Planificación y Pruebas de Sistemas Software

Curso 2015-16

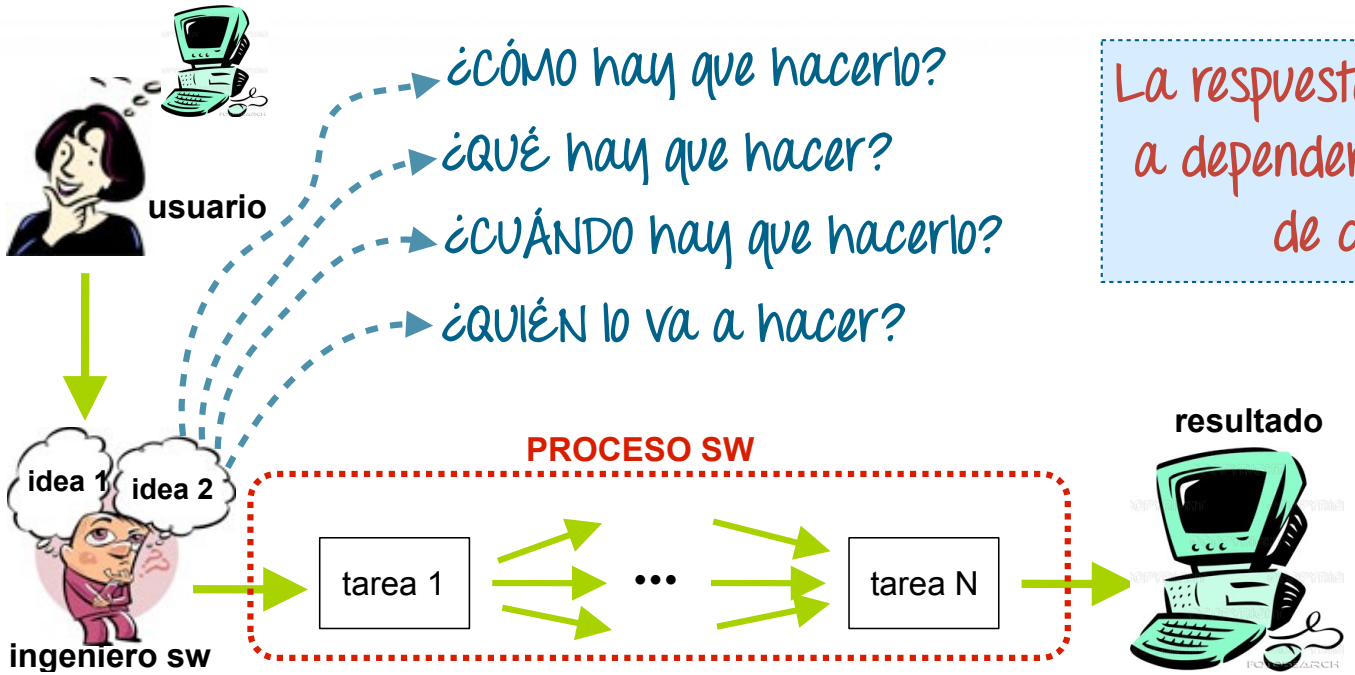
Sesión 13: Planificación de pruebas

Planificación del proyecto
Efectividad y alcance de las pruebas
Proceso de pruebas
Niveles de pruebas
Planificación de pruebas
Las pruebas en diferentes modelos de proceso
Otras prácticas de pruebas: Integraciones continuas (CI)

María Isabel Alfonso Galipienso
Universidad de Alicante eli@ua.es

La importancia de la planificación

- Cuando se planifica un proyecto, aunque el propósito siempre es el mismo...



La respuesta a estas preguntas va a depender del modelo de proceso de desarrollo elegido

Las pruebas SIEMPRE formarán parte de la planificación del proyecto!!!!

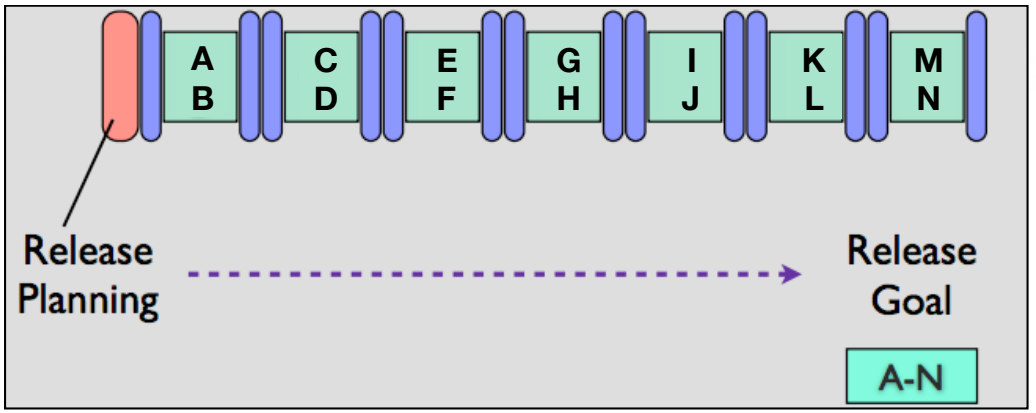
- se hace de forma diferente dependiendo del modelo de proceso:
  - Planificación predictiva vs. adaptativa
    - Los modelos iterativos y ágiles realizan una planificación adaptativa
  - Se establecen diferentes niveles de planificación: cada modelo de proceso considera determinados horizontes:
    - Los modelos ágiles planifican como mínimo a nivel de día, iteración y release

Planificación predictiva vs. adaptativa

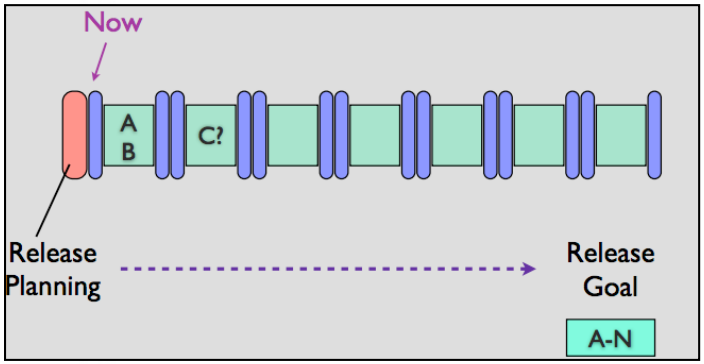
Imágenes adaptadas de: http://www.odd-e.com/material/2012/07_mda_nanyang/short_intro_agile.pdf

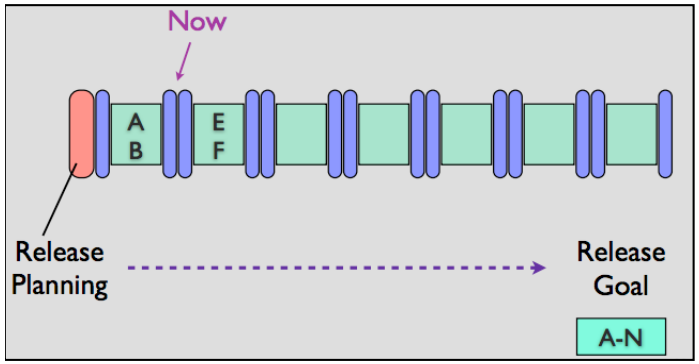
- Predictiva

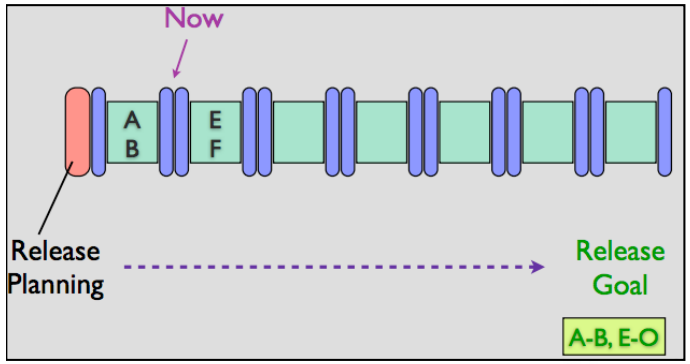
Soporta muy mal los cambios!!!



