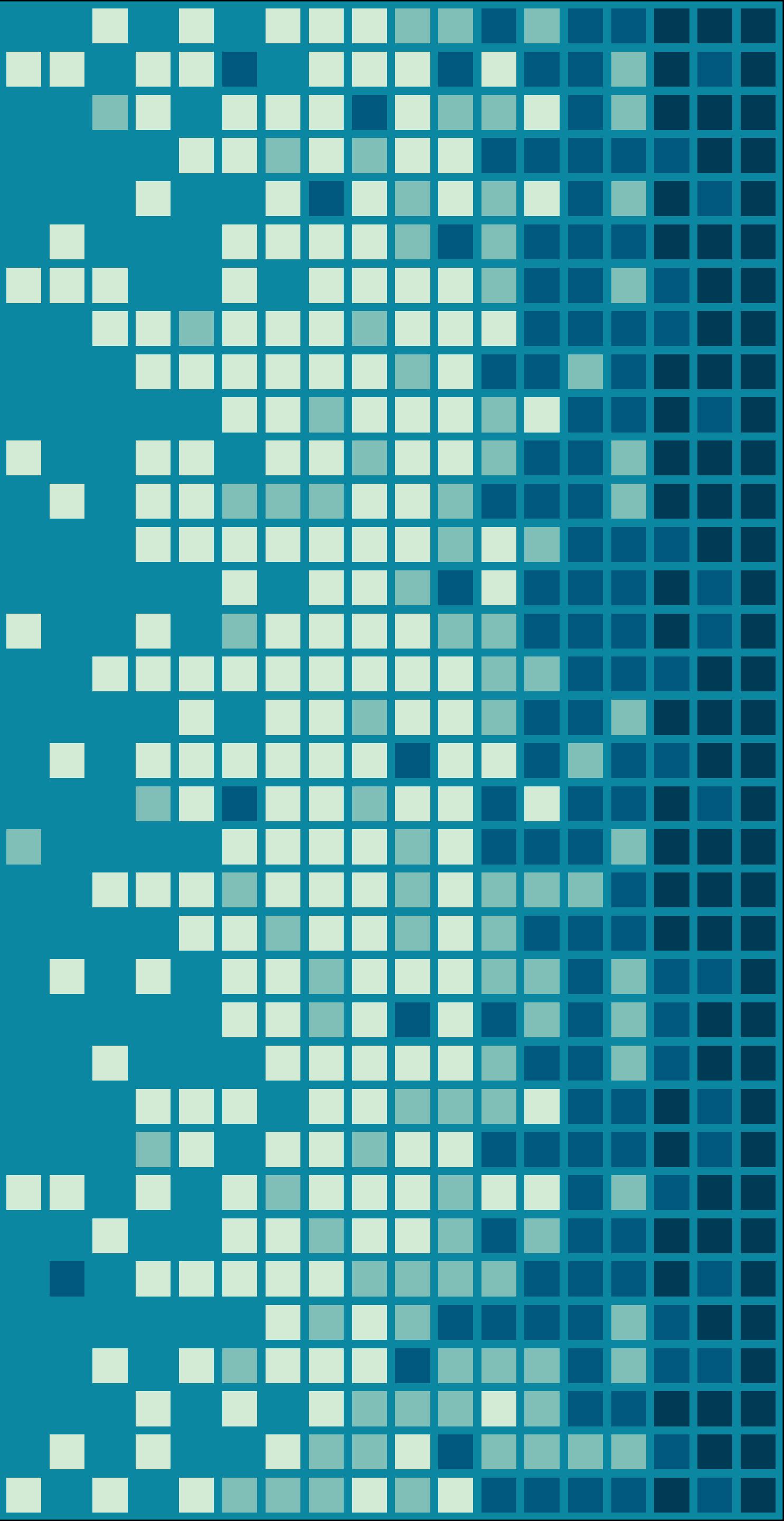
Evaluación independiente de los productos de software, controlando que la funcionalidad y performance reales de cada ítem de configuración sean consistentes con la especificación de requerimientos.

Auditoría de Gestión de Configuración y V&V

Sirve a dos procesos básicos: la validación y la verificación

- ❖ **Validación:** el problema es resuelto de manera apropiada que el usuario obtenga el producto correcto.
- ❖ **Verificación:** asegura que un producto cumple con los objetivos preestablecidos, definidos en la documentación de líneas base (línea base). Todas las funciones son llevadas a cabo con éxito y los test cases tengan status “ok” o bien consten como “problemas reportados” en la nota de release.

II. Informes de Estado



Registro e Informe de Estado

- ❖ Se ocupa de mantener los registros de la evolución del sistema.
- ❖ Maneja mucha información y salidas por lo que se suele implementar dentro de procesos automáticos.
- ❖ Incluye reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida.

Algunas preguntas que podría responder

- ❖ ¿Cuál es el estado del ítem?
- ❖ ¿Un requerimiento de cambio ha sido aprobado o rechazado por el CCB?
- ❖ ¿Qué versión de ítem implementa un requerimiento de cambio aprobado (saber cuál es el componente que contiene la mejora)?
- ❖ ¿Cuál es la diferencia entre una versión y otra dada?

HORA DE UN PLAN!

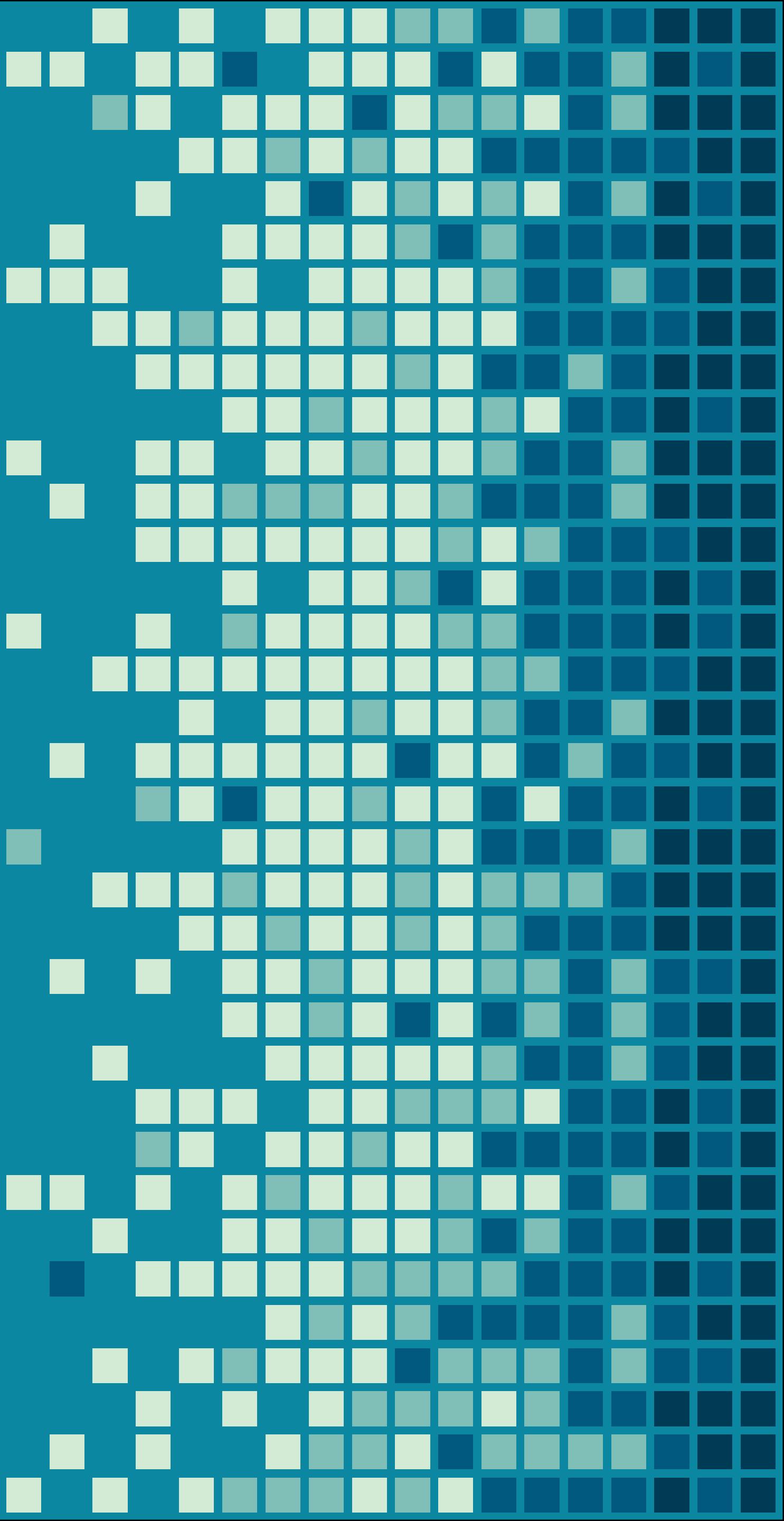


Plan de Gestión de Configuración

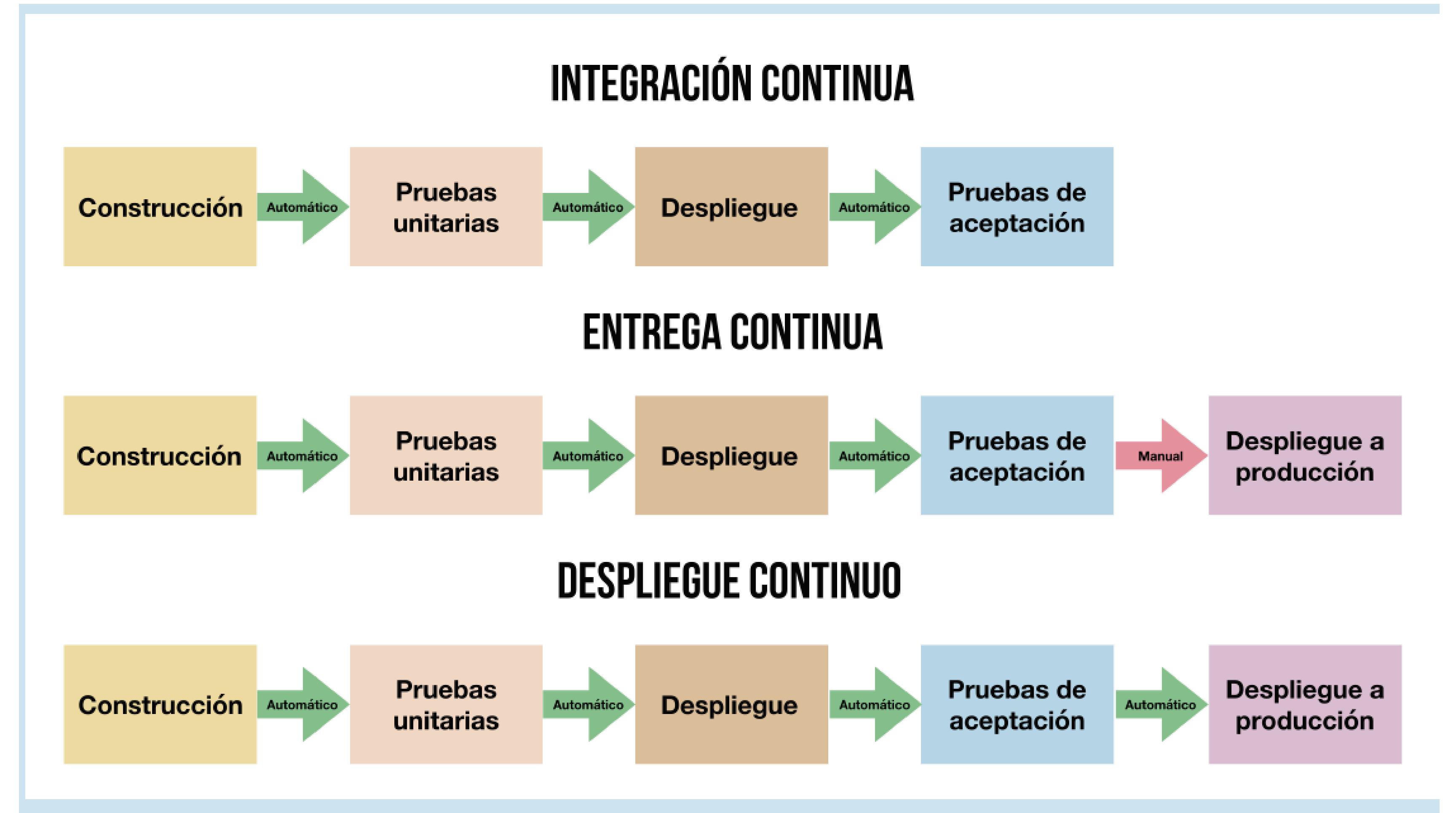
También se planifica! Qué debería incluir el plan?

- ❖ Reglas de nombrado de los CI
- ❖ Herramientas a utilizar para SCM
- ❖ Roles e integrantes del Comité
- ❖ Procedimiento formal de cambios
- ❖ Plantillas de formularios
- ❖ Procesos de Auditoría

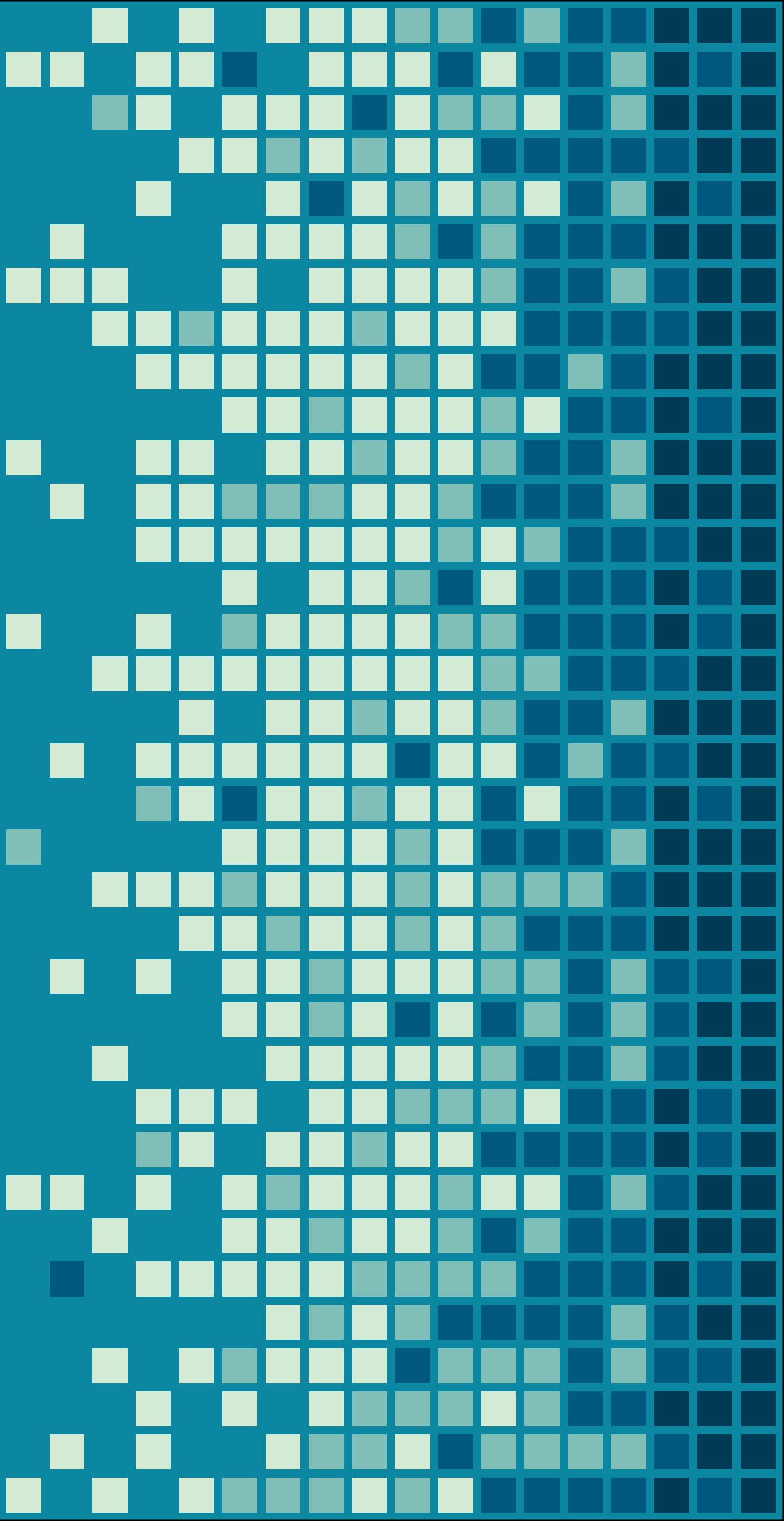
II. Evolución de la Gestión de Configuración de Software



Integración, Entrega y Despliegue Continuos



II. Gestión de Configuración de Software en ambientes Ágiles



Recuerdan... Manifiesto Ágil



SCM en Agile

- ❖ Sirve a los practicantes (equipo de desarrollo) y no viceversa.
- ❖ Hace seguimiento y coordina el desarrollo en lugar de controlar a los desarrolladores.
- ❖ Responde a los cambios en lugar de tratar de evitarlos.
- ❖ Esforzarse por ser transparente y "sin fricción", automatizando tanto como sea posible.
- ❖ Coordinación y automatización frecuente y rápida.
- ❖ Eliminar el desperdicio - no agregar nada más que valor.
- ❖ Documentación Lean y Trazabilidad.
- ❖ Feedback continuo y visible sobre calidad, estabilidad e integridad

SCM en Agile, algunos tips....

