INGENIERÍA DE APLICACIONES

Desarrollo Ágil de Software

Dra. María Luján Ganuza
mlg@cs.uns.edu.ar
DCIC - Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2019

Temario

Desarrollo Ágil de Software.

- Características.
- Principios.
- Desarrollo ágil vs. Desarrollo impulsado por plan.
- Enfoques:
  - Programación Extrema
  - Enfoque Scrum

Desarrollo Ágil de Software

- Los métodos ágiles de desarrollo de software son **enfoques iterativos** donde el software se desarrolla y se entrega a los clientes en **incrementos**.
- A diferencia del enfoque impulsado por plan, la funcionalidad de estos incrementos **no se planifica de antemano**, sino que se decide durante el desarrollo del proyecto.
- La decisión sobre qué incluir en un incremento depende del progreso y de las prioridades del cliente.
- El argumento para este enfoque es que las prioridades y los requisitos del cliente cambian por lo que tiene sentido tener un plan flexible que pueda acomodar estos cambios.

Desarrollo Ágil de Software

- Las empresas ahora operan en un entorno global que cambia rápidamente.
- Tienen que responder a las nuevas oportunidades y mercados y a la aparición de productos y servicios competitivos.
- El **nuevo software se desarrolla rápidamente** para aprovechar las nuevas oportunidades y para responder a la presión competitiva

Desarrollo Ágil de Software

- Ante un entorno tan cambiante, **los requisitos iniciales cambian inevitablemente**.
- Es posible, que solo después de entregado el sistema, y que los usuarios adquieran experiencia con el mismo, que los requisitos reales se vuelvan claros.
- Incluso, es probable que los requisitos cambien de forma rápida e impredecible debido a factores externos.
- El software puede estar desactualizado cuando se entregue.

Desarrollo Ágil de Software

- Un enfoque **impulsado por plan** es el correcto para algunos tipos de software, donde un análisis completo del sistema es esencial.
- Sin embargo, en un entorno empresarial en rápido movimiento, esto puede causar problemas reales.
- Cuando el software está disponible para su uso, la razón original de su adquisición puede haber cambiado tan radicalmente que el software es efectivamente inútil.
- Por lo tanto, para este tipo de sistemas es esencial un **desarrollo rápido** que sea **capaz de manejar cambios en los requerimientos**.

Desarrollo Ágil de Software

- Los **procesos rápidos** de desarrollo de software están diseñados para producir software útil con rapidez.
- El software no se desarrolla como una sola unidad, sino como una serie de incrementos, y con cada incremento, se incluyen nuevas funcionalidades.
- Involucran a los clientes en el proceso de desarrollo para obtener una respuesta rápida sobre los requisitos cambiantes.
- Minimizan la documentación por utilizando comunicaciones informales.

Desarrollo Ágil de Software

- Existen varios enfoques, pero todos comparten estas características fundamentales:
  - Los procesos de especificación, diseño e implementación están intercalados.
  - El sistema se desarrolla en una serie de **versiones**.
  - Las interfaces de usuario se desarrollan a menudo usando un sistema de desarrollo interactivo.

Principios de los Métodos Ágiles.

1. Participación del Cliente
2. Entrega Incremental
3. Personas, no procesos.
4. Aceptar el cambio.
5. Mantener la simplicidad.

Principios de los Métodos Ágiles.

1. **Participación del Cliente**
   Los clientes deben participar activamente en todo el proceso de desarrollo. Su función es proporcionar y priorizar nuevos requisitos del sistema y evaluar las iteraciones del sistema.
2. Entrega Incremental
3. Personas, no procesos.
4. Aceptar el cambio.
5. Mantener la simplicidad.

Principios de los Métodos Ágiles.

1. Participación del Cliente
2. **Entrega Incremental**
   El software se desarrolla en incrementos. El cliente especifica los requisitos para ser incluidos en cada incremento.
3. Personas, no procesos.
4. Aceptar el cambio.
5. Mantener la simplicidad.

Principios de los Métodos Ágiles.

1. Participación del Cliente
2. Entrega Incremental
3. **Personas, no procesos.**
   Los miembros del equipo deben ser libres de desarrollar en sus propias formas de trabajar sin procesos prescriptivos.
4. Aceptar el cambio.
5. Mantener la simplicidad.

Principios de los Métodos Ágiles.

1. Participación del Cliente
2. Entrega Incremental
3. Personas, no procesos.
4. **Aceptar el cambio.**
   Esperar que los requisitos del sistema cambien y diseñar el sistema para adaptarse a estos cambios.
5. Mantener la simplicidad

Principios de los Métodos Ágiles.

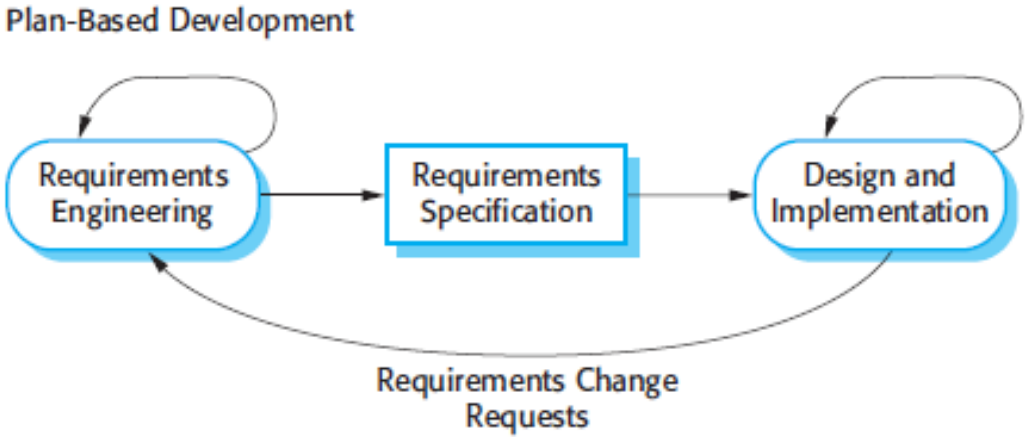
1. Participación del Cliente
2. Entrega Incremental
3. Personas, no procesos.
4. Aceptar el cambio.
5. **Mantener la simplicidad.**
   Centrarse en la simplicidad tanto en el software que se está desarrollando como en el proceso de desarrollo.

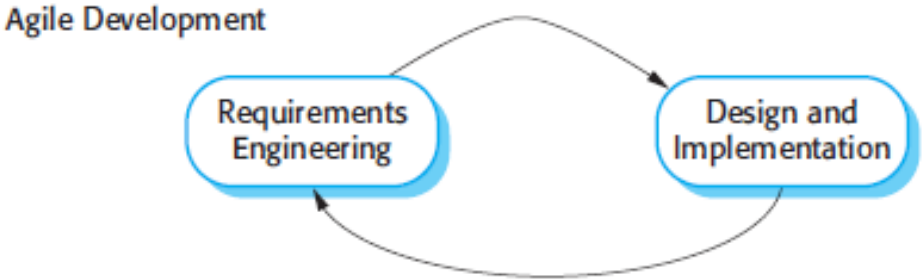
Principios de los Métodos Ágiles.

Dificultades a la hora de aplicar estos principios:

- Disponibilidad baja del cliente.
- La personalidad de los miembros del equipo pueden no ser adecuadas para la participación intensa típica de los métodos ágiles y la interacción con otros miembros.
- Si hay muchas partes interesadas, priorizar los cambios puede ser extremadamente difícil. Cada parte interesada da diferentes prioridades a diferentes cambios.
- Mantener la simplicidad requiere trabajo adicional.

Desarrollo Ágil vs. Impulsado por Plan





¿Desarrollo Ágil o Impulsado por Plan?

- De hecho, la mayoría de los proyectos de software incluyen prácticas basadas en enfoques impulsados por planes y enfoques de desarrollo ágiles.
- Decidir sobre el equilibrio entre estos enfoques debe responder a una serie de preguntas técnicas, humanas y organizativas.

¿Desarrollo Ágil o Impulsado por Plan?

- ¿Es importante tener una especificación y diseño muy detallados antes de pasar a la implementación?
- ¿Es una estrategia de entrega incremental realista?
- ¿Qué tan grande es el sistema que se está desarrollando?
- ¿Cuál es la duración esperada del desarrollo del sistema?
- ¿Cómo está organizado el equipo de desarrollo?

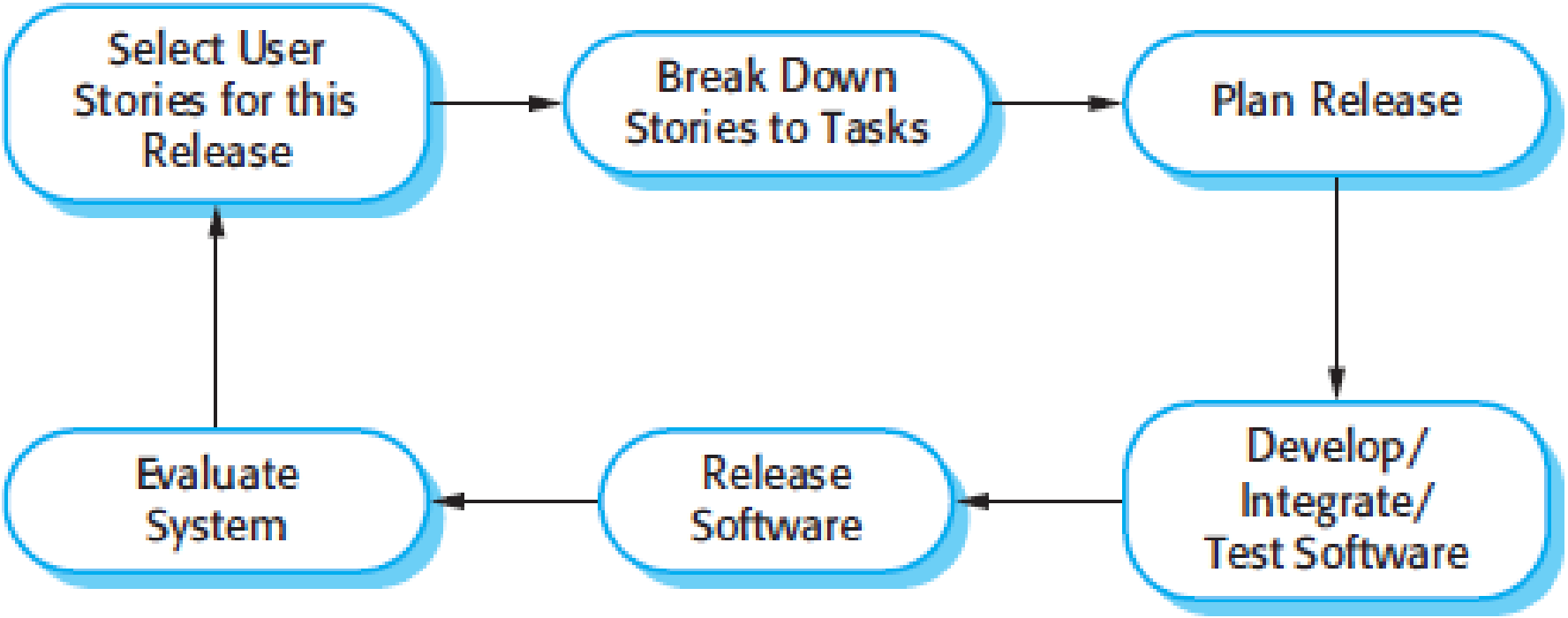
Enfoques del Desarrollo Ágil

- Programación Extrema (XP)
- Enfoque Scrum

Programación Extrema (XP)

- El nombre fue acuñado por Beck (2000) porque el enfoque se basó en impulsar buenas prácticas reconocidas, como el desarrollo iterativo, a niveles "extremos".
- En XP los requisitos se expresan como escenarios, que se implementan directamente como una serie de tareas.
- Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código.
- Todas las pruebas deben ser ejecutadas exitosamente cuando el nuevo código está integrado en el sistema.

Programación Extrema (XP)



Prácticas de la XP

- Planificación Incremental
- Versiones pequeñas
- Diseño simple
- Desarrollo “test-first”
- Refactorización
- Programación en pares
- Propiedad colectiva
- Integración continua
- Ritmo sostenible
- Cliente “on-site”

Prácticas de la XP

- **Planificación Incremental:** Los requisitos de planificación incremental se registran en “Story Cards” y las historias que se incluirán en una están determinadas por el tiempo disponible y su prioridad relativa. Los desarrolladores descomponen estas Historias en 'Tareas' de desarrollo.
- **Versiones pequeñas:** Se desarrolla primero el conjunto mínimo útil de funcionalidad. Las versiones agregan gradualmente funcionalidad.
- **Diseño simple:** Se diseña lