

Ingeniería de Software

Desarrollo de Software Ágil

Sommerville capítulo 3

Temario

- 1) Métodos ágiles
- 2) Técnicas de desarrollo ágil
- 3) Gestión de proyectos ágiles
- 4) Escalando los métodos ágiles

Desarrollo de software “rápido”

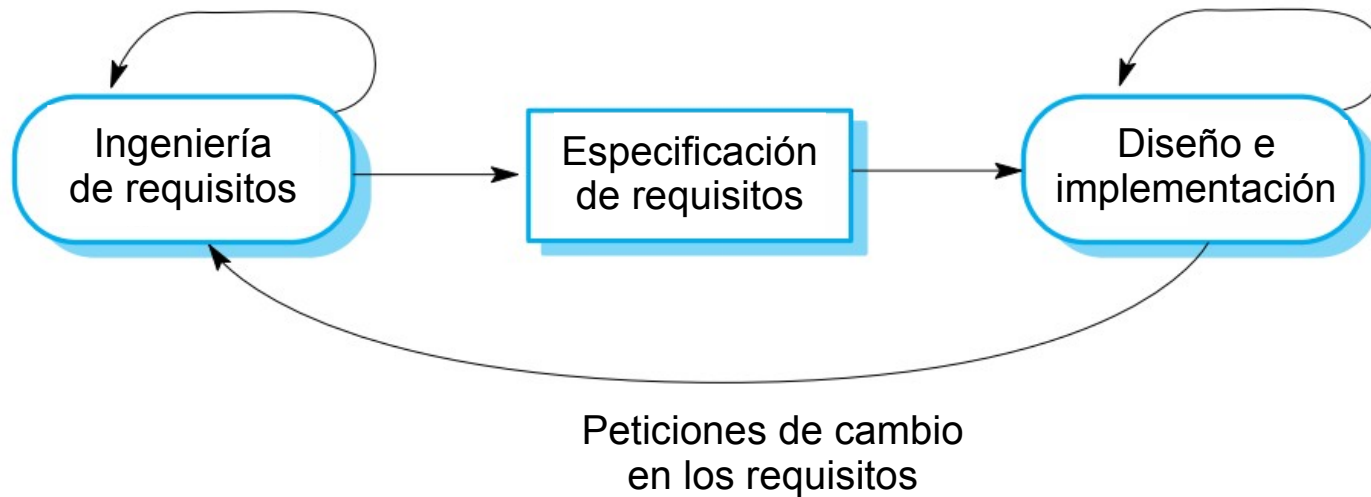
- Hoy en día uno de los más importantes requisitos para los sistemas de software es el desarrollo y entrega “**rápida**” de software.
 - Los negocios operan dentro de entornos con requisitos **altamente cambiantes**, en donde es prácticamente imposible generar un conjunto de requisitos estables.
 - El software debe evolucionar rápidamente para reflejar los cambios en las necesidades de negocio.
- El desarrollo guiado por planes es esencial para algunos tipos de sistema de software, pero no cumple dichas necesidades de negocio.
- Los métodos ágiles emergen a finales de los 90', cuyo objetivo se enfoca en reducir radicalmente el **tiempo de entrega de software funcionando**.