- ❖ Es responsabilidad de todo el equipo.
- ❖ Automatizar lo más posible.
- ❖ Educar al equipo.
- ❖ Tareas de SCM embebidas en las demás tareas requeridas para alcanzar el objetivo del Sprint.

SCM en Agile, para debatir....

- ❖ ¿Qué pasa con el Comité de Control de Cambios?
- ❖ ¿Qué ítems de configuración podemos tener?
- ❖ ¿Qué pasa con las auditorías?
- ❖ ¿Qué pasa con los reportes de estado?

Referencias

- Bersoff, E.H., “Elements of Software Configuration Management”,
IEEE Transactions on Software Engineering, vol 10, nro. 1, enero 1984, pp 79-87
- Little Book of Configuration Management – <http://www.spmn.com>
- SCM & the Agile Manifesto - <http://www.scmpatterns.com/agilescm/>

Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra de Ingeniería de Software
Docentes: Judith Meles - Laura Covaro



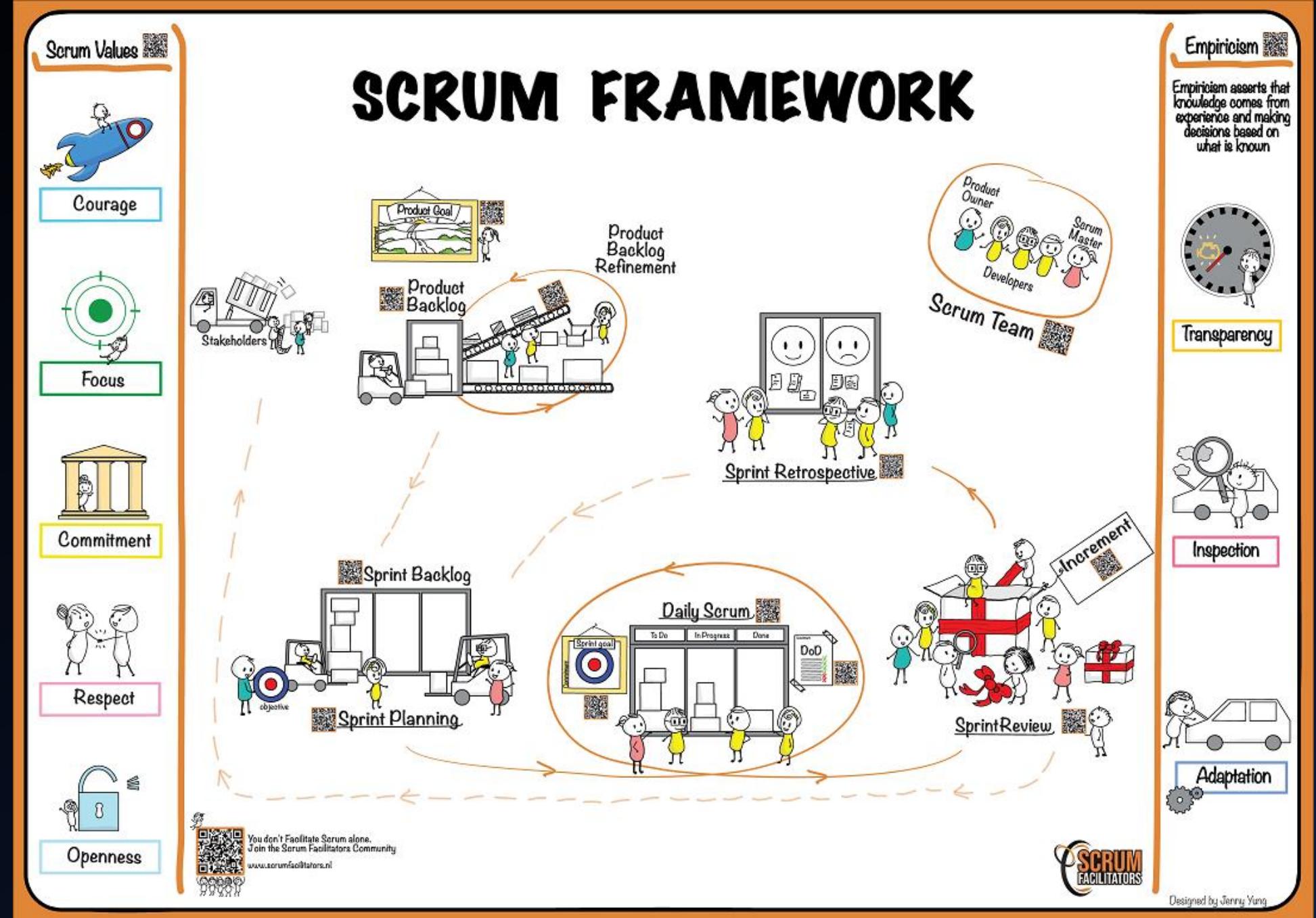
JUDITH MELES

SCRUM 2020

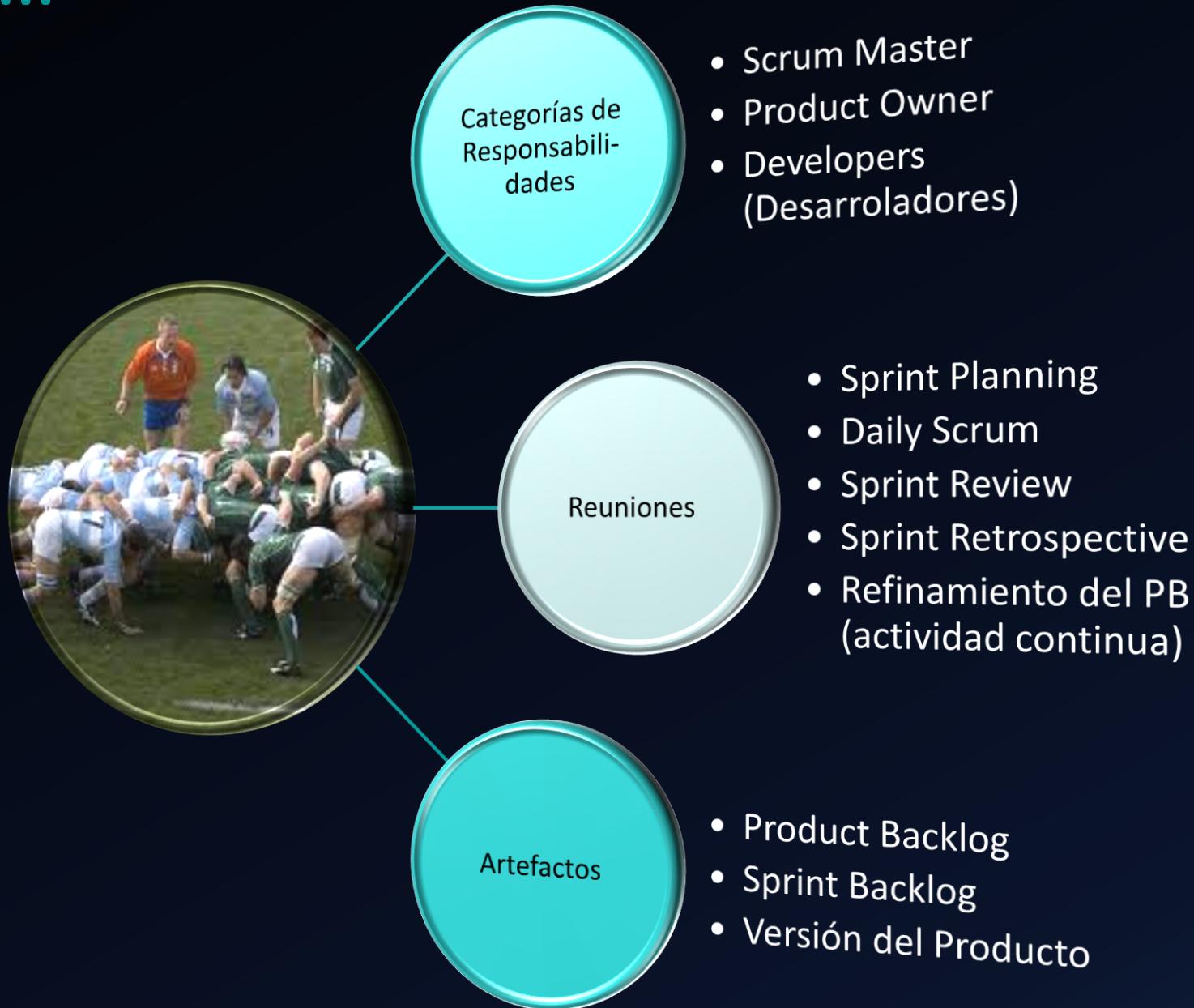
Scrum: El framework 2020

Scrum es un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos.

EL FRAMEWORK SCRUM 2020...



Resumiendo...