- Adaptativa







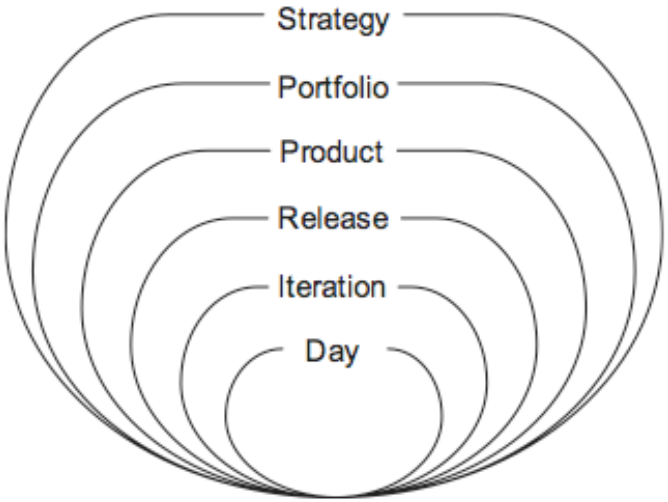
Un plan adaptativo intenta encontrar un equilibrio entre el esfuerzo invertido en realizar el plan, frente a la información (conocimiento) disponible en cada momento. Proporciona "agilidad": resulta "sencillo" incluir cambios!!!

No es posible tener claras todas las "variables" que intervienen en el desarrollo de un proyecto al comienzo del mismo, como por ejemplo requerimientos específicos, los detalles de la solución, cuestiones sobre el personal del proyecto, etc.

Múltiples niveles de planificación (I)

- Cuando nos marcamos objetivos, es importante recordar que no podemos "ver más allá del horizonte" y que la exactitud en nuestro plan decrecerá rápidamente tanto más cuanto más sobrepasemos dicho horizonte
  - Un proyecto "está en riesgo" si su planificación se extiende más allá del horizonte del planificador y no incluye tiempo para que éste "levante la cabeza", vuelva a mirar al nuevo horizonte, y realice los ajustes necesarios.

Imagen extraída de: Agile estimating and planning. Chap 3

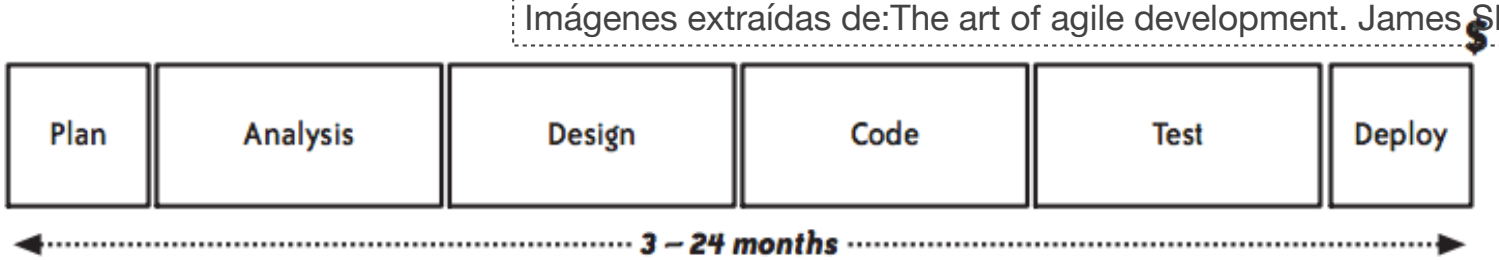


Los modelos ágiles planifican como mínimo a nivel de día, iteración y *release*

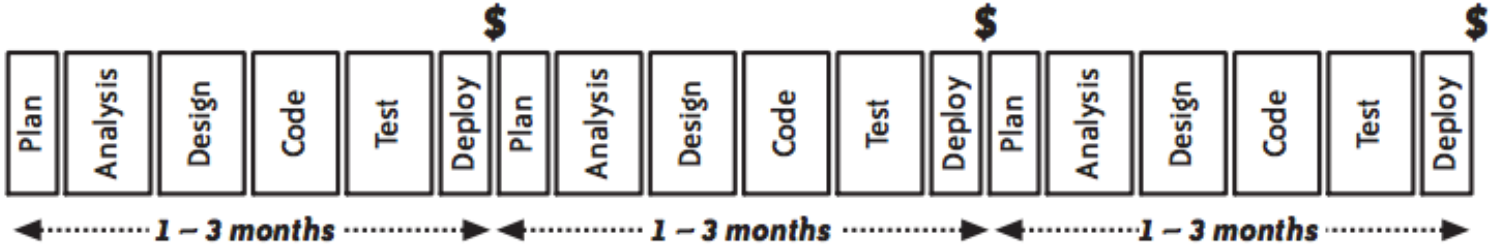
  - Cada modelo de proceso considera determinados horizontes

Múltiples niveles de planificación (II)

Imágenes extraídas de:The art of agile development. James Shore. 2008. Cap. 3

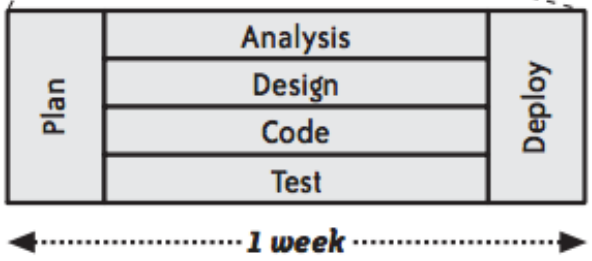


Modelo secuencial



Modelo iterativo

Modelo ágil (XP)



Esta aproximación no necesariamente implica una mayor productividad. Significa que el equipo tiene una retroalimentación mucho más frecuente. Como consecuencia el equipo fácilmente "relaciona" los éxitos y fallos con sus causas subyacentes. Es mucho menos costoso REPARAR LOS FALLOS

Alcance y efectividad de las pruebas

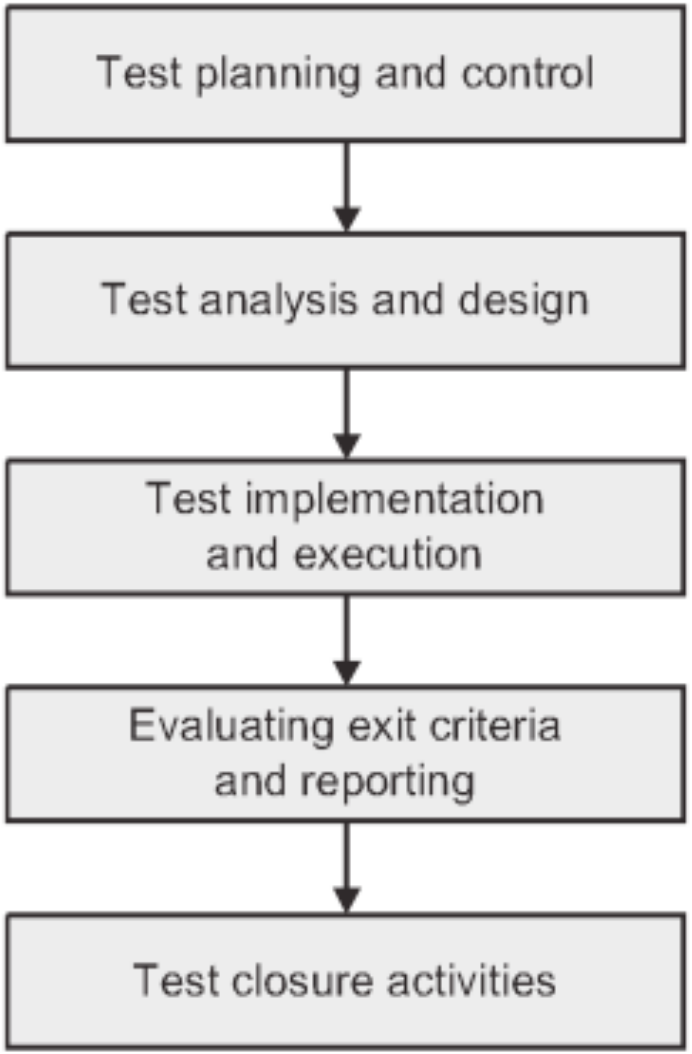
Para planificar las pruebas es importante tener en cuenta las siguientes cuestiones:

- ¿Cuántas pruebas son suficientes, y cómo decidir cuándo parar?
  - Factores para tomar la decisión: triángulo de recursos
- Para conseguir un resultado aceptable hay que:
  - Priorizar: decidir qué tests son más importantes
  - Fijar criterios para conseguir unos objetivos de pruebas de forma que sepamos cuándo parar. P.ej. qué áreas van a ser más probadas y con qué cobertura, qué nivel de defectos se van a tolerar en un producto entregado... (*completion criteria*)
- El objetivo final es asegurar que las pruebas son EFECTIVAS
  - Un buen test es aquél que encuentra un defecto. Si encontramos un defecto estamos creando una oportunidad de mejorar la calidad del producto
- El proceso de pruebas debe ser EFICIENTE:
  - Encontrar el mayor número de defectos con el menor número de pruebas posibles
  - Para ello se deben utilizar buenas técnicas de DISEÑO de casos de prueba

El proceso de pruebas

OBJETIVOS DEL PROCESO DE PRUEBAS

- DEMOSTRAR que el software satisface los requerimientos
- Descubrir situaciones en las que el comportamiento del software es incorrecto, indeseable, o no conforme a las especificaciones



- Se determina ¿QUÉ? se va a probar, ¿CÓMO?, ¿QUIÉN? y ¿CUÁNDO?
- Se determina QUÉ se va a hacer si los planes no se ajustan a la realidad

- Se determina qué CASOS DE PRUEBA se van a utilizar
- Un caso de prueba es un DATO CONCRETO + RESULTADO ESPERADO

- Se implementan y ejecutan los tests. Es la parte más visible, pero no es posible que los test sean efectivos sin realizar los pasos anteriores

- Se verifica que se alcanzan los *completion criteria* (p.ej 85% de cobertura) y se genera un informe

- En este punto las pruebas ya han finalizado. Se trata de asegurar que los informes están disponibles, ...

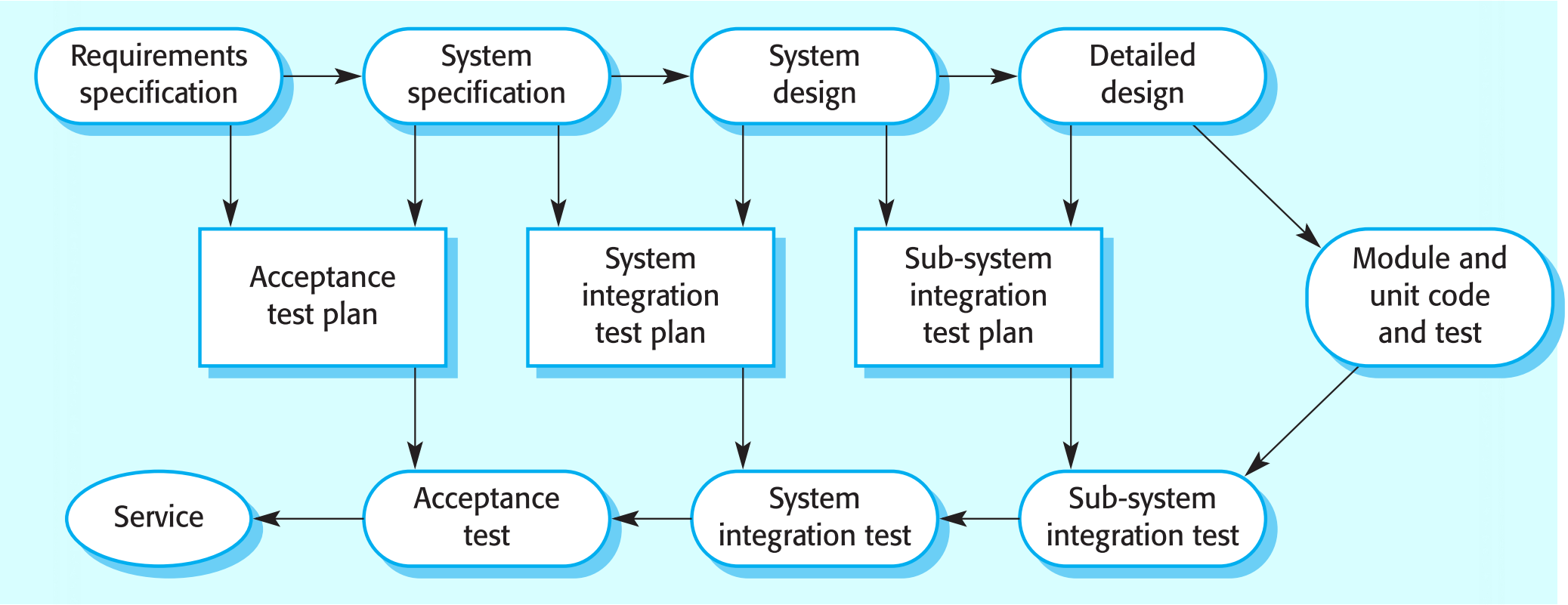
Niveles de pruebas

ver ISQTB Foundation Level Syllabus (2011)

- Pruebas de unidades (componentes)
  - Objetivo: encontrar defectos en el código de las unidades probadas
  - Una cuestión fundamental es "**aislar**" nuestra unidad del resto del código (independientemente de la granularidad de lo que definamos como unidad)
- Pruebas de integración
  - Objetivo: encontrar defectos derivados de la **interacción** entre las unidades, que previamente han sido probadas
  - La cuestión fundamental es el "**orden**" en el que vamos a realizar dicha integración
- Pruebas del sistema
  - Objetivo: encontrar defectos derivados del **comportamiento** del software como un **todo**
- Pruebas de aceptación
  - Objetivo: valorar en qué grado el software desarrollado **satisface** las **expectativas** del cliente

Planificación temporal de los niveles de pruebas

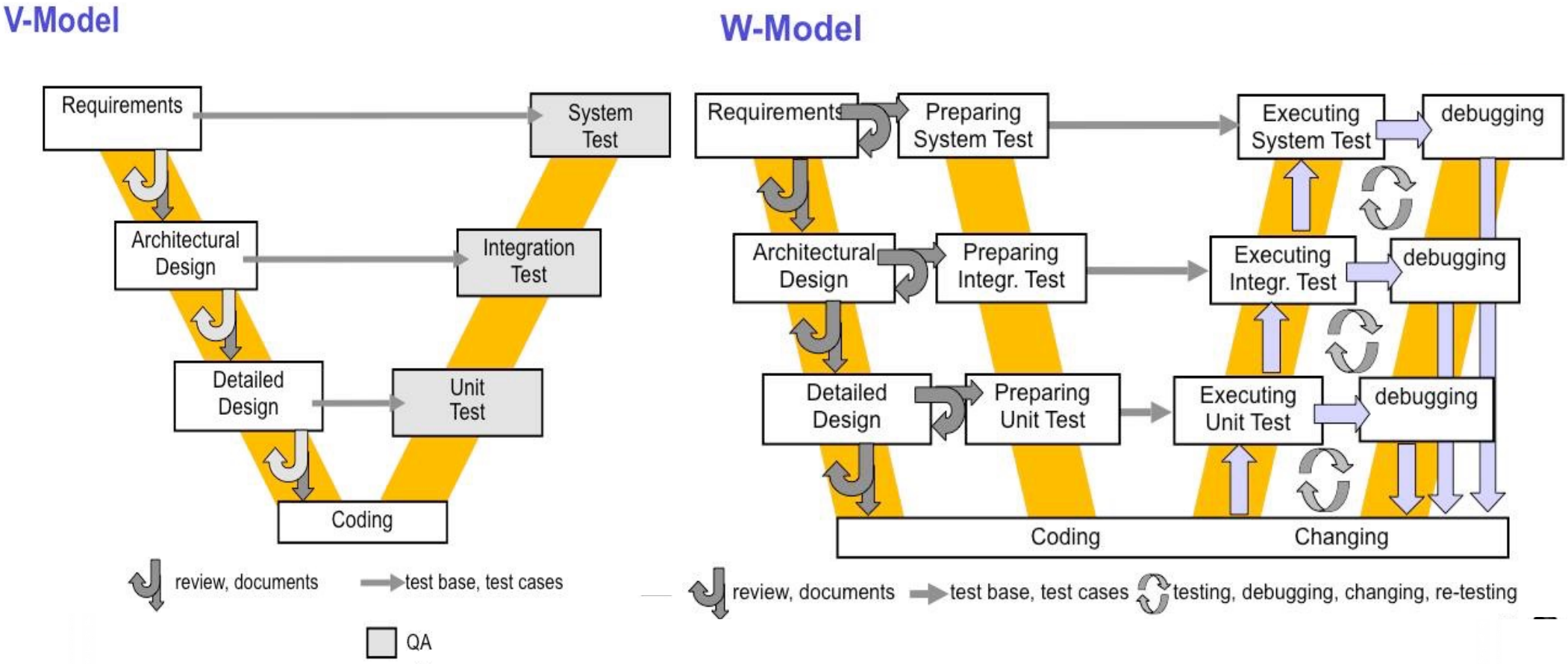
- Es importante conocer:
  - Cuándo se debe realizar cada tipo de prueba
  - Qué artefactos son necesarios para diseñar y ejecutar cada tipo de pruebas



Pruebas en modelos secuenciales

- Se trata de incluir en el plan de desarrollo las actividades de pruebas. Se debe contemplar un equilibrio entre las pruebas estáticas y las dinámicas

PLANIFICACIÓN DE PRUEBAS EN UN MODELO DE DESARROLLO SECUENCIAL

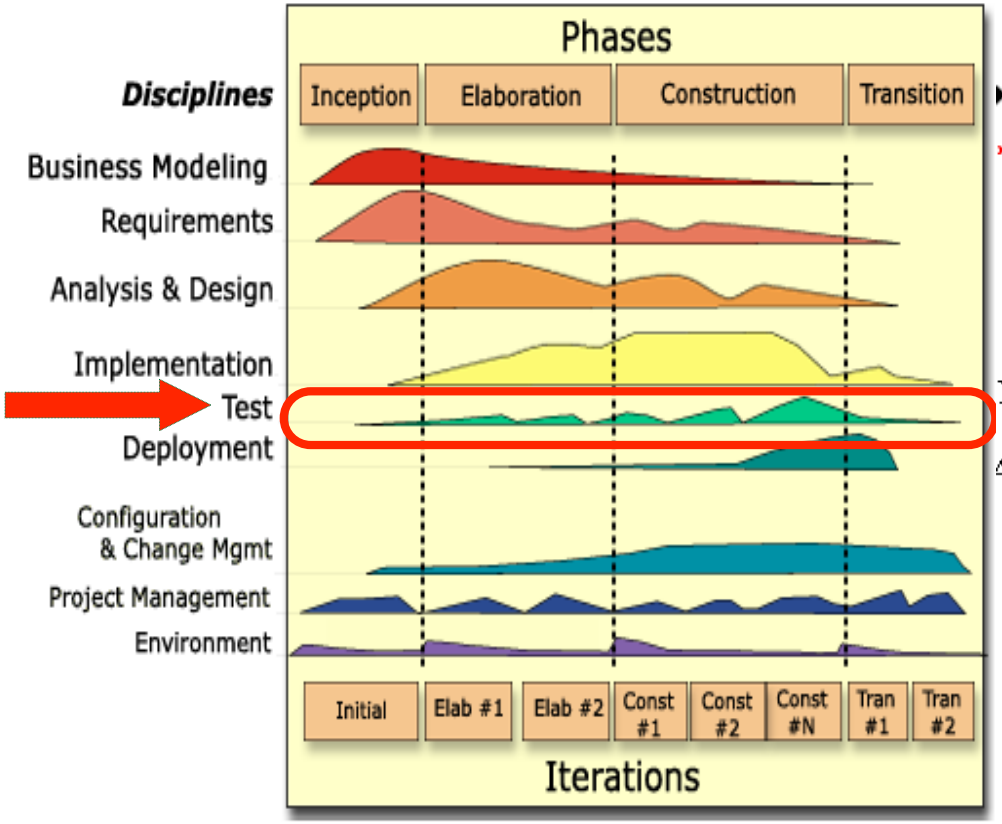


Pruebas en modelos iterativos y ágiles (I)

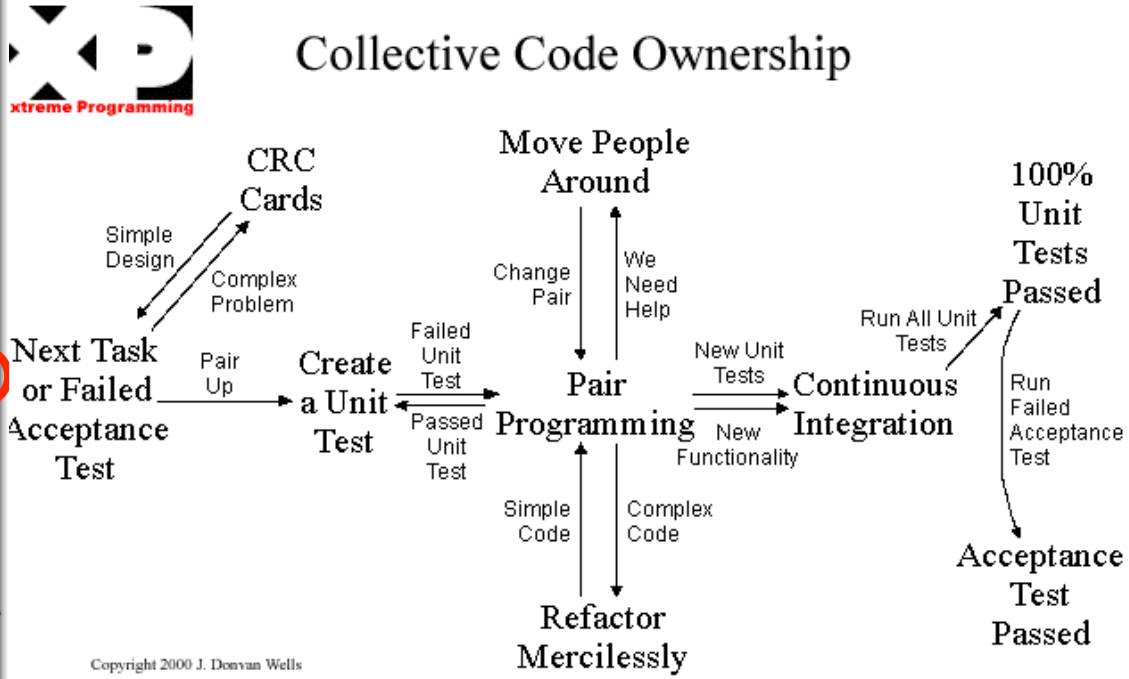
- En modelos **iterativos**, las pruebas se planifican a nivel de ITERACIÓN y RELEASE (una release está formada por un conjunto de iteraciones)

PRUEBAS EN MODELOS ITERATIVOS

Modelo UP



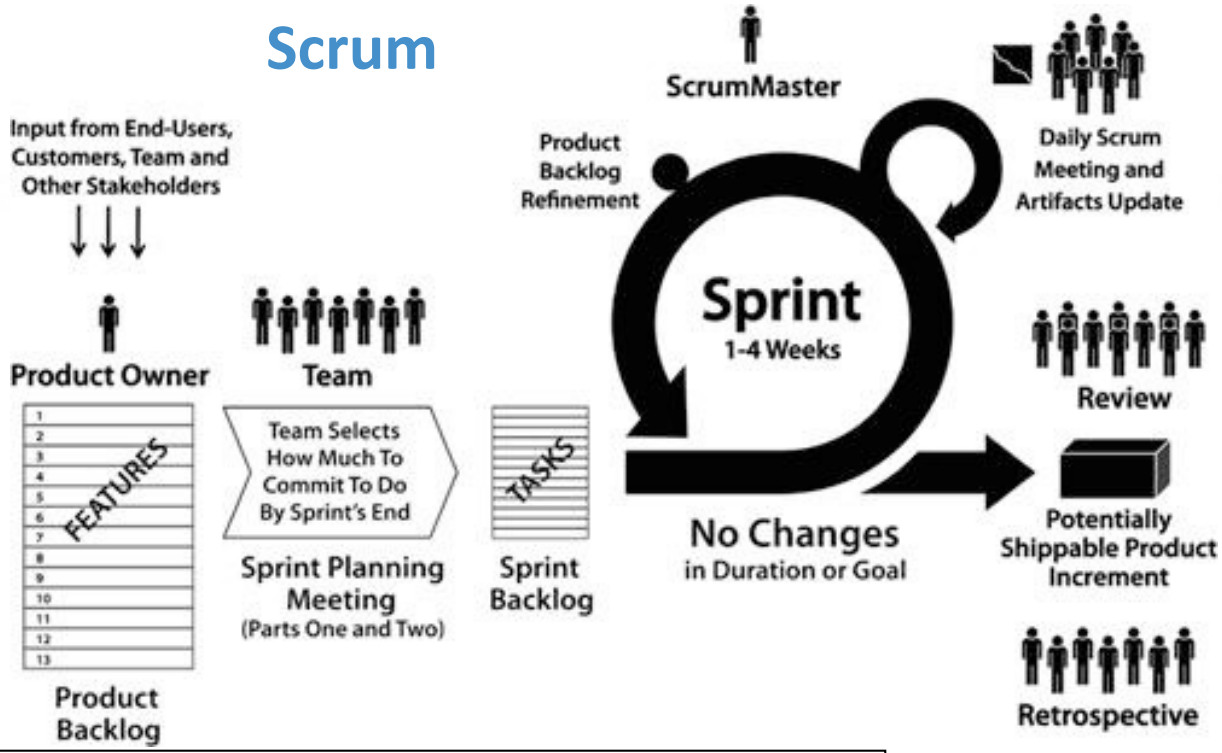
Modelo XP



En modelos **ágiles**, las pruebas se planifican a nivel de DÍA, ITERACIÓN y RELEASE

Pruebas en modelos iterativos y ágiles (II)

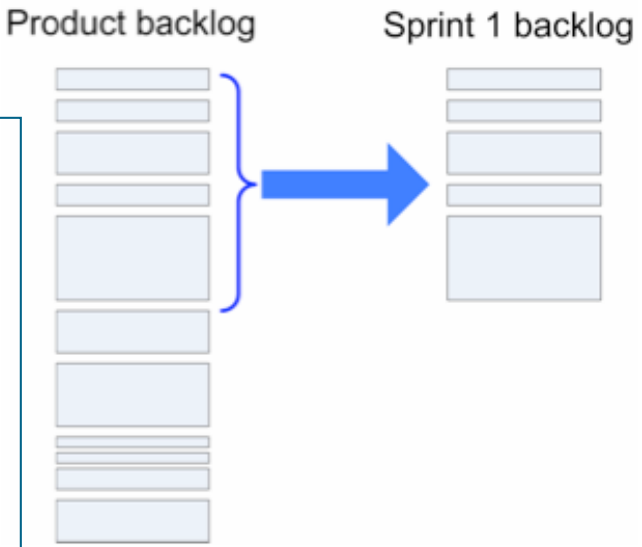
Scrum



El ***product owner*** es responsable de conseguir el máximo valor de negocio del producto o

El ***scrumMaster*** es el que actúa como guía del grupo durante el proceso, "protege" al grupo del exterior, y sirve como ayuda al mismo. NO es un gestor de proyectos

Product backlog: Lista de requerimientos priorizados por su valor de negocio (los valores más altos al principio de la lista).

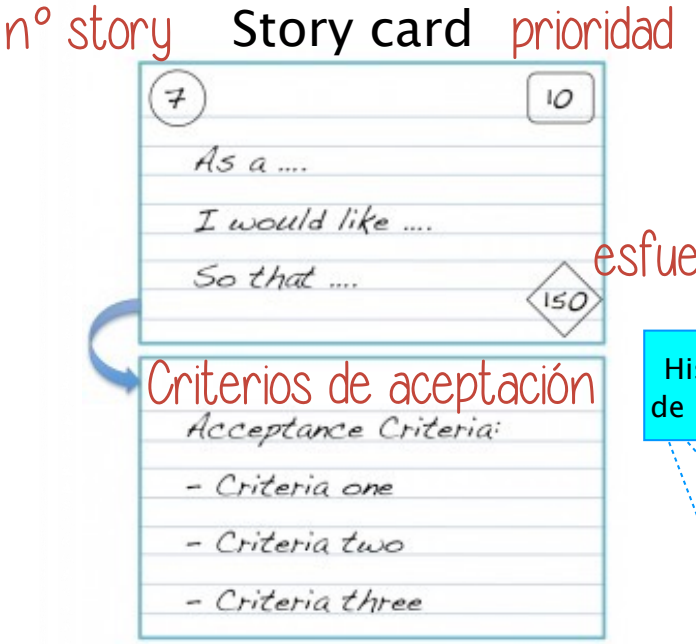


En un proceso Scrum NO hay una fase de testing "separada" del resto de actividades del desarrollo.

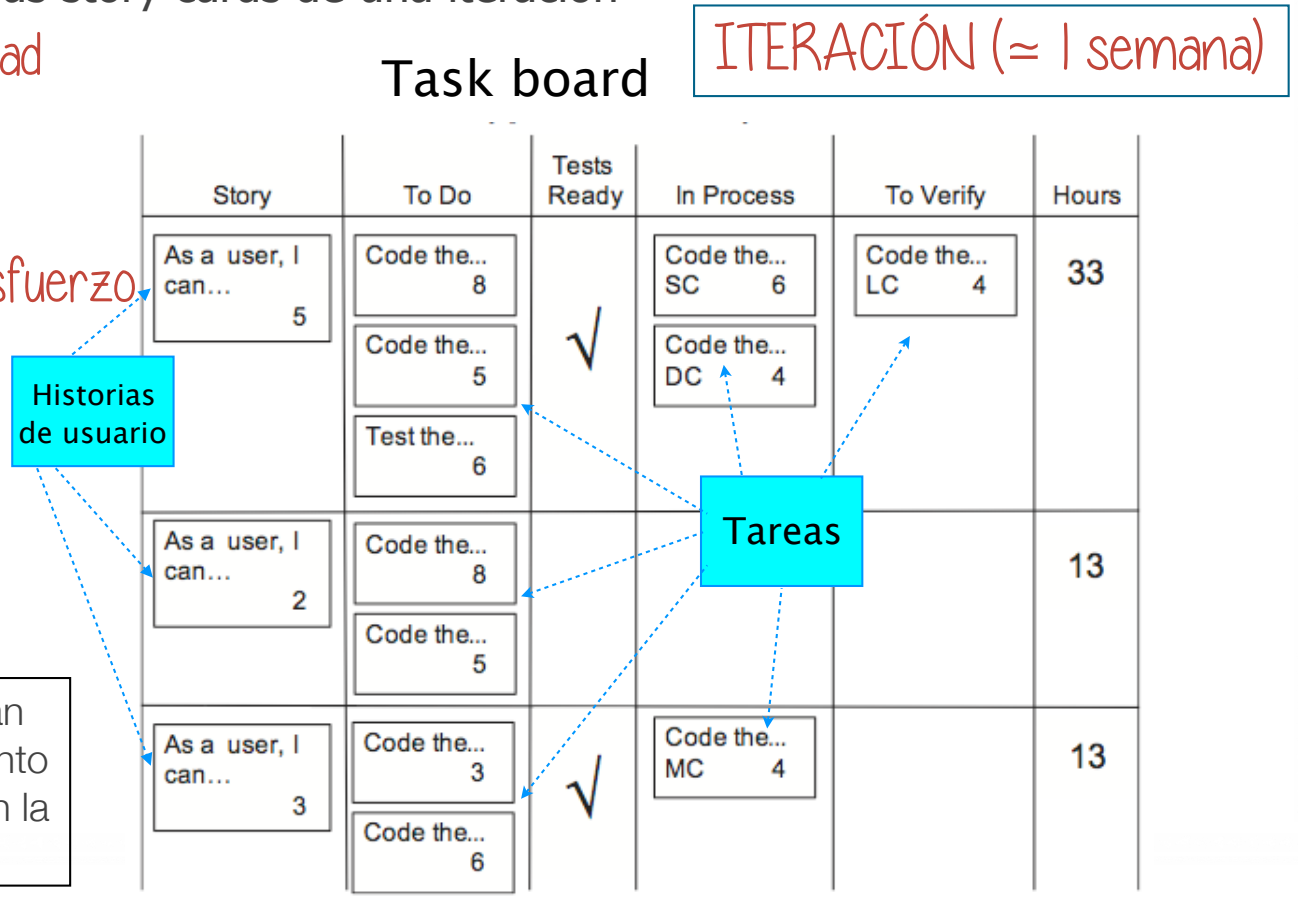
Cuando un desarrollador termina una historia de usuario, los tests tienen que estar preparados para su ejecución. Si el test pasa, la historia es aceptada y se pasa a la siguiente. Una vez que se han probado todas las historias y han pasado los tests, se da por concluido el sprint y se pasa al siguiente

¿Qué aspecto tiene un plan en XP? (I)

- Story cards: contienen historias de usuario. Representan características requeridas por el usuario (no casos de uso o escenarios). Se centran en el "beneficio" o resultado (value) que se pretende obtener. Deben describir un objetivo que permita a los testers evaluar una implementación exitosa de la misma. El formato suele ser:
  - `As an [actor] I want [action] so that [achievement]. For example: As a Facebook member, I want to set different privacy levels on my page so that I can control who sees my page.`
- Task list: lista de tareas para las story cards de una iteración



Las historias de usuario se priorizan según su valor de negocio. El conjunto de historias de usuario proporcionan la visión del producto.

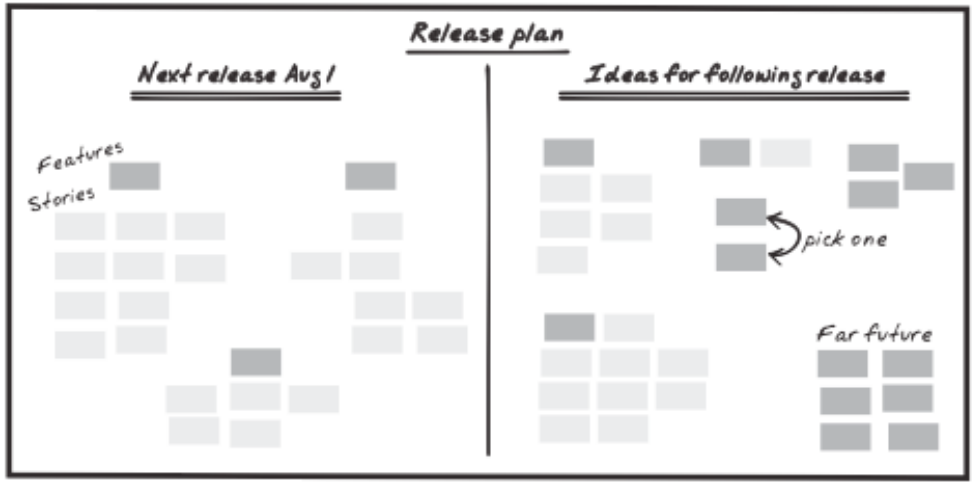


¿Qué aspecto tiene un plan en XP? (II)

Release plan board

El plan de entregas consiste en una la lista de historias de usuario priorizados por su valor de negocio (los valores más altos al principio de la lista). Se obtiene como resultado del proceso denominado *planning game*

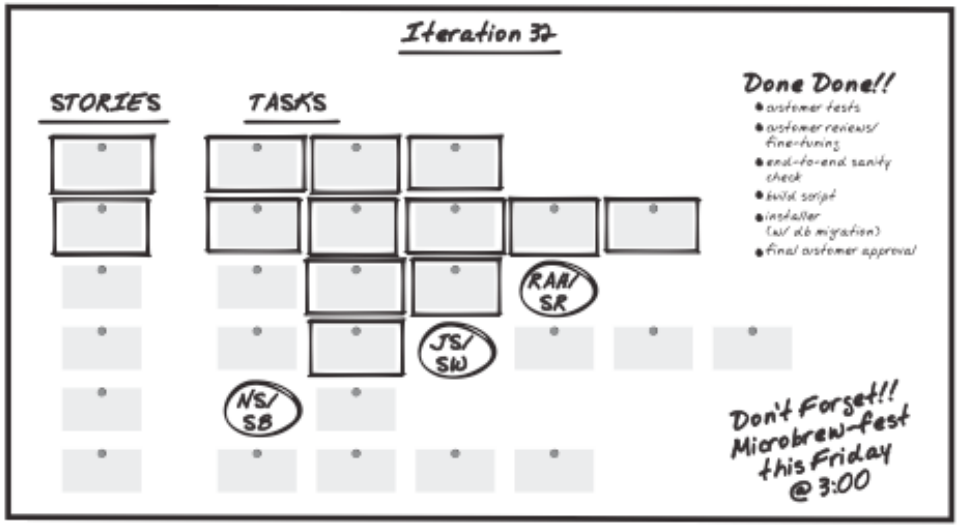
No se asignan recursos (¿quién?), ni tiempo (¿cúando?) hasta que comience cada iteración



Iteration plan board

Las historias de usuario se dividen en tareas concretas y son desarrolladas por los programadores. Cuando se empieza a trabajar en una tarea, el programador "pega" la tarjeta del tablero en su ordenador, dejando sus iniciales. Cuando la tarea se termina, se vuelve a colocar en el tablero y se marca en verde

Cada TASK tiene asociado un conjunto de TESTS (en el reverso de la tarjeta)



¿Qué aspecto tiene un plan en Scrum?

Scrum task board

SPRINT (3-4 semanas)

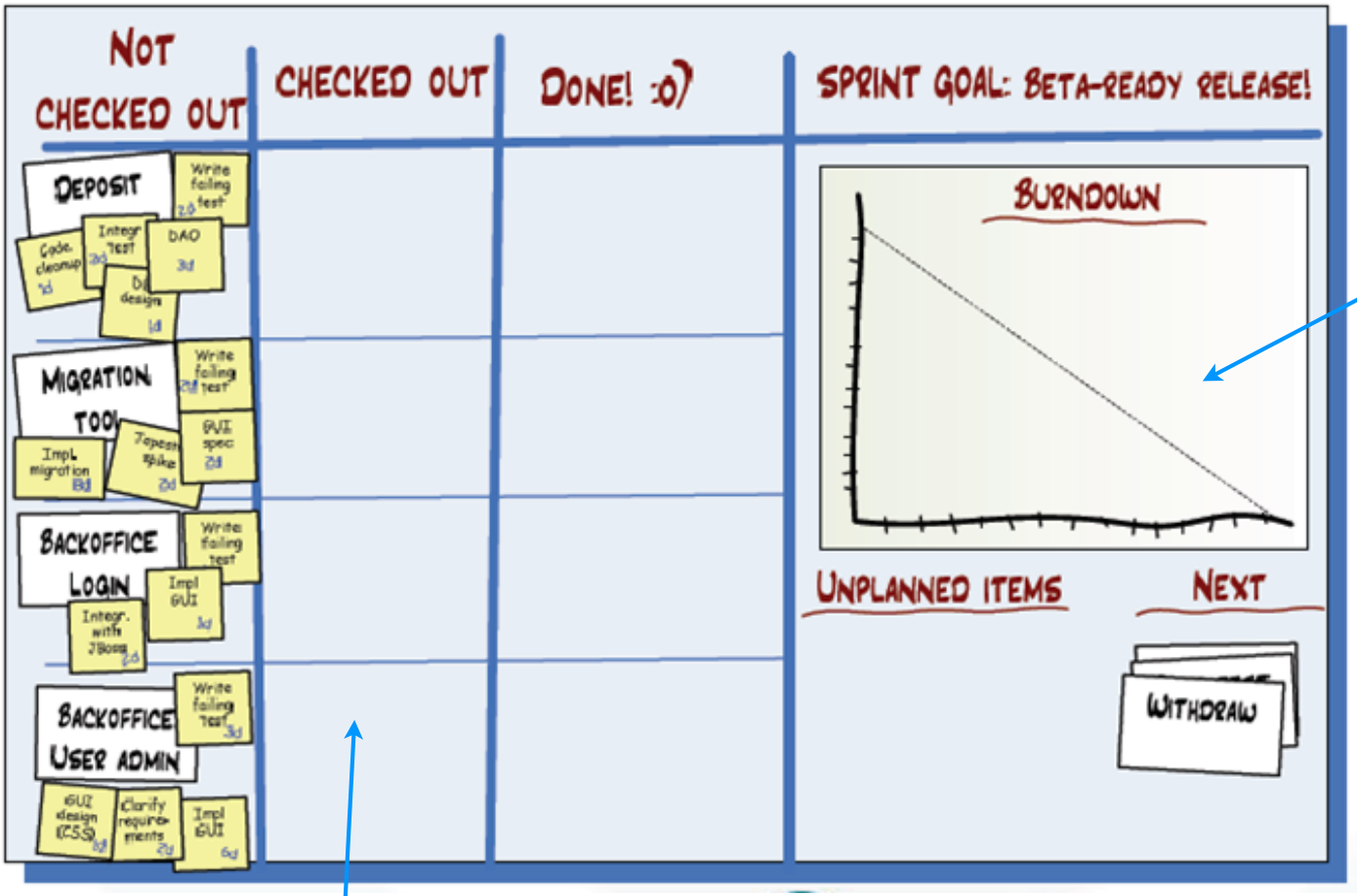


Gráfico de seguimiento

Pila sprint: Lista de historias de usuario + tareas

Tareas en proceso

Cada TASK tiene asociado un conjunto de TESTS (en el reverso de la tarjeta)

Integraciones continuas (CI)

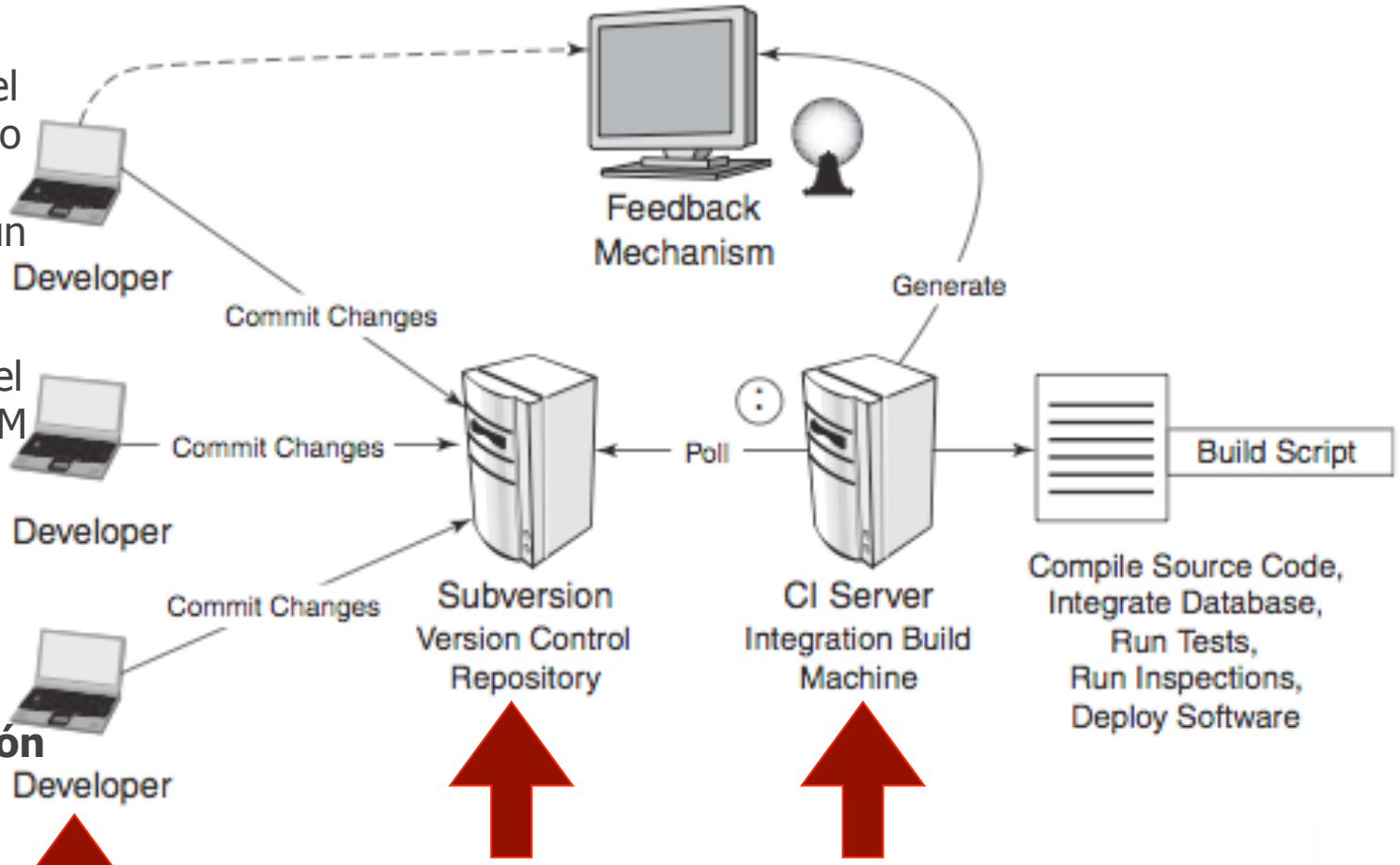
- El término "integración continua" nace con el proceso de desarrollo XP, como una de sus doce prácticas fundamentales
- Las integraciones continuas (**CI**: Continuous Integration) consisten en INTEGRAR el código del proyecto de forma ininterrumpida (en ciclos cortos) en una máquina aparte de la de cada desarrollador, la cual debe estar funcionando 24/7
- Si bien la práctica de CI no requiere de una herramienta específica, es habitual utilizar un Servidor de Integraciones Continuas para automatizar todo el proceso
- Las integraciones continuas realizan CONSTRUCCIONES PLANIFICADAS del sistema
  - Ejecutan el proceso de construcción (a partir de comandos Maven, por ejemplo) tantas veces como queramos y con la frecuencia que deseemos, sin mover ni un dedo.
  - Es importante que se ejecuten a intervalos regulares lo más cortos posible

Componentes y funcionamiento de un sistema de CI

Imagen extraída de: "Continuous integration". Paul M. Duval. Addison Wesley (Capítulo 1)

Funcionamiento:

1. Un desarrollador "sube" el código en el que ha estado trabajando al repositorio **SCM** (después de hacer un build local), y sólo si las pruebas unitarias han "pasado". Mientras tanto el servidor CI consulta el SCM para detectar cambios
2. El CI recupera la última versión del SCM y ejecuta un script de construcción del proyecto (**construcción planificada**)
3. El CI genera un feedback. En el momento en el que se "rompe" el sistema, hay que reparar el error



SCM: System Configuration Manager

Beneficios de las integraciones continuas

- Reducción de riesgos
  - La integración continua no evita los fallos, pero reduce drásticamente el esfuerzo de encontrar los errores y repararlos. Por ejemplo, supongamos que integramos el proyecto cada hora. Cada 60 minutos sabremos si nuestra construcción funciona o no. Esto hace que la búsqueda de errores sea más fácil (sólo hemos de mirar en los cambios que han ocurrido durante dicho intervalo). Además, estos problemas serán fáciles de resolver, porque en una hora no hemos tenido oportunidad de realizar grandes cambios que se habrían convertido en grandes problemas.
- Se agiliza el proceso de desarrollo mediante la automatización de las construcciones planificadas del sistema. La tarea de construir el sistema cada poco tiempo interfiere en el trabajo de los programadores (la construcción puede llevar desde unos pocos minutos a unas pocas horas). Un servidor de CI no tiene otra cosa mejor que hacer que construir el sistema, probarlo e informar de los resultados

Referencias bibliográficas

- Software Testing: An ISTQB-ISEB Foundation Guide. Brian Hambling. British Computer Society; 2nd New edition. 2010
  - Capítulo 1
- Agile estimating and planning. Mike Cohn. Prentice Hall. 2006
  - Capítulo 3
- The art of agile development. James Shore. O'Reilly. 2008
  - Capítulo 3
- The Scrum primer. An introduction to project manager with scrum. Pete Deemer. 2007
  - http://www.scrumprimer.com/
- Continuous integration. Martin Fowler. 2006
  - http://www.martinfowler.com/articles/continuousIntegration.html