Desarrollo ágil

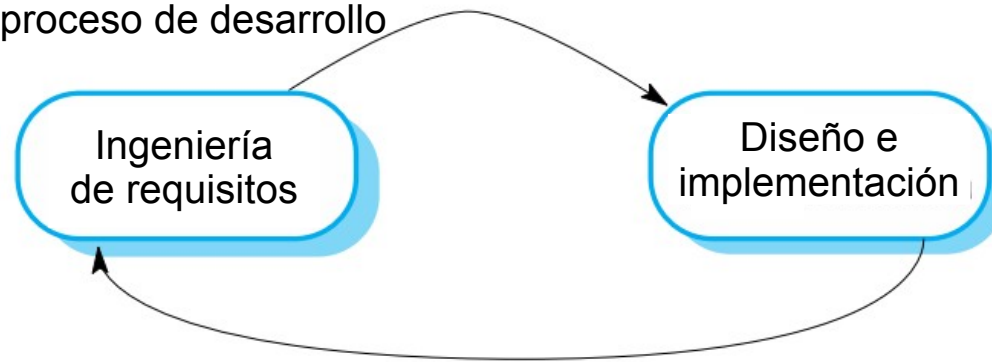
- Las actividades de especificación, diseño e implementación de los programas están intercaladas.
- El sistema es desarrollado a través de una serie de versiones o incrementos, en donde el cliente (y otros interesados) están involucrados en la especificación y evaluación de cada versión.
- Se realizan entregas frecuentes de nuevas versiones para evaluación
- El desarrollo de software es soportado por el uso extensivo de herramientas (por ejemplo: herramientas de testing automatizado)
- Documentación mínima – foco en código que funcione

Desarrollo ágil y guiado por planes

Desarrollo basado en planes: se planifica por adelantado cada etapa y los productos que deben producirse



Desarrollo ágil: se negocia lo que se va a producir durante el proceso de desarrollo



Manifiesto ágil

- “Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:
 - *Individuos e interacciones* sobre *procesos y herramientas*.
 - *Software funcionando* sobre *documentación extensiva*.
 - *Colaboración con el cliente* sobre *negociación contractual*.
 - *Respuesta ante el cambio* sobre *seguir un plan*.
- Esto es, aunque valoramos los **elementos de la derecha**, valoramos más los de la izquierda”.

Principios de los métodos ágiles

Principio	Descripción
Involucramiento del cliente	Los clientes deben estar estrechamente involucrados a lo largo del proceso de desarrollo. Su rol es proveer y priorizar nuevos requisitos del sistema y evaluar el resultado de las iteraciones.
Desarrollo incremental	El software es desarrollado en incrementos, en donde el cliente especifica qué requisitos deben ser incluidos en cada incremento.
Personas antes que los procesos	Las habilidades del equipo de desarrollo deben ser reconocidas y explotadas. Se debe permitir que los miembros del equipo desarrollen sus propias formas de trabajar, sin procesos prescriptivos.
Aceptar el cambio	Esperar que los requisitos cambien y por ende, modificar el sistema para acomodar dichos cambios.
Mantener la simplicidad	Centrarse en la simplicidad tanto del proceso de desarrollo como en el producto que se está desarrollando. Siempre que sea posible, trabajar activamente en eliminar la complejidad del sistema.

Principios del manifiesto ágil

- De deberes: <http://agilemanifesto.org/iso/es/principles.html>

Aplicabilidad de las metodologías ágiles

- Desarrollo de productos de pequeño o mediano porte
- Desarrollo de sistemas “a medida” (*custom systems*), en donde hay un claro compromiso del cliente en participar del proceso de desarrollo del software, en donde hay pocas reglas y no hay regulaciones externas que afecten al software.
- ¿Alguna otra? Van para el parcial y para el obligatorio...

El equipo ágil

“El desarrollo ágil se centra en los talentos y habilidades de los individuos, y adapta el proceso a personas y equipos específicos.”

<<Cockburn y Highsmith>>

Equipos auto-gestionados

- Equipos disciplinados con total autonomía.
- Se espera mayor responsabilidad a momento de cumplir con los compromisos que se han fijado.
- Se espera que tomen riesgos razonables y que aprendan a través del error y la introspección.
- La auto-organización no es una opción en Scrum; es un principio básico. Sin ella nunca tendremos equipos de alta performance.

Técnicas de desarrollo ágil

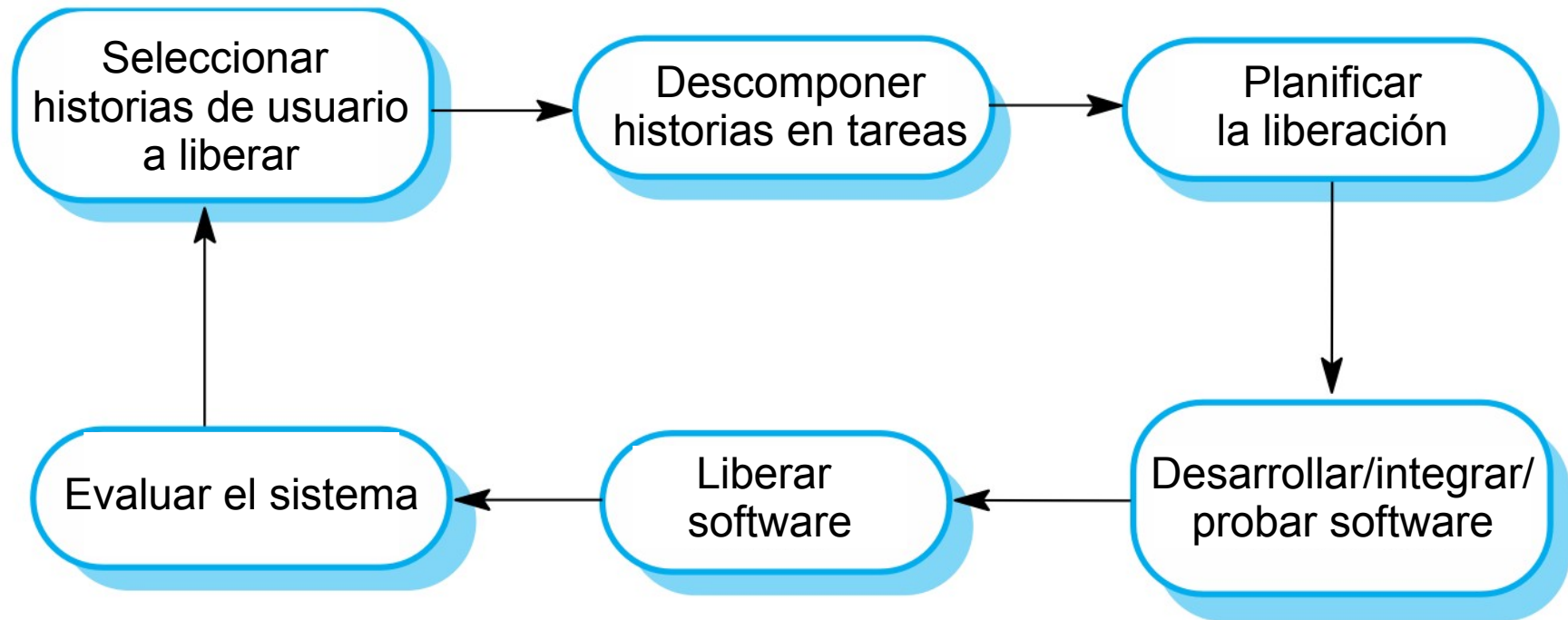
- *Extreme programing* (programación extrema)

¿Qué leyeron/conocen de las técnicas de Extreme Programing?

Extreme Programming

- Uno de los métodos ágiles con gran influencia, desarrollado a fines de los 90' el cual introduce varias técnicas de desarrollo ágil
- Extreme Programming (XP) toma en “extremo” el enfoque de desarrollo iterativo:
 - Nuevas versiones (builds) pueden generarse varias veces por día (integración continua)
 - Los incrementos son entregados a los clientes cada 2 semanas
 - Todas las pruebas deben ser ejecutadas para cada versión, y la versión es aceptada únicamente si todas las pruebas se ejecutan satisfactoriamente

Ciclo de liberación de XP



Prácticas de XP (1)

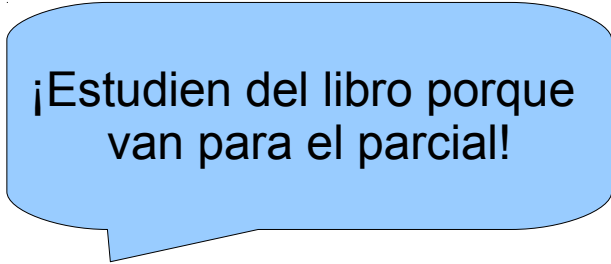
Principio o práctica	Descripción
Planificación incremental	Los requisitos que serán incluidos en una liberación serán determinados por el tiempo disponible y su prioridad relativa. Los desarrolladores dividirán la especificación del requisito en “tareas” de desarrollo.
Liberaciones pequeñas	se debe desarrollar primero el conjunto de funcionalidades que sea “mínimo y útil” y que dé valor de negocio. Las liberaciones del sistema deben realizarse con frecuencia, agregando funcionalidad a la primer versión liberada.
Diseño simple	Diseño suficiente para cumplir los requisitos actuales y no más.
<i>Desarrollo dirigido por pruebas (Test-first development o TDD - Test driven development)</i>	Se utiliza un framework de pruebas unitarias automatizadas para generar las pruebas de una nueva funcionalidad, antes que la funcionalidad esté implementada.
Refactorización	Se espera que todos los desarrolladores realicen refactorización al código de forma continua, tan pronto como se encuentren posibles mejoras al mismo. Esto mantiene el código simple y mantenible.

Prácticas de XP (2)

Programación por pares	Los desarrolladores trabajan de a pares, revisándose el trabajo mutuamente y proporcionando soporte para realizar siempre un buen trabajo
Propiedad colectiva	Los “pares” de desarrolladores trabajan en todas las áreas del sistema, por lo que no se generan islas de experiencia. Además, los desarrolladores toman responsabilidad por todo el código. Cualquiera puede cambiar cualquier cosa
Integración continua	No bien una tarea es terminada, se debe integrar con todo el sistema. Luego de cada integración, todas las pruebas unitarias deben ser ejecutadas con resultados exitosos
Ritmo sostenible	Gran cantidad de horas extras son consideradas inaceptables. El efecto neto a menudo reduce la calidad del código y a mediano plazo la productividad
Cliente	Una cantidad representativa de los usuarios del sistema (o el cliente) deben estar disponible para el equipo de XP. En un proceso XP, el cliente es un miembro del equipo de desarrollo y es el responsable de brindar al equipo los requerimientos del sistema a ser implementados.

Prácticas influyentes de XP en metodologías ágiles

- XP tiene un foco técnico y no es fácil de integrar con prácticas de gestión en la mayoría de las organizaciones.
 - Como consecuencia, si bien el desarrollo ágil utiliza prácticas de XP, no es utilizado ampliamente como fue definido en sus orígenes.
- Prácticas claves
 - Historias de usuario para la especificación (las van a ver más en detalle en el tema “Ingeniería de Requisitos)
 - Refactorización
 - Desarrollo dirigido por pruebas
 - Programación por pares



¡Estudien del libro porque van para el parcial!

Gestión de proyectos ágiles

- La principal responsabilidad de los líderes o jefes de proyecto es gestionar el proyecto de forma tal de que el software sea liberado en fecha y dentro del presupuesto planificado para el mismo.
- El enfoque estándar para un líder de proyecto es el “dirigido por planes”. Los líderes trazan un plan para el proyecto, el cual muestra qué va a ser entregado, cuándo va a ser entregado y quiénes van a trabajar en dichas entregas.
- La gestión de proyectos ágiles requieren un enfoque ligeramente diferente, el cual se adapte al desarrollo incremental y a las prácticas usadas en los métodos ágiles.

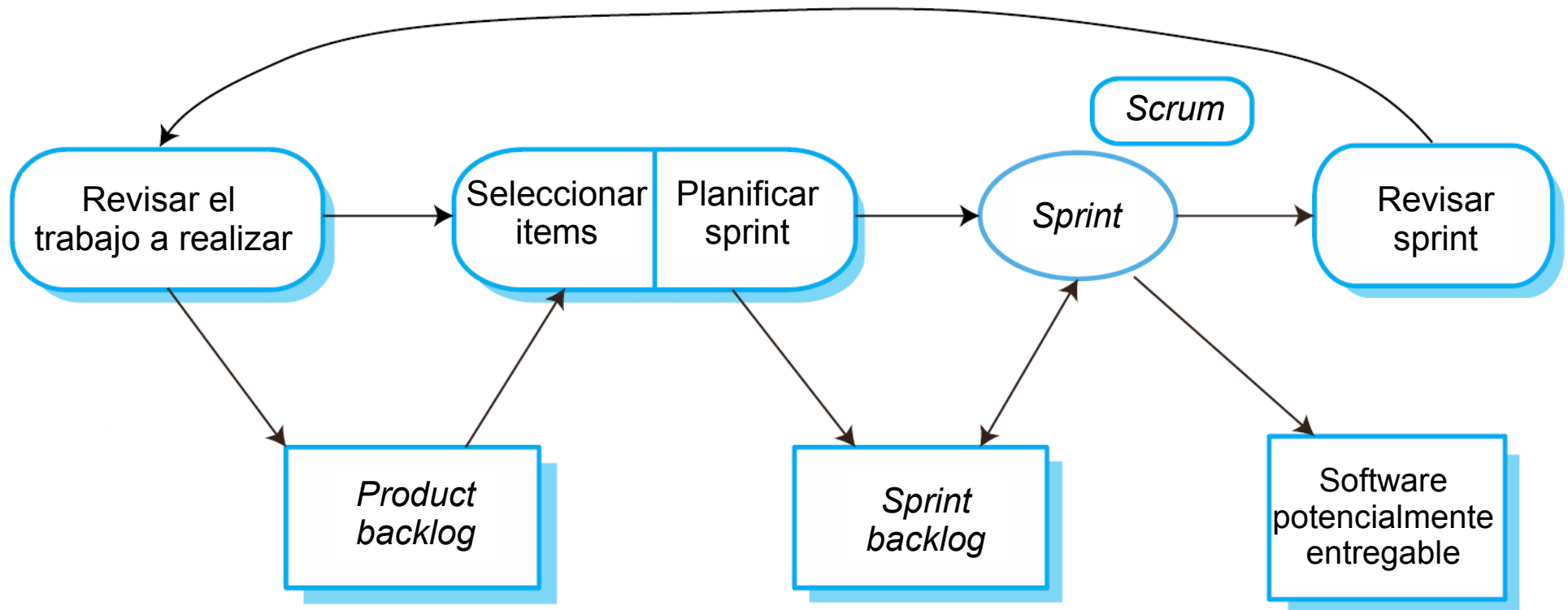
Scrum

- Scrum es un método ágil que se enfoca en la gestión del desarrollo iterativo, más que en las prácticas ágiles específicas
- Comprende tres fases
 - **La fase inicial:** se genera un esquema de la planificación en donde se establecen los objetivos generales del proyecto y el diseño de la arquitectura del software
 - **Ciclos de sprints:** la fase inicial es seguida de una serie de iteraciones, en donde se desarrolla un incremento del sistema en cada una.
 - **Fase de clausura:** en esta fase se cierra el proyecto, se completa la documentación requerida y se evalúan las lecciones aprendidas del mismo.

Elementos de Scrum

- Equipo de desarrollo
- Incremento del producto (potencialmente entregable)
- *Product Backlog*
- *Product Owner*
- *Scrum meeting*
- *Scrum Master*
- *Sprint*
- *Velocity*

Ciclos de sprints en Scrum



Equipo de trabajo en Scrum

- El '*Scrum master*' es un facilitador, quien concilia las reuniones diarias, gestiona el backlog de trabajo a ser realizado, registra las decisiones tomadas, mide el progreso del proyecto contra el backlog y se comunica con clientes y gerencias externas al proyecto.
- Todo el equipo asiste a una corta reunión diaria (*daily meeting*) en donde todo el equipo comparte información, describe su progreso desde la última reunión, problemas que hayan aparecido y lo que planifica para el día siguiente.
- Esto implica que todos en el equipo conocen qué está pasando. En caso de aparecer problemas, pueden re-planificar a corto plazo para abordarlos.

Beneficios de Scrum

- El producto es dividido en partes fácilmente entendibles y gestionables.
- Los requisitos inestables no truncan el progreso.
- Todo el equipo tiene visibilidad de todo y consecuentemente se mejora la comunicación entre los integrantes
- Los clientes reciben a tiempo las entregas de los incrementos y se gana *feedback* en cómo el producto funciona (y debería funcionar).
- Se establece una confianza entre el cliente y el equipo, en donde se crea una cultura positiva en la cual cada uno espera que el proyecto tenga éxito.

Escalando los métodos ágiles

- Los métodos ágiles han probado ser efectivos para ciertos tipos de proyectos: productos pequeños-medianos, equipos reducidos y ubicados en el mismo lugar físico, con ciertas habilidades, etc.
- “Escalar hacia arriba” refiere a la utilización de métodos ágiles para el desarrollo de grandes sistemas de software, los cuales no pueden ser desarrollados por equipos reducidos.
- “Escalar hacia afuera” refiere a cómo los métodos ágiles pueden ser introducidos a lo largo de una gran organización con muchos años de experiencia en el desarrollo de software
- Cuando se escalan los métodos ágiles es importante mantener:
 - Planificación flexible, liberaciones frecuentes y buena comunicación de equipo.

Problemas al escalar

- La informalidad de los métodos ágiles es incompatible con la definición de “contrato” comúnmente usada en grandes compañías.
- Los métodos ágiles son más apropiados para desarrollo de nuevos productos más que para el mantenimiento de los mismos. La mayor parte de los costos en las grandes compañías refieren a mantener sus sistemas de software ya existentes.
- Los métodos ágiles están diseñados para pequeños equipos que están ubicados en el mismo lugar físico. Grandes compañías desarrollan software con equipos distribuidos a lo largo de todo el mundo.

Problemas con el mantenimiento ágil

- La mayoría de las organizaciones invierten más tiempo en mantener software existente que en desarrollar nuevo software.
- Problemas clásicos:
 - Falta de documentación del producto.
 - Mantener a los clientes involucrados en el proceso de desarrollo.
 - Mantener la continuidad del equipo de desarrollo.
- El desarrollo ágil confía en que el equipo de desarrollo sabe y entiende qué es lo que se debe hacer
- Para sistemas “larga vida” un gran problema es que los desarrolladores originales no siempre trabajan en el mantenimiento.

Enfoques dirigidos por planes y ágiles

- La mayoría de los proyectos incluyen elementos de **ambos** enfoques, la decisión en el “balance” entre ambos depende de:
 - ¿Es importante tener los requisitos y el diseño detallados antes de implementar?
 - ¿Es factible obtener feedback del cliente con entregas rápidas?
 - ¿Qué tan grande es el sistema a ser implementado? ¿de qué tipo? ¿tiempo de vida esperado? ¿regido por alguna regulación en particular?
 - ¿Es muy difícil priorizar los cambios? ¿hay múltiples clientes/interesados?
 - ¿Se necesita saber las funcionalidades con anterioridad por el departamento de marketing?
 - ¿Qué tan adaptados a la presión sobre las entregas están los miembros del equipo?
 - ¿Qué tan buenos son los programadores y diseñadores del equipo?
 - ¿Cómo está organizado el equipo?
 - ¿Qué tecnologías hay disponibles?

Temas a estudiar en casa...

- Propuestas de escalamiento de metodologías ágiles
 - *IBM's agility at scale model*
 - *Multi-team Scrum*