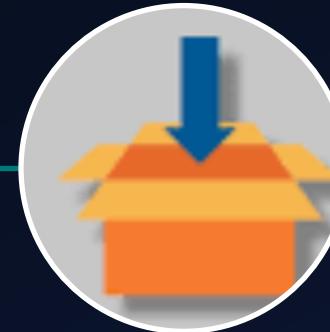
Artefactos de Scrum



Product Backlog
→ Objetivo del
Producto

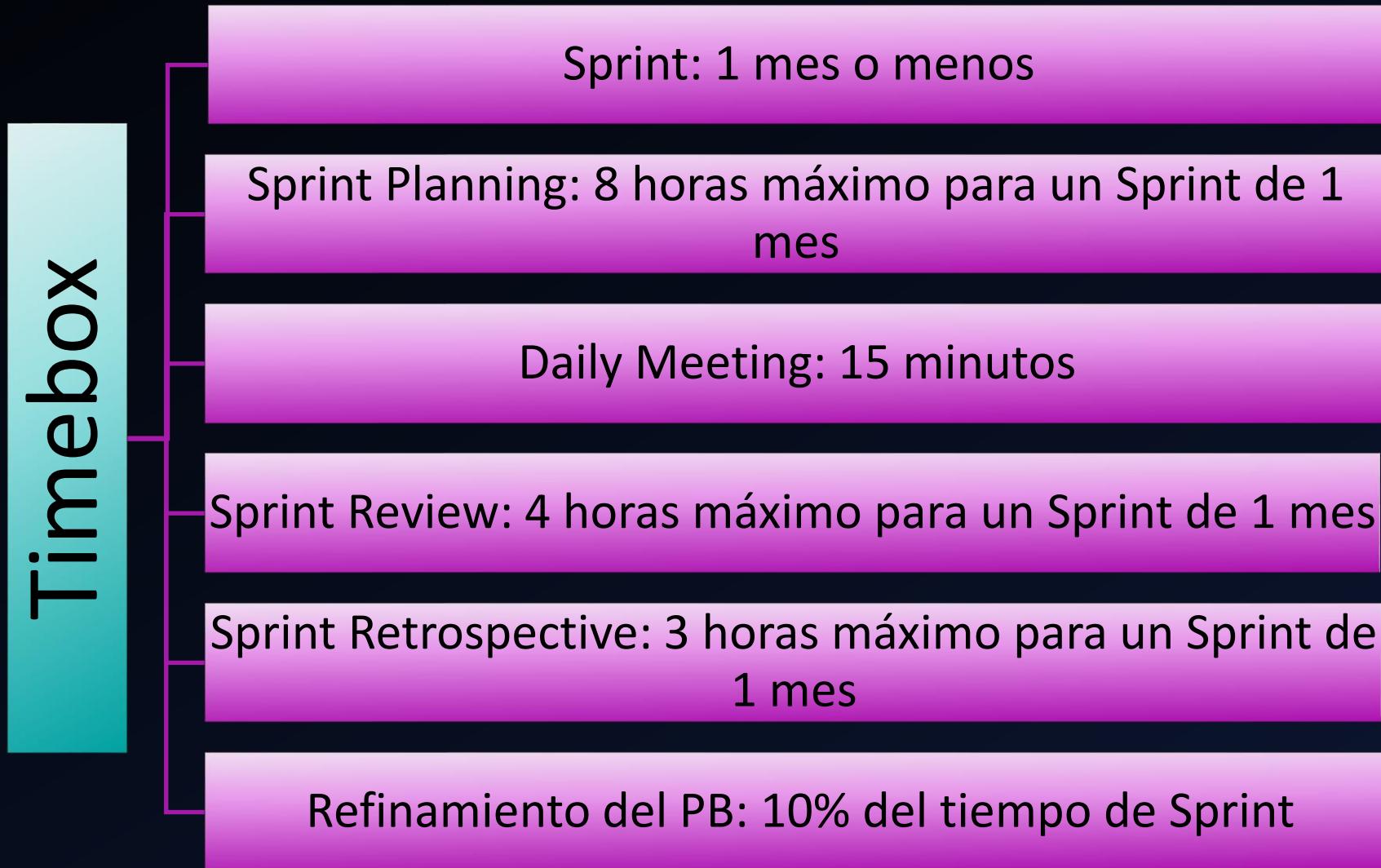


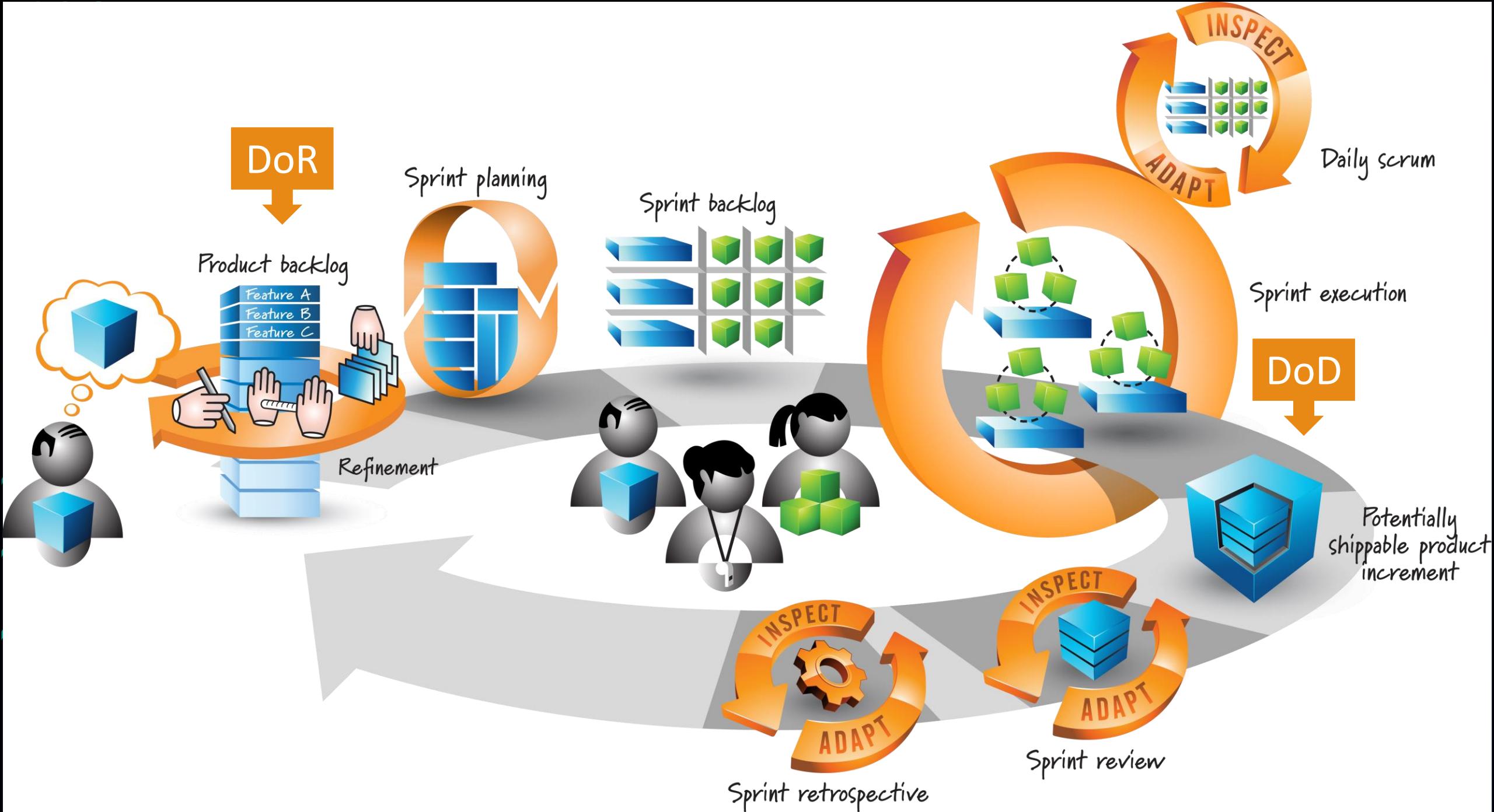
Sprint Backlog →
Objetivo del Sprint



Product
Increment
→ DoD

Timebox en Scrum





Definición de “Listo” (Ready)

- Valor de negocio claramente expresada.
- Detalles suficientemente comprendidos por el Equipo de forma tal que puedan tomar una decisión informada sobre si pueden completar **el ítem del product Backlog (PBI)**.
- Dependencias identificadas y no hay dependencias externas que puedan impedir que el PBI se complete.
- El equipo ha sido asignado adecuadamente para completar el PBI.
- El PBI ha sido debidamente estimado y es lo suficientemente pequeño para ser completado en un Sprint.
- Los criterios de aceptación son claros y testeables.
- Los criterios de performance si hay, son claros y testeables.
- El equipo comprende como mostrar el PBI en la Sprint Review.

9



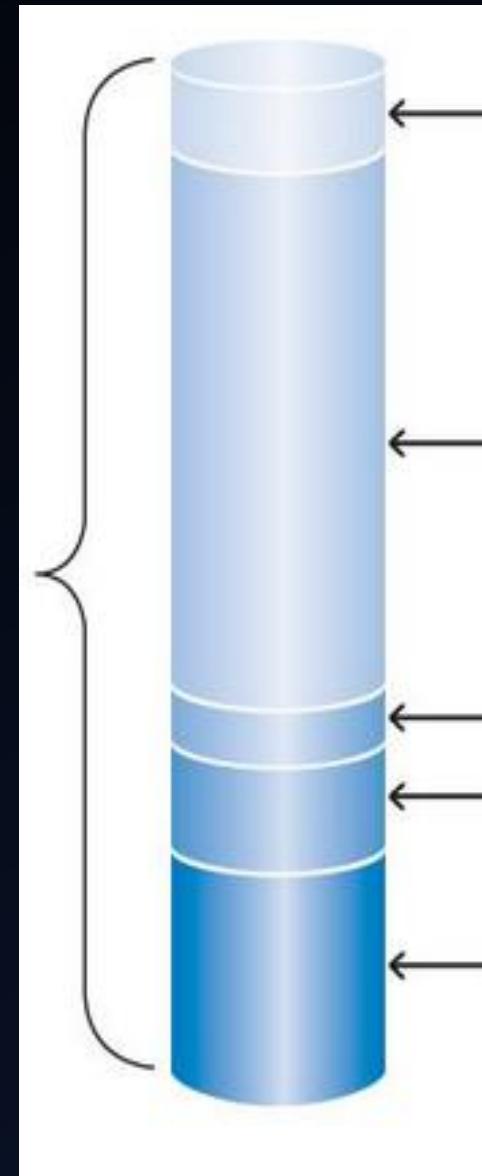
Definición de Hecho (DONE)	
<input type="checkbox"/>	Diseño revisado
<input type="checkbox"/>	Código Completo
<input type="checkbox"/>	Código refactorizado
<input type="checkbox"/>	Código con formato estándar
<input type="checkbox"/>	Código Comentado
<input type="checkbox"/>	Código en el repositorio
<input type="checkbox"/>	Código Inspeccionado
<input type="checkbox"/>	Documentación de Usuario actualizada
<input type="checkbox"/>	Probado
<input type="checkbox"/>	Prueba de unidad hecha
<input type="checkbox"/>	Prueba de integración hecha
<input type="checkbox"/>	Prueba de sistema hecha
<input type="checkbox"/>	Cero defectos conocidos
<input type="checkbox"/>	Prueba de Aceptación realizada
<input type="checkbox"/>	En los servidores de producción



Capacidad del Equipo en un Sprint

11

**Capacidad total
del SPRINT**



Buffer del Sprint

Capacidad de trabajo en los PBI's

Tiempo fuera del personal

Otros compromisos externos: soporte,
mantenimiento, trabajo en otros proyectos

Otras actividades del sprint: sprint Planning,
sprint Review, retrospective, grooming

Cálculo de Capacidad del Equipo en un Sprint

Persona	Días disponibles (sin tiempo personal)	Días para otras actividades Scrum	Horas por día	Horas de Esfuerzo disponibles
Jorge	10	2	4-7	32-56
Betty	8	2	5-6	30-36
Simón	8	2	4-6	24-36
Pedro	9	2	2-3	14-21
Raúl	10	2	5-6	40-48
Total				140-197

Ambiente de trabajo

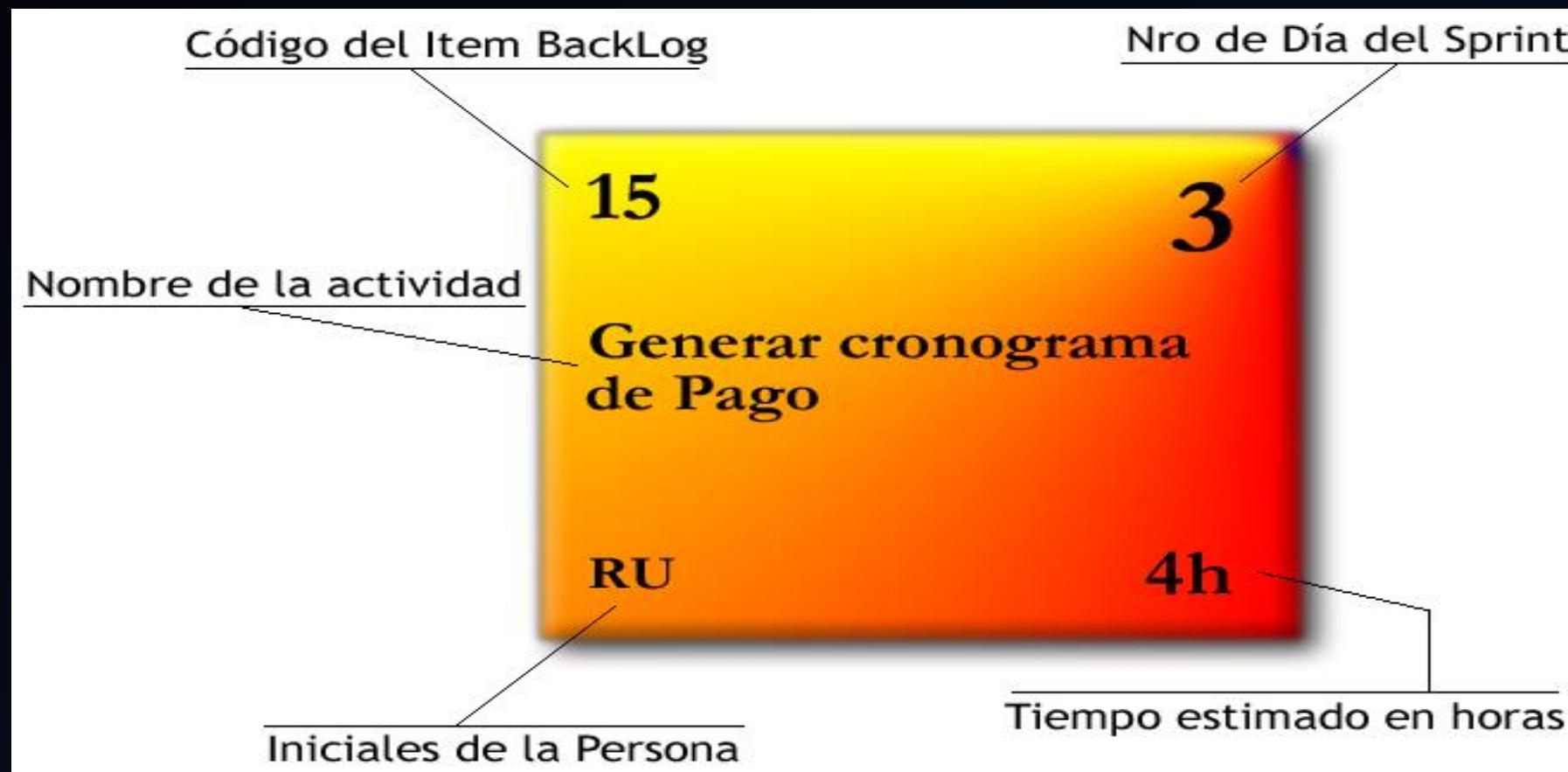
- Abierto
 - El silencio absoluto es un mal signo
- Pizarrones
- El equipo define sus horarios



Herramientas de Scrum



Taskboard



Taskboard

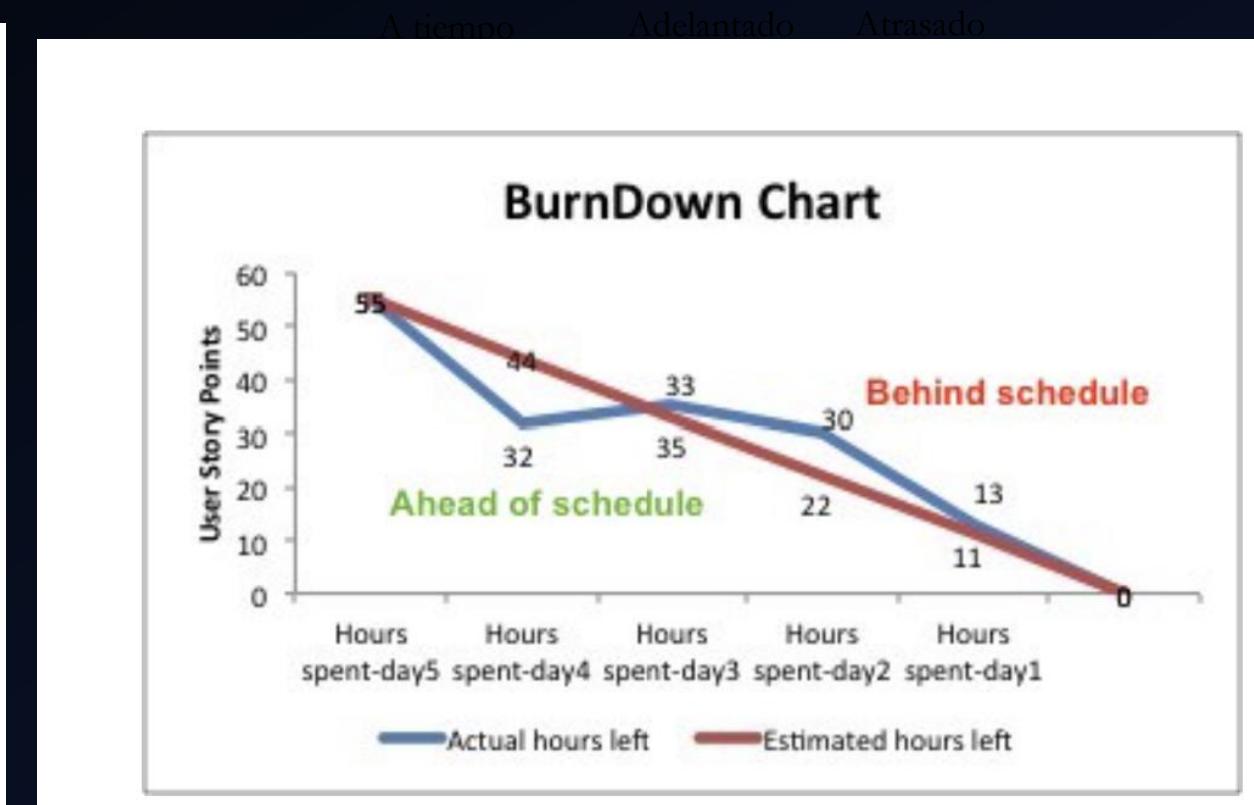
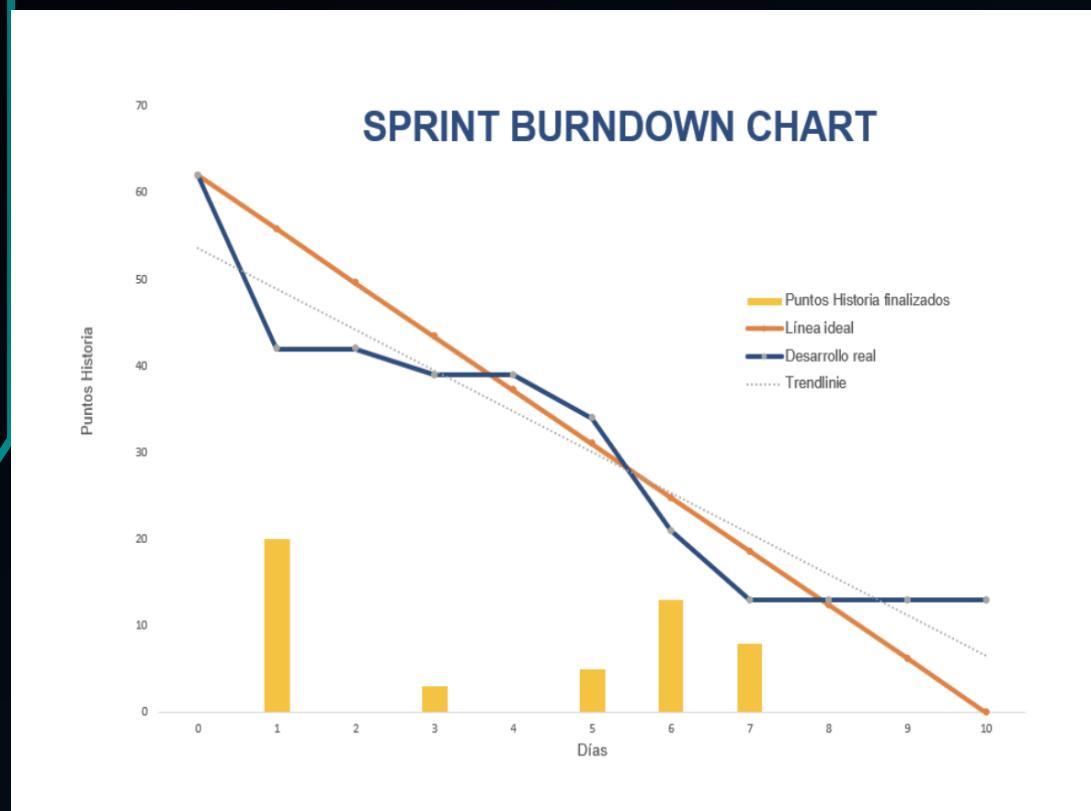
Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 Code the... 2 Test the... 8	Test the... 8 Code the... 8 Test the... SC	Code the... DC 4 Test the... SC 8	Test the... SC 6 Code the... DC 8 Test the... SC 8 Test the... SC 8 Test the... SC 6
As a user, I... 5 points	Code the... 8 Code the... 4 Test the... 8 Code the... 6	Code the... DC 8		Test the... SC 8 Test the... SC 6

BUENA GRANULARIDAD

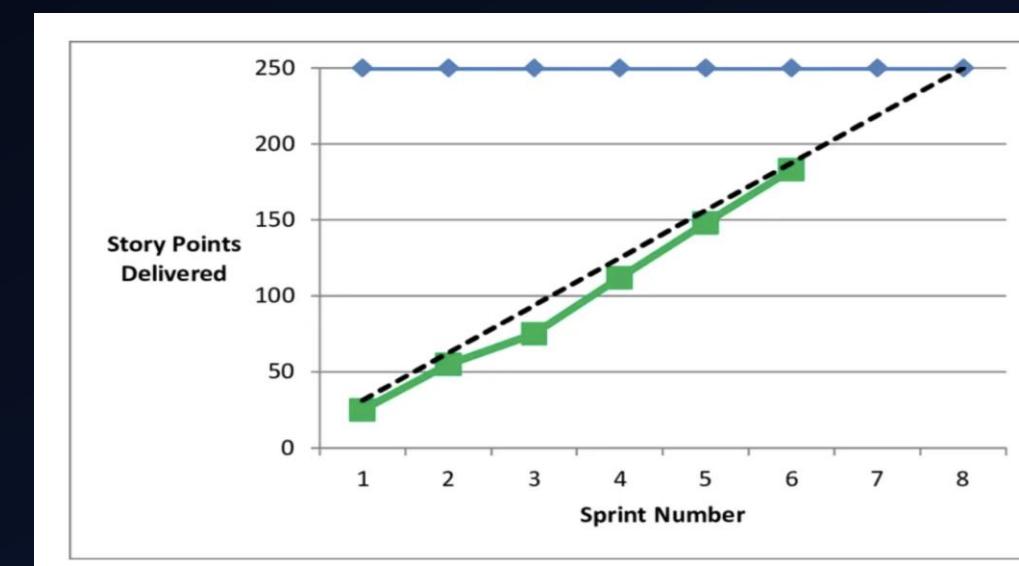
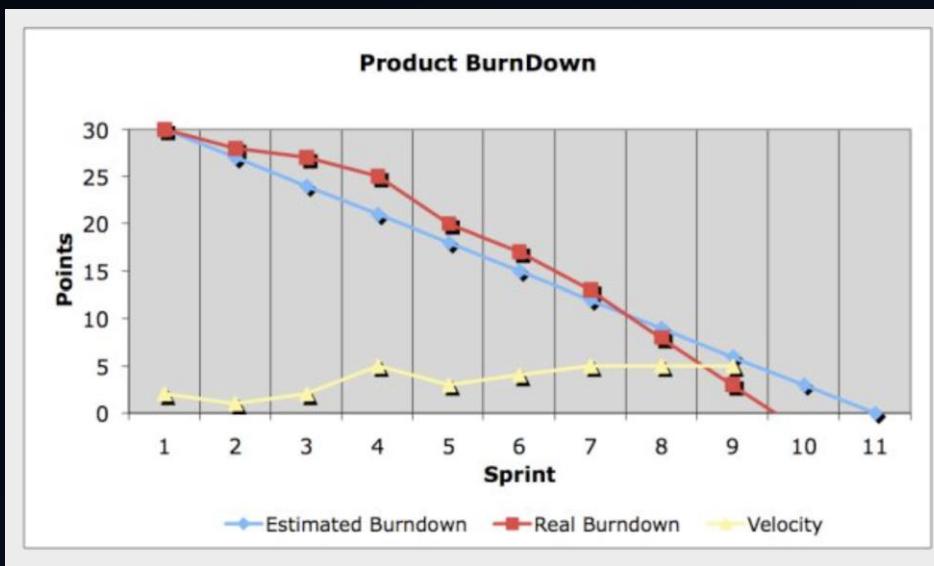
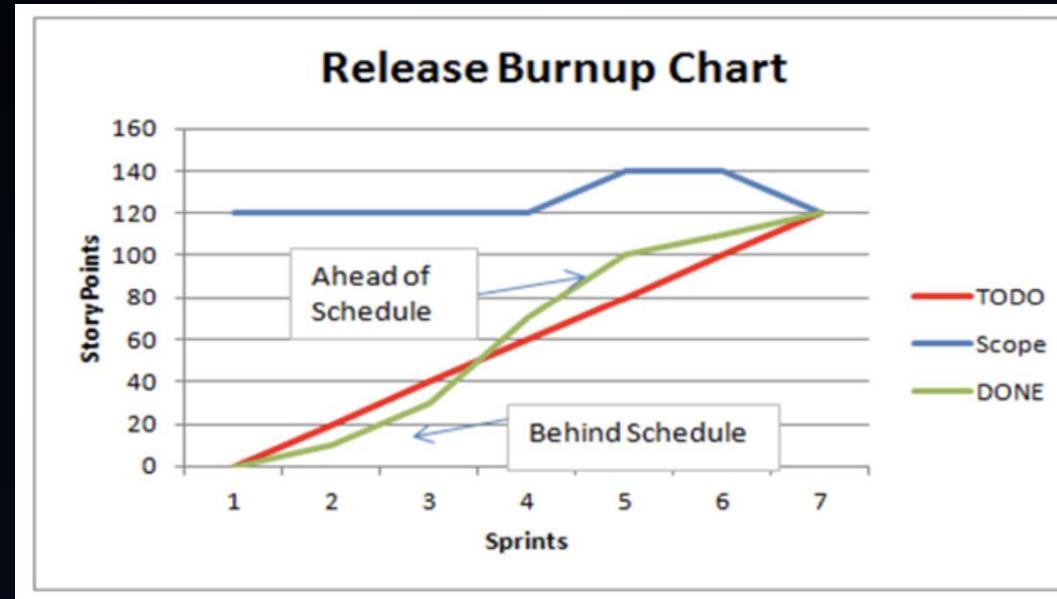


**MALA
GRANULARIDAD**

Sprint Burndown Charts

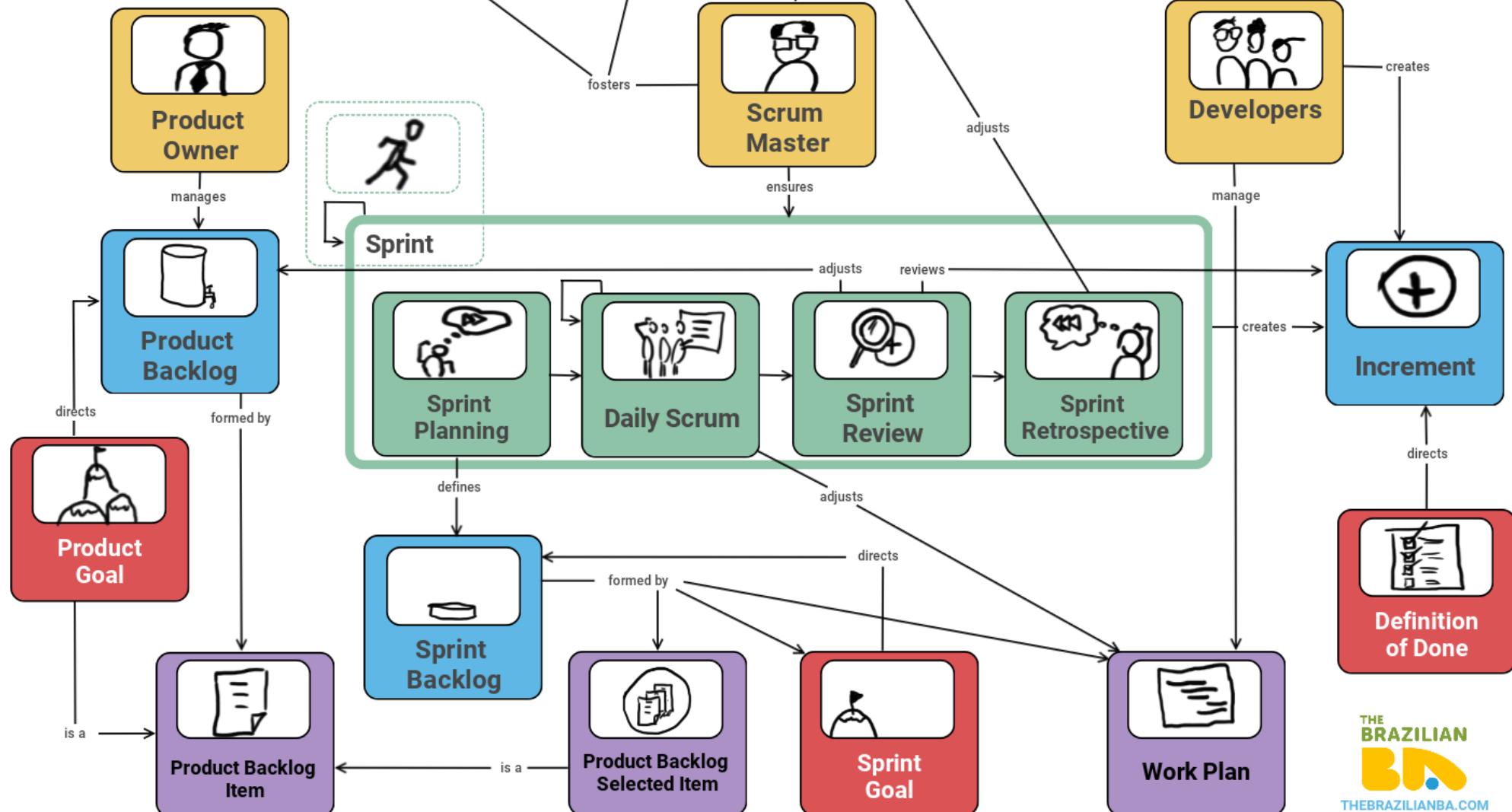
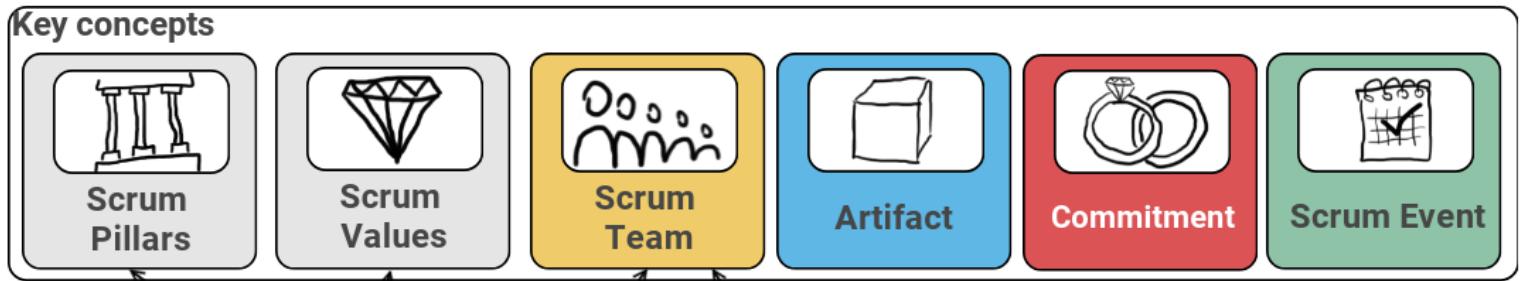


Gráficos de Seguimiento del Producto

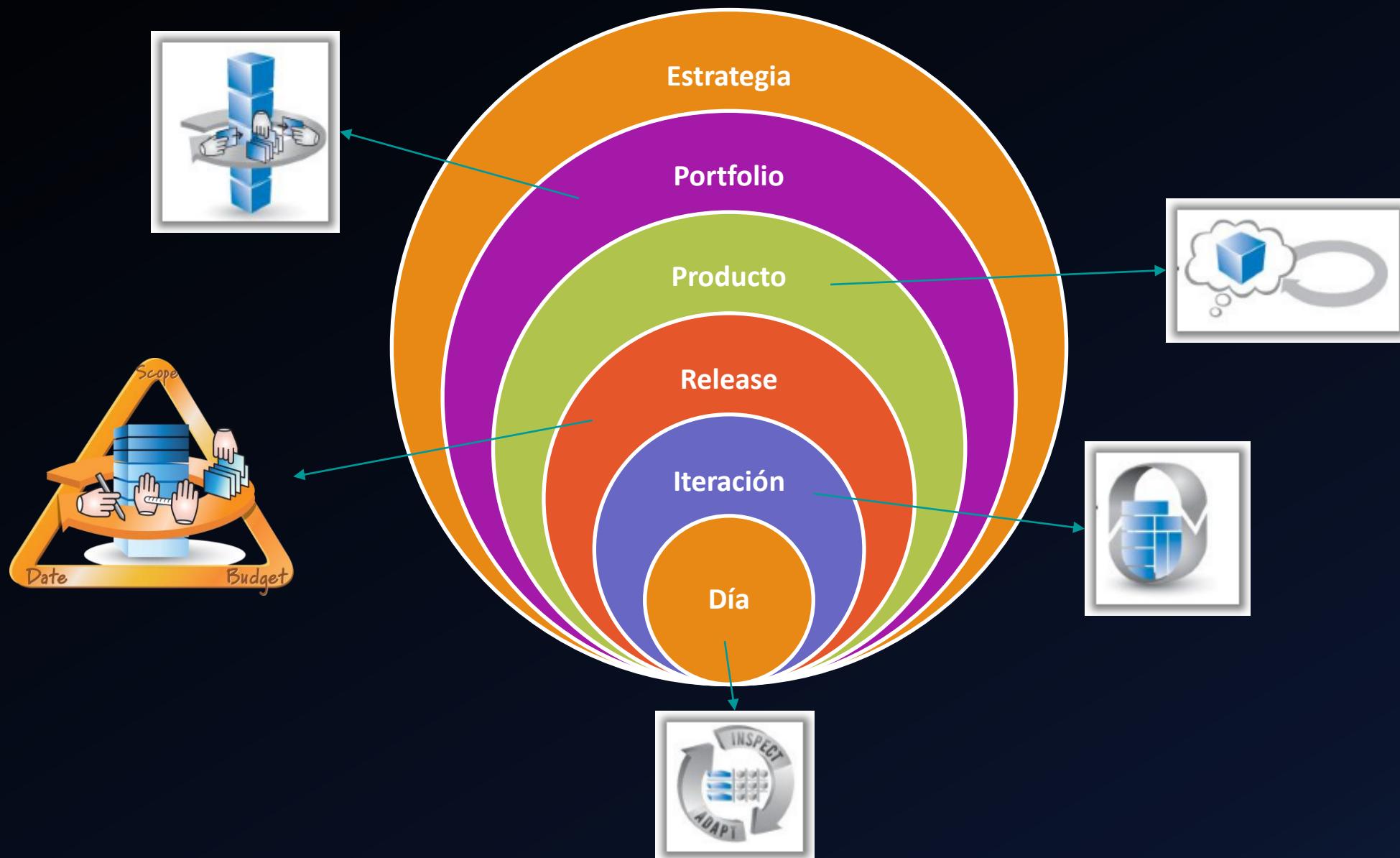


The Scrum Guide on a single page

adapted from The Scrum Guide, 2020
by Ken Schwaber & Jeff Sutherland

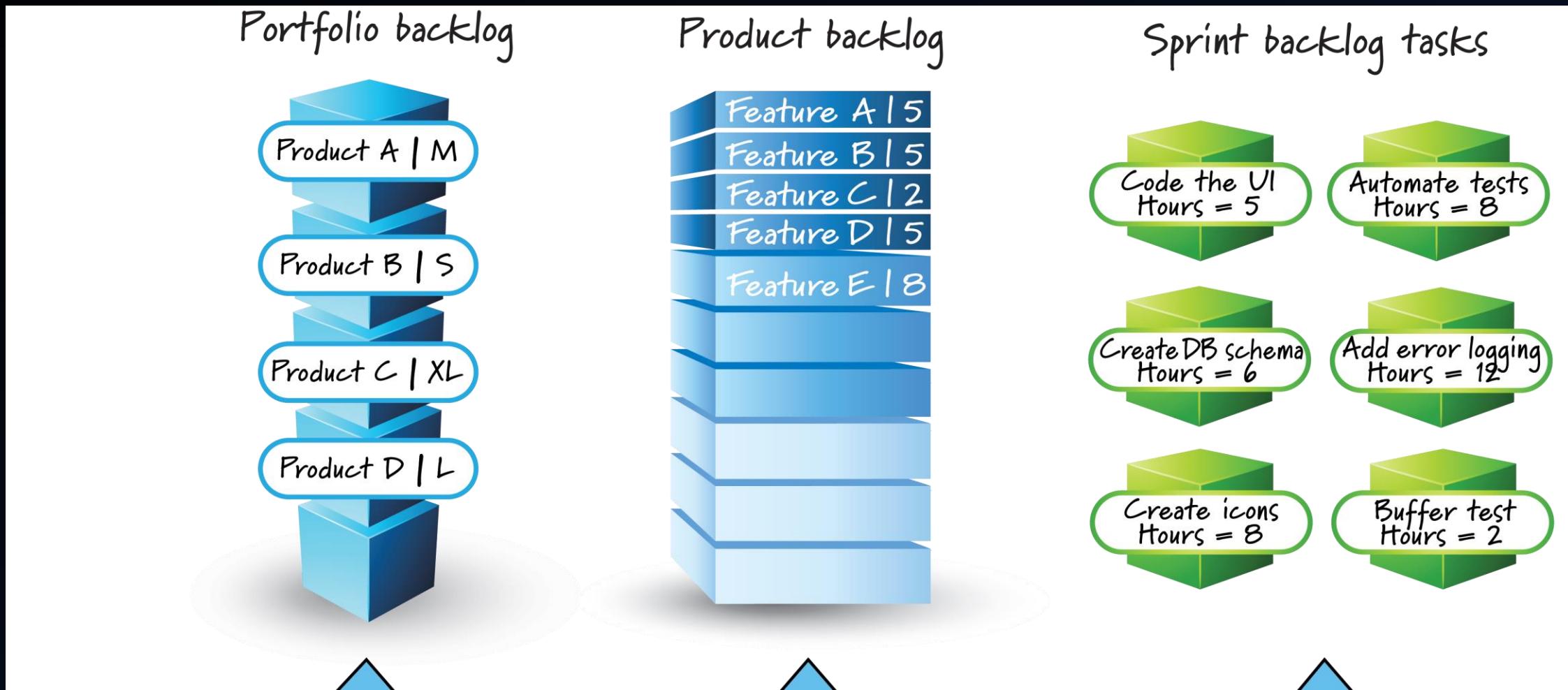


Múltiples niveles de planificación

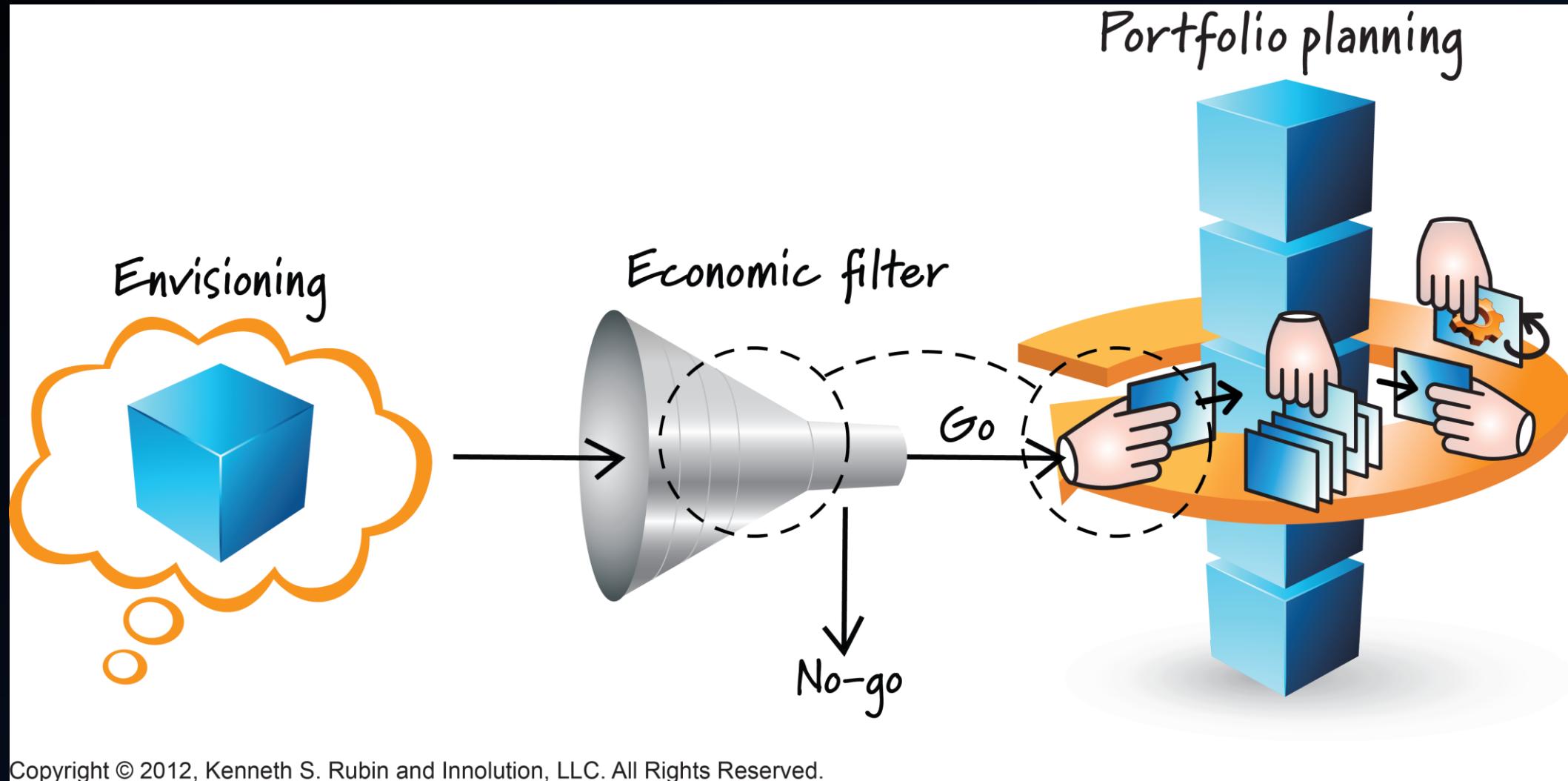


Nivel	Horizonte	Quién	Foco	Entregable
Portfolio	1 año o más	Stakeholders y Product Owners	Administración de un Portfolio de Producto	Backlog de Portfolio
Producto	Arriba de varios meses o más	Product Owner y Stakeholders	Visión y evolución del producto a través del tiempo	Visión de Producto, Roadmap y características de alto nivel
Release	3 (o menos) a 9 meses	Equipo Scrum, Stakeholders	Balancear continuamente el valor de cliente y la calidad global con las restricciones de alcance, cronograma y presupuesto	Plan de Release
Iteración	Cada iteración (de 1 semana a 1 mes)	Equipo Scrum	Que aspectos entregar en el siguiente sprint	Objetivo del Sprint y Sprint Backlog
Día	Diaria	Equipo Scrum (al menos los que trabajan en IPB)	Cómo completar lo comprometido	Inspección del progreso actual y adaptación a la mejor forma de organizar el siguiente día de trabajo

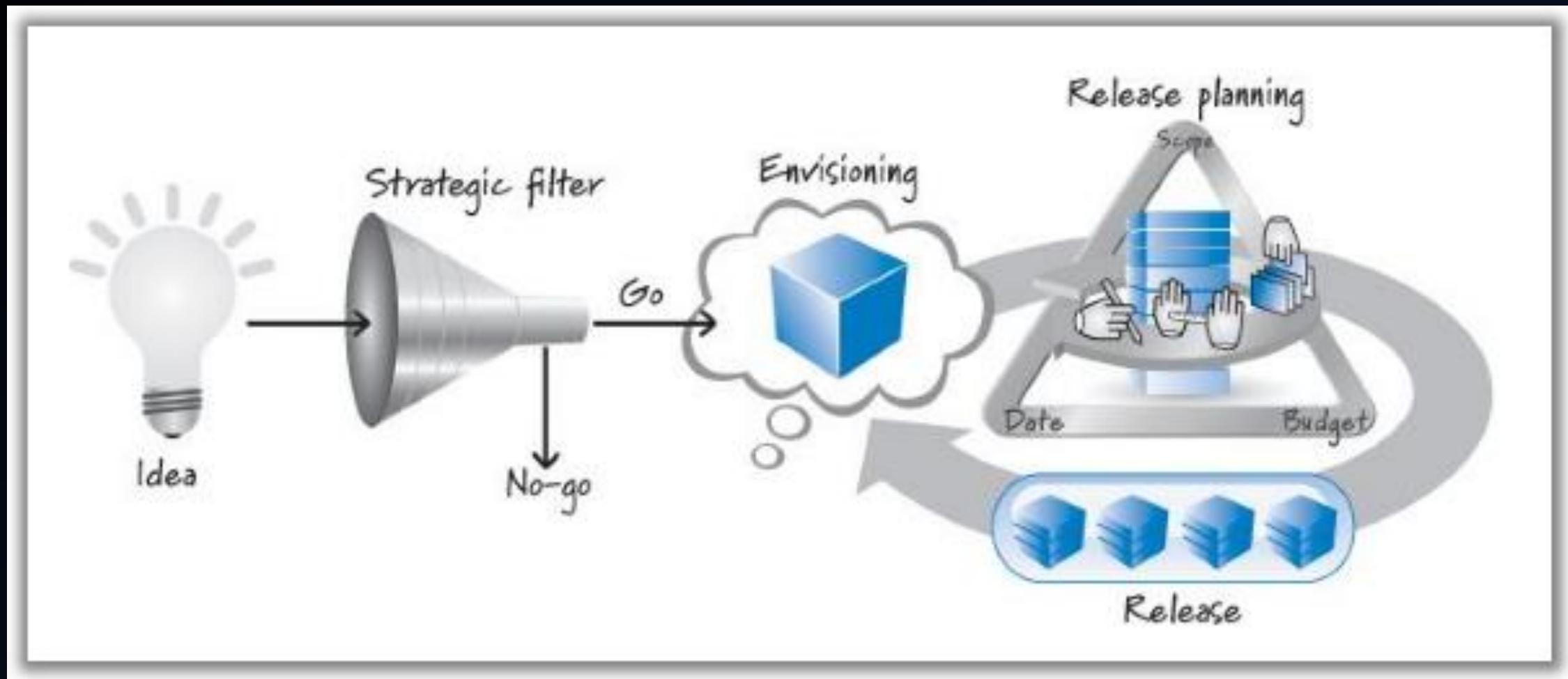
¿Qué y Cuándo estimar?

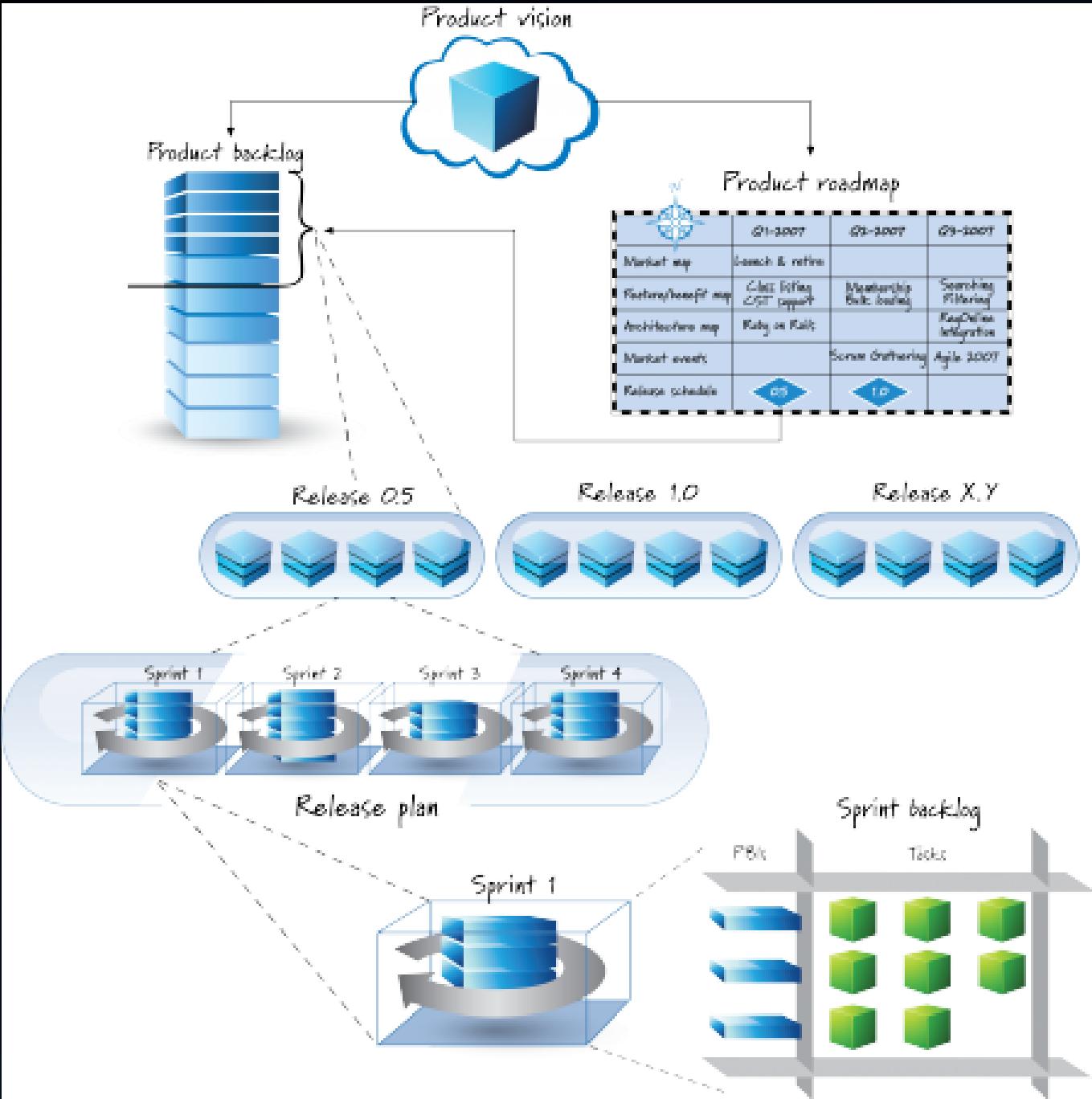


Planificación de Portfolio

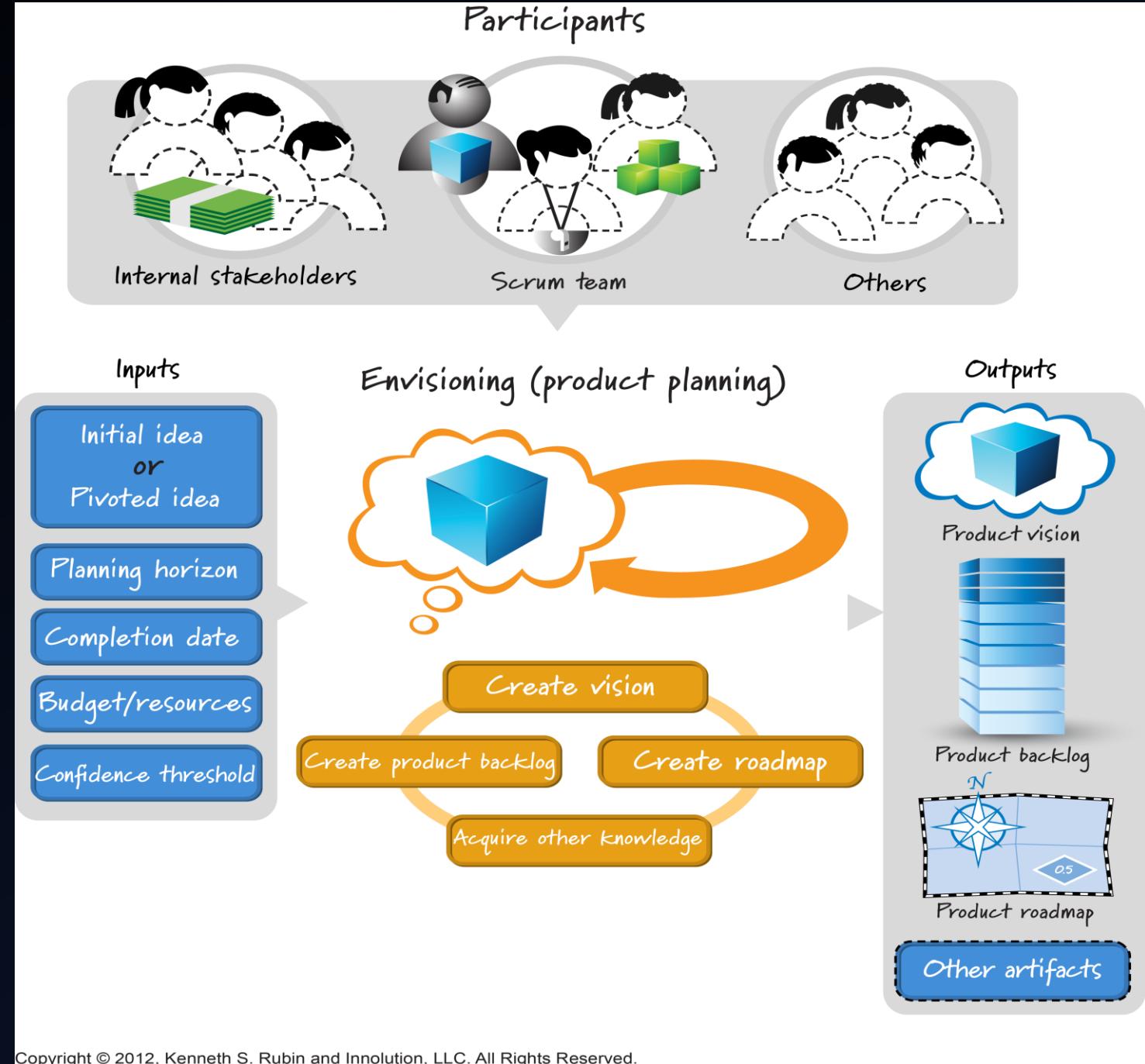


Planificación de Producto

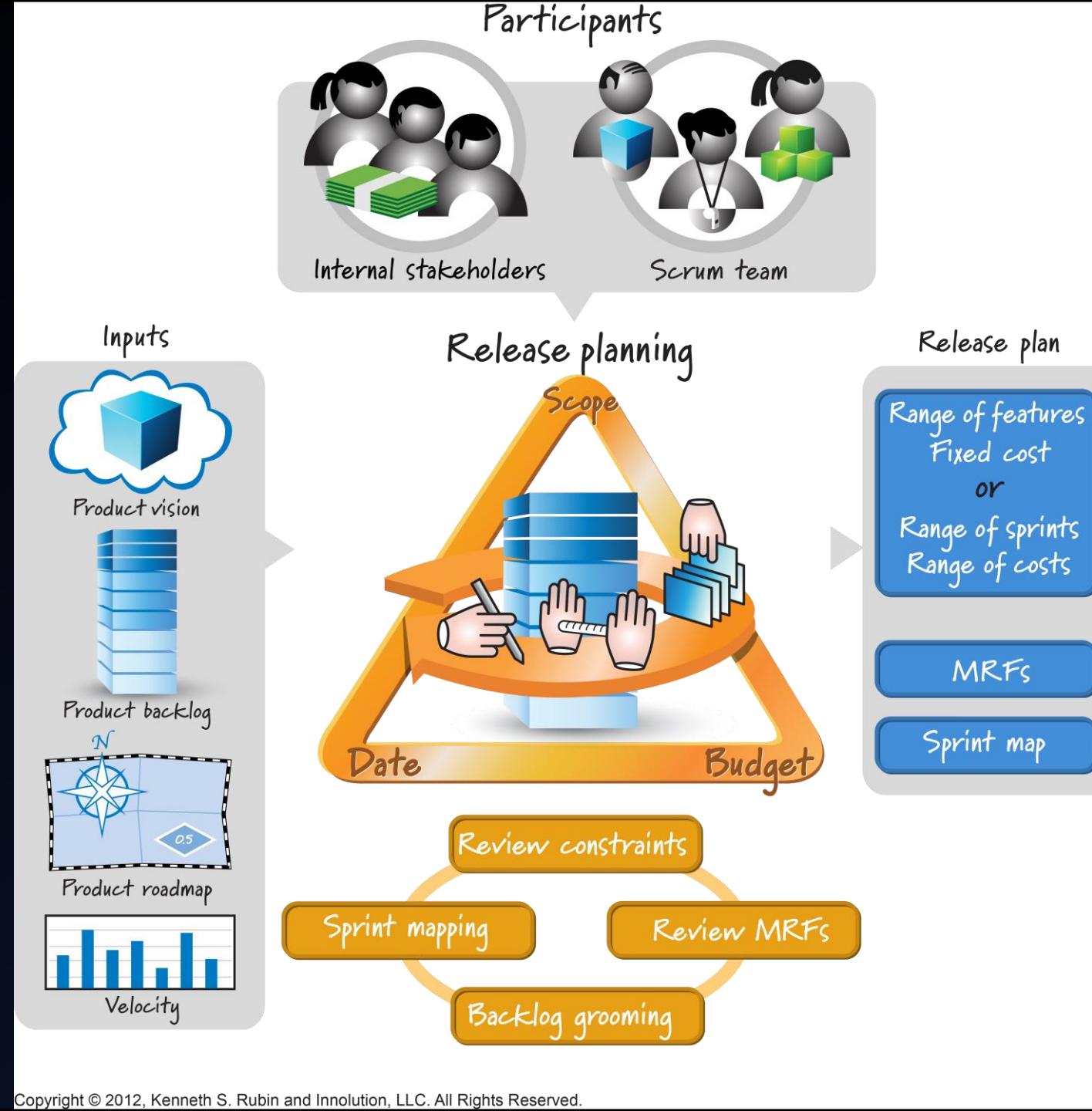




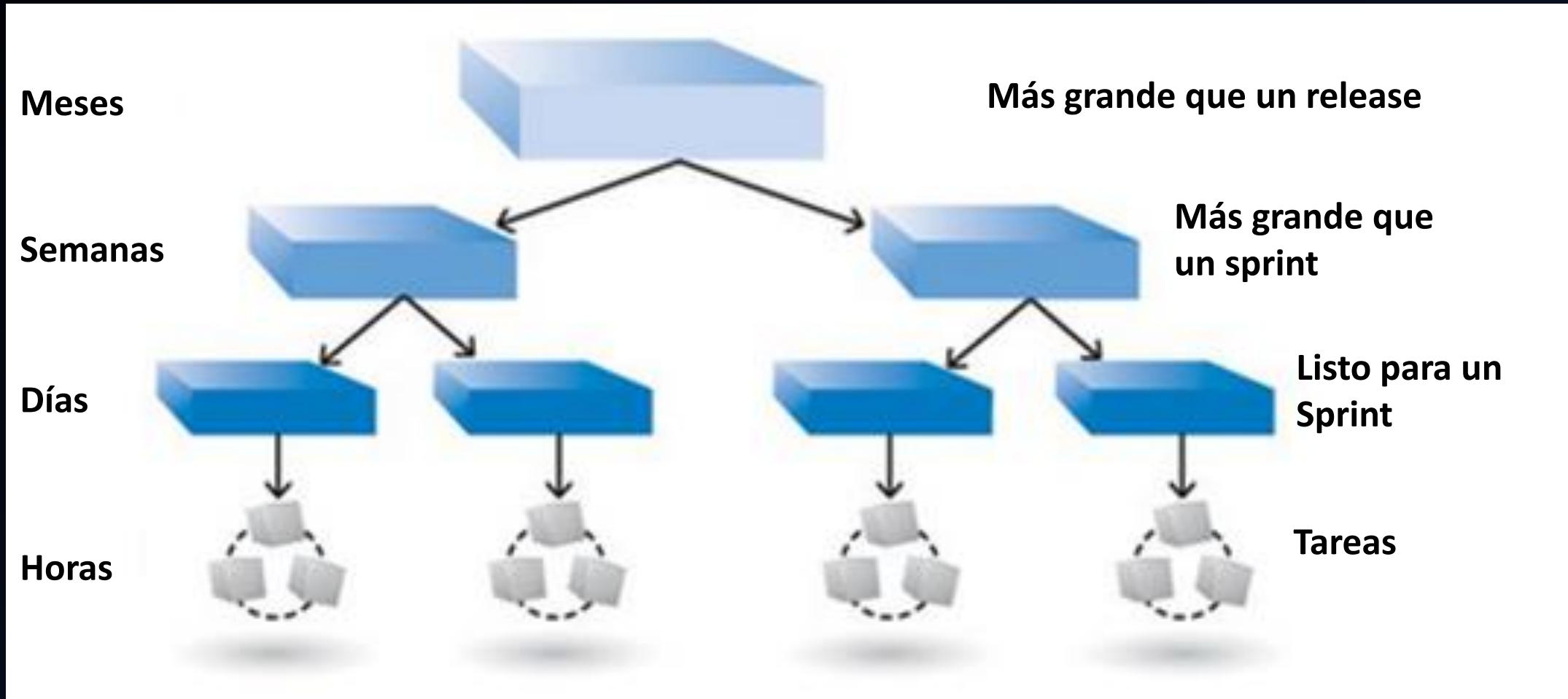
Planificación de Producto



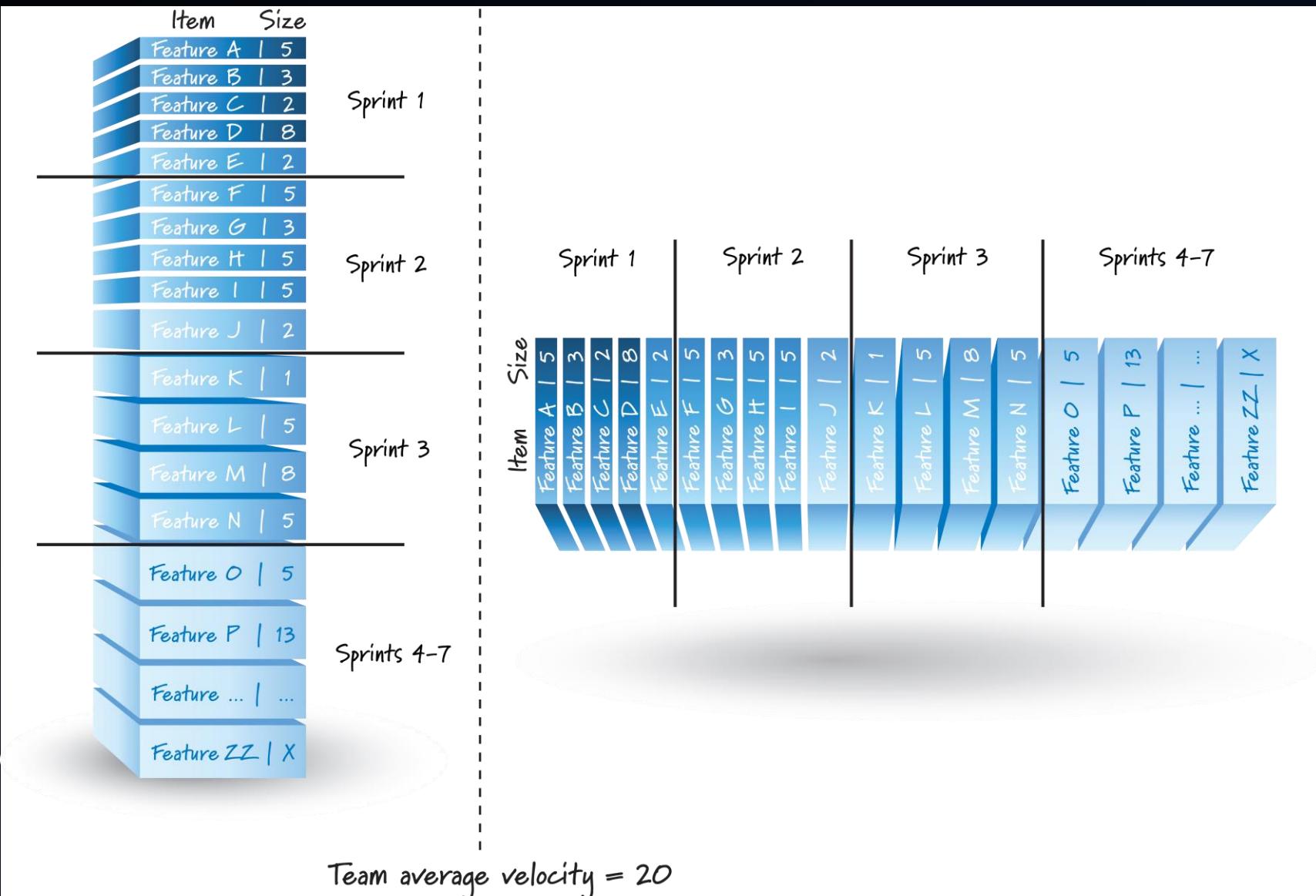
Planificación de Release



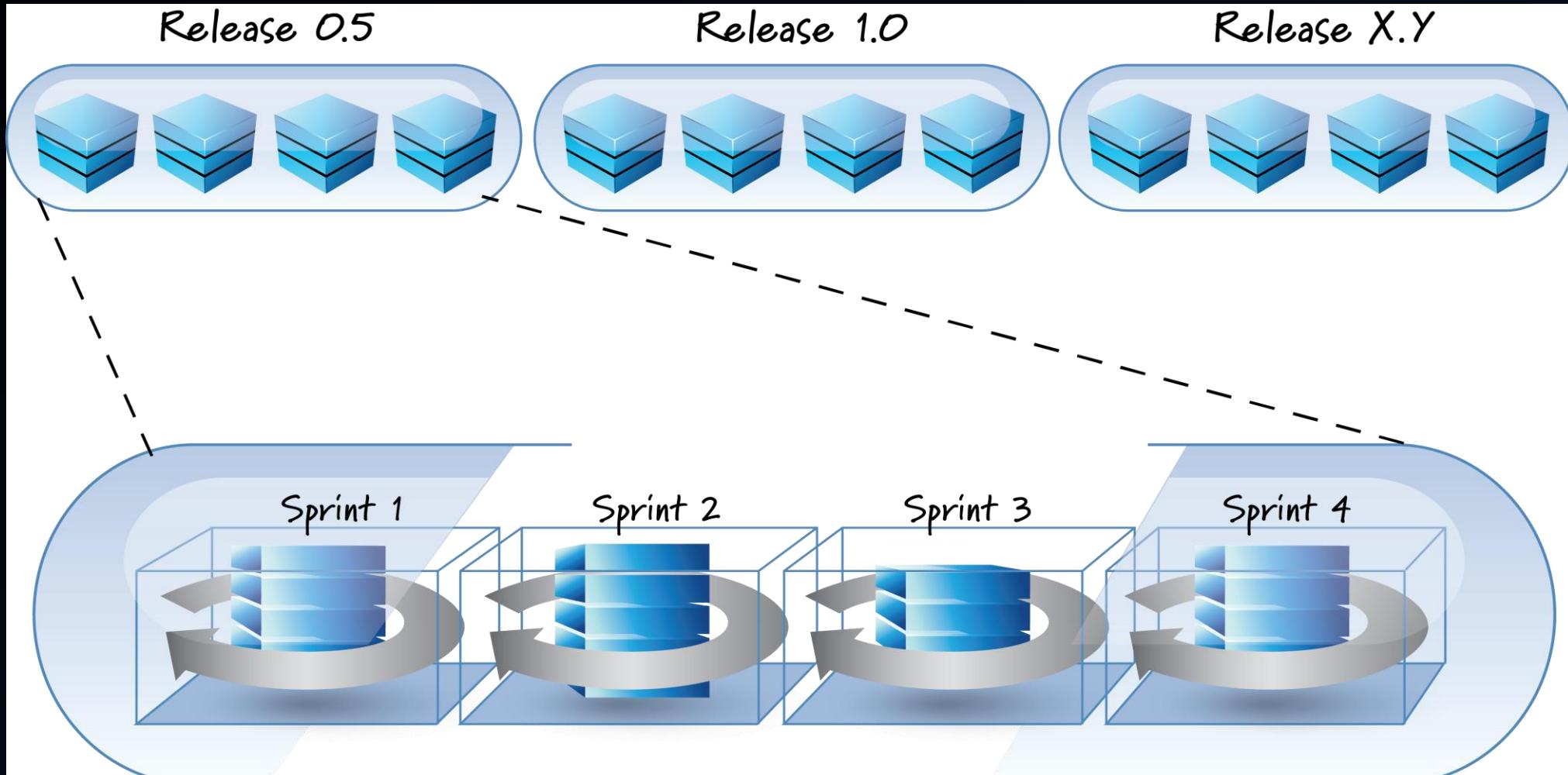
Niveles de granularidad distintos...



Planificación de Release



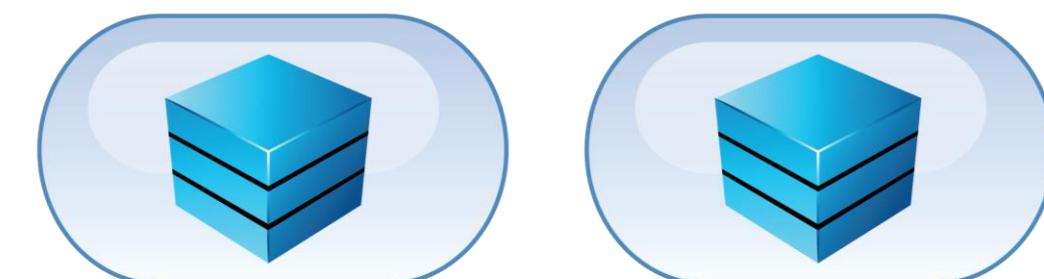
Releases y Sprints



Cadencia de los Sprints



Release luego de múltiples sprints

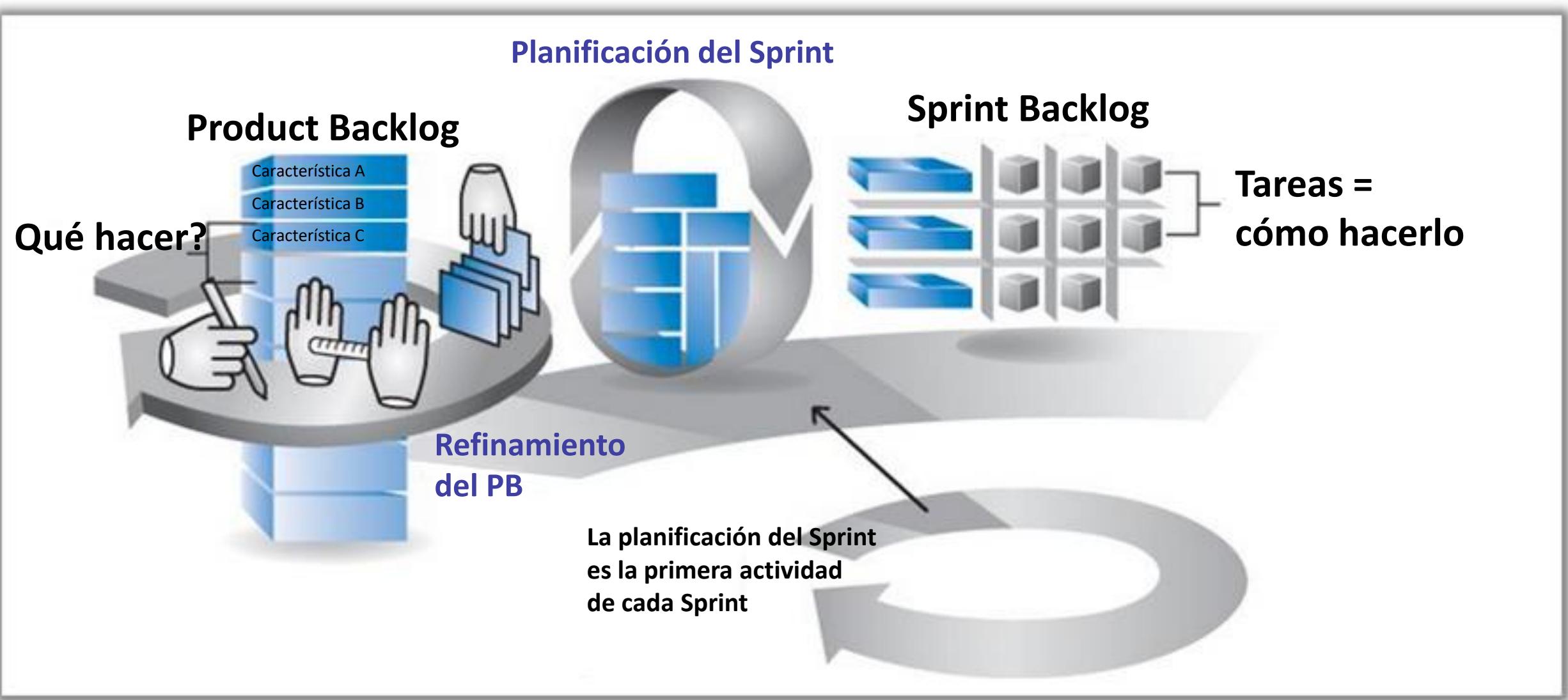


Release luego de cada sprint



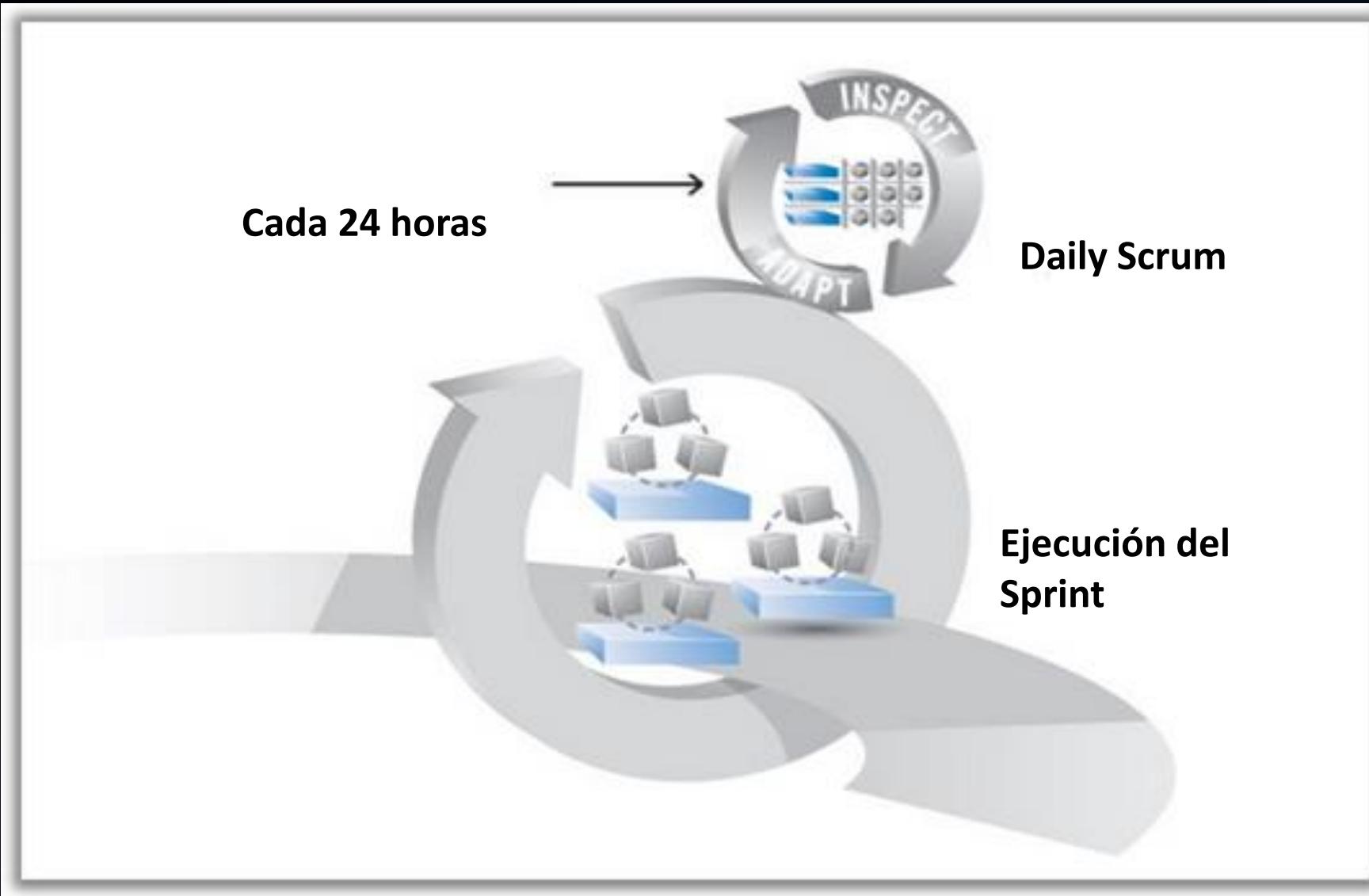
Release luego de cada feature

Planificación de Iteración: Sprint Planning

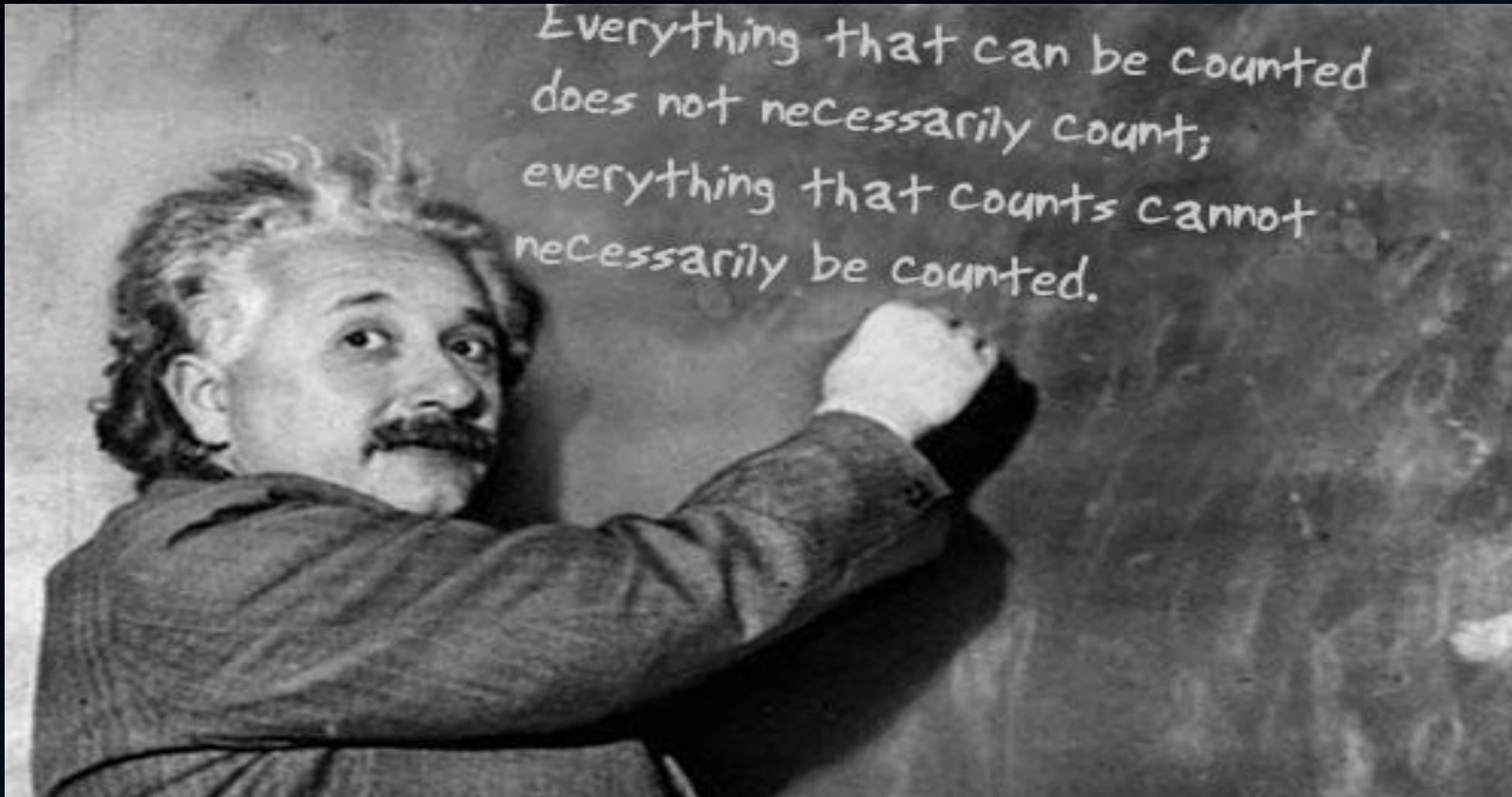


Planificación Diaria: Daily Scrum

34



Métricas en ambientes ágiles





Regla de Oro Ágil sobre Métricas

La medición es una salida, no una actividad

Una filosofía minimalista
sobre las Métricas:

Medir la que sea necesario
y nada más.



Dos principios ágiles que guían la elección de las Métricas

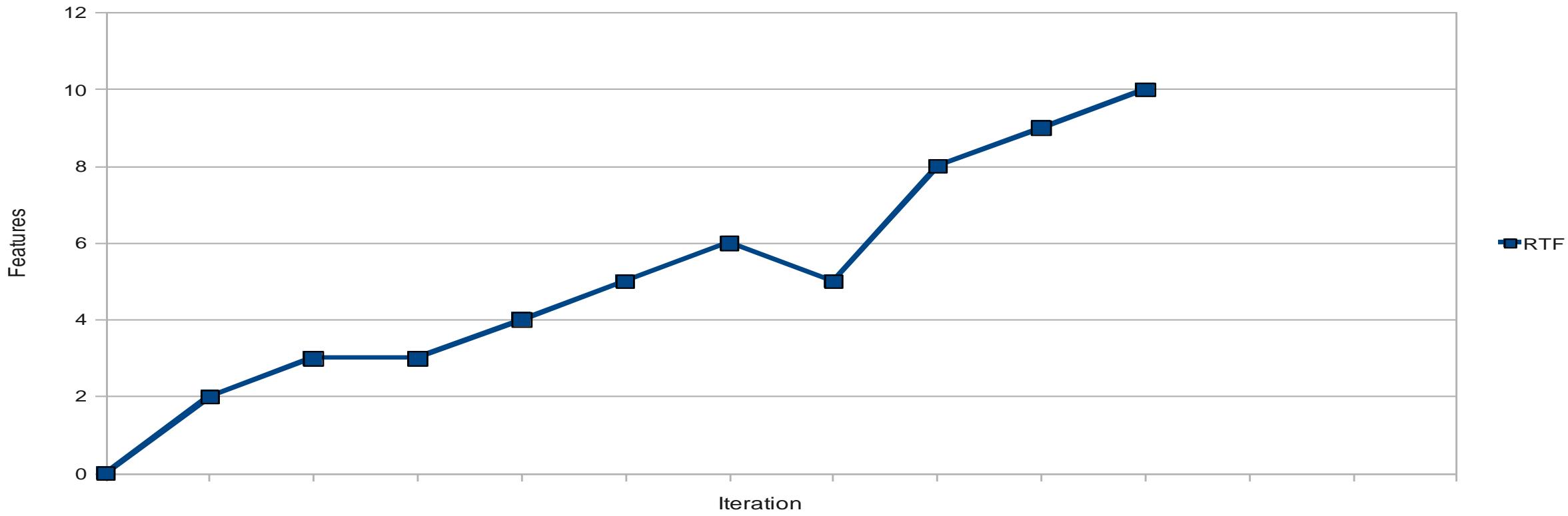
“Nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software valioso.”

y

“El Software trabajando es la principal medida de progreso.”

Running Tested Features (RTF)

Running Tested Features



Capacidad

- **Horas de Trabajo Disponibles por día (WH) X Días Disponibles Iteración (DA) = Capacidad**

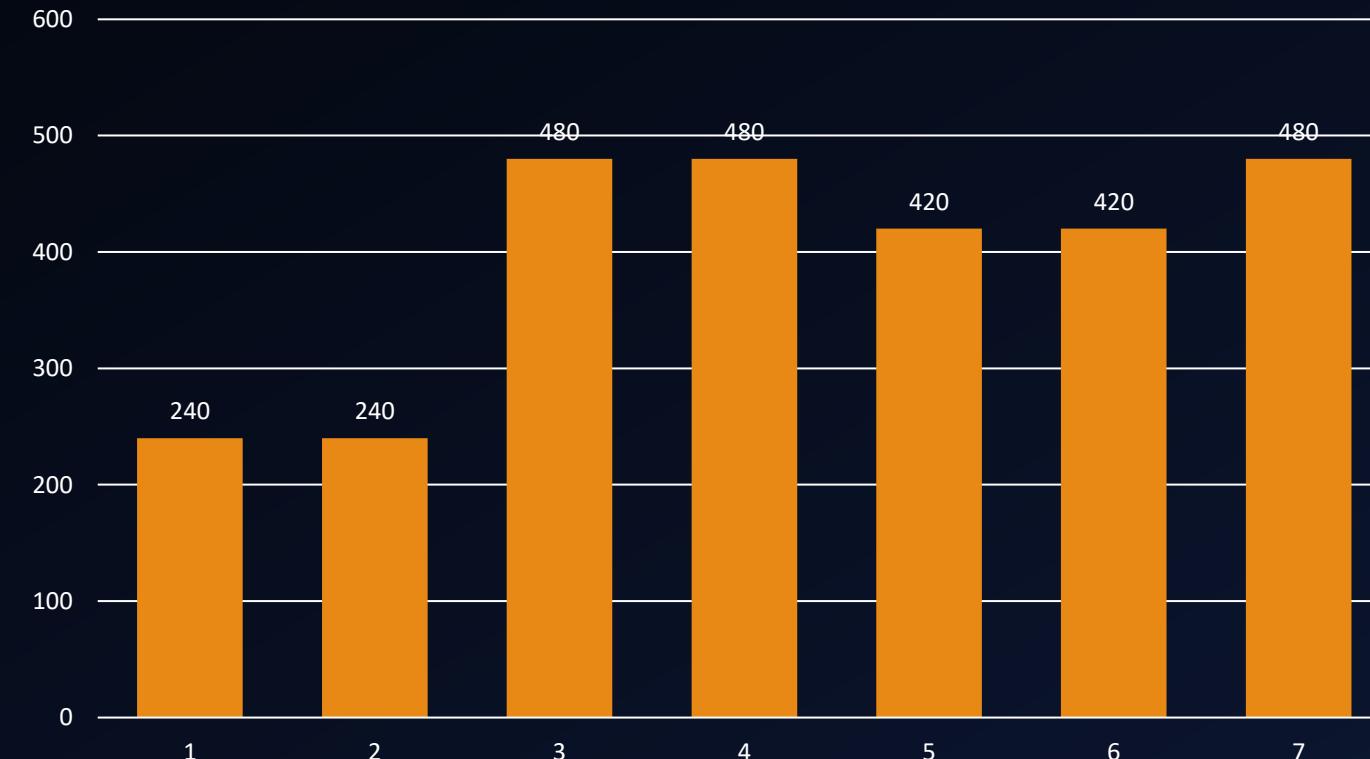
$$WH \times DA = \text{Capacidad}$$

- Ejemplo:
 - Equipo de 8 miembros
 - 4 miembros disponibles los 2 primeros sprints.
 - 1 miembro se casa en sprints 5 y 6
 - 6 horas de trabajo

Capacidad

Sprint	1	2	3	4	5	6	7	Total
Horas	240	240	480	480	420	420	480	2760
Puntos de Historia	30	30	45	60	58	52	60	335

Capacidad

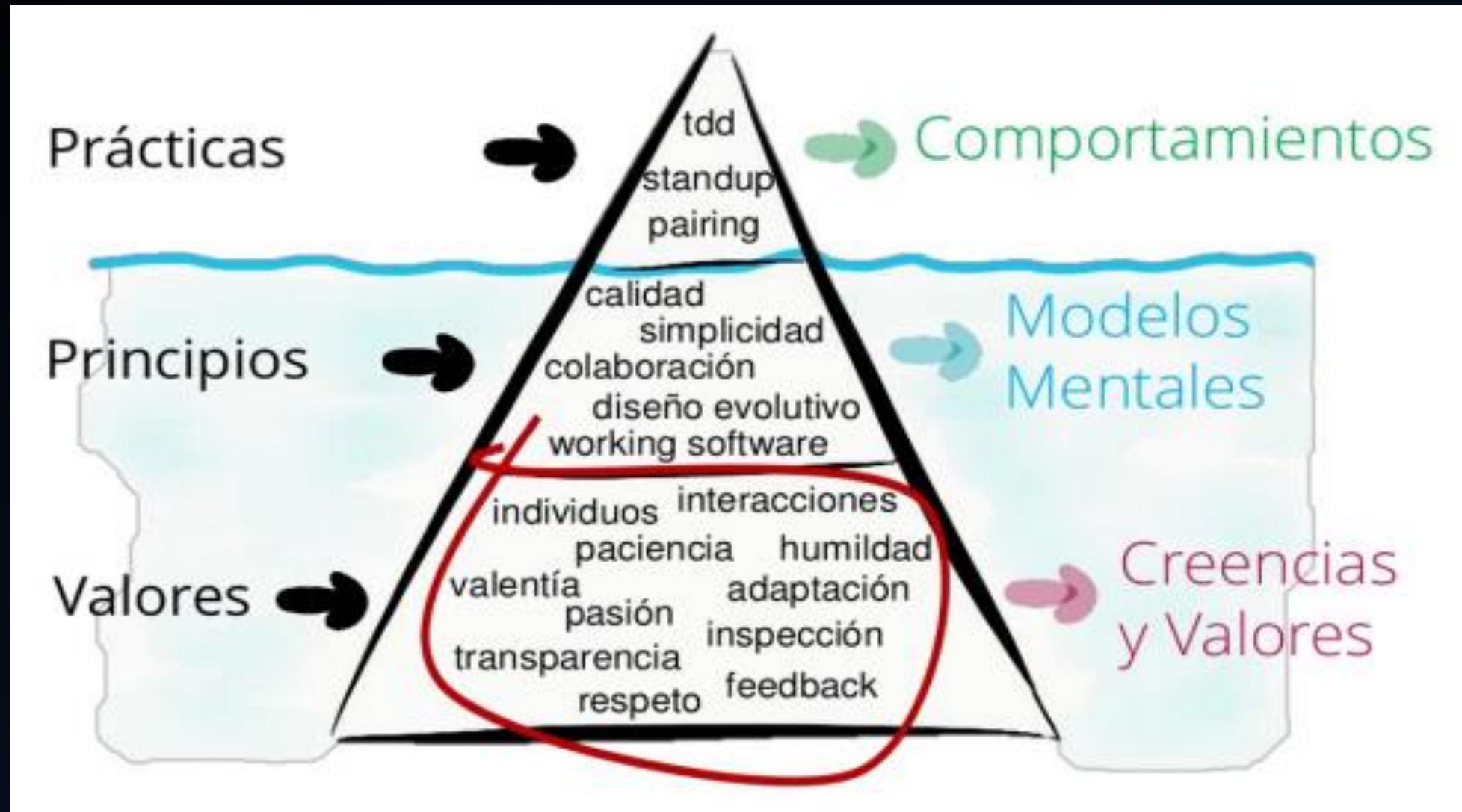


Velocidad (Velocity)

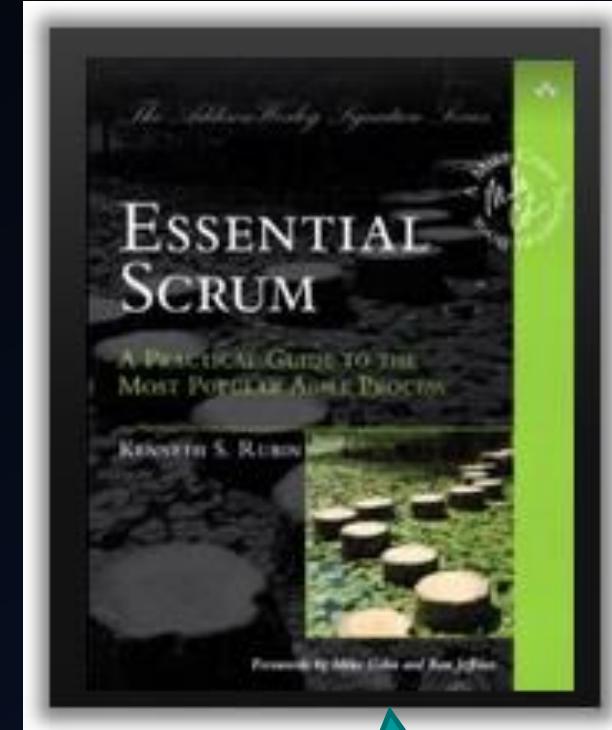
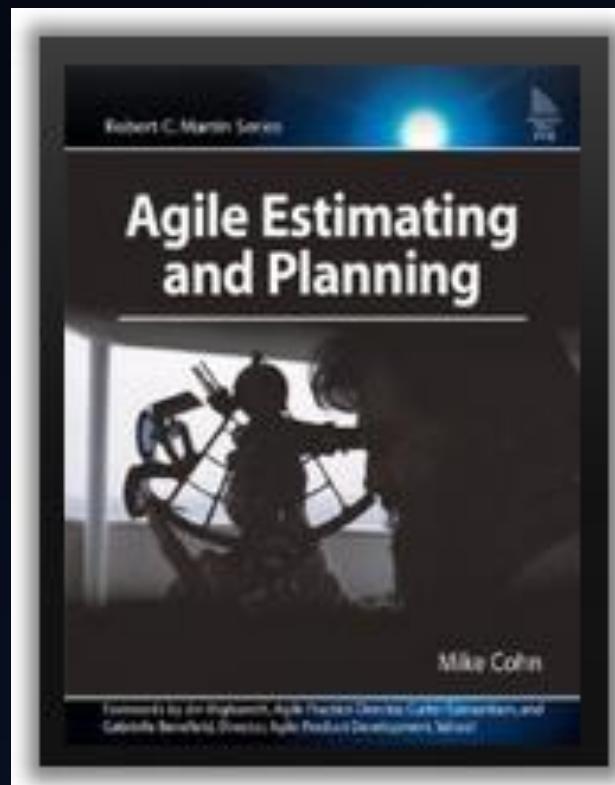
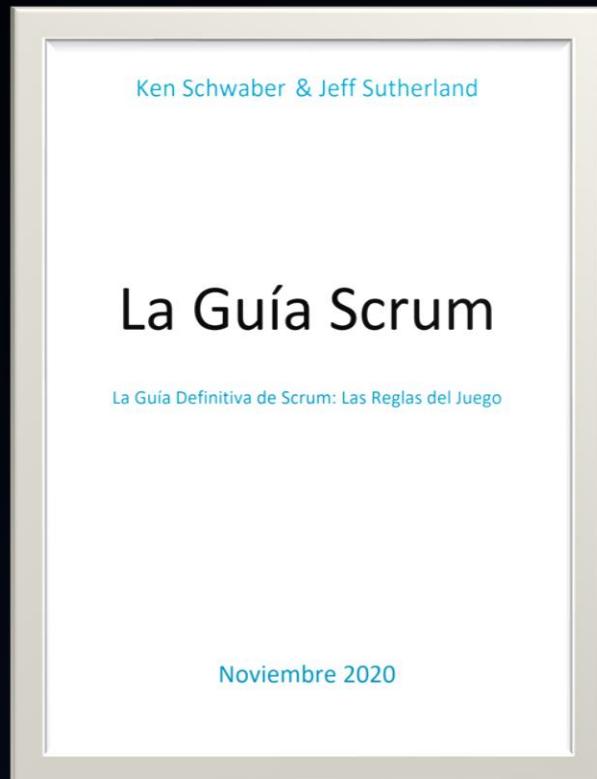
- Velocidad/ Velocity es una medida (métrica) del progreso de un equipo. Se calcula sumando el número de story points (asignados a cada user story) que el equipo completa durante la iteración.
- Se cuentan los story points de las users Stories que están completas, no parcialmente completas.
- La Velocidad corrige los errores de estimación.



Recordemos.... ¿Dónde está la Agilidad?



Bibliografía



Un agradecimiento especial por los gráficos que son de este libro ☺!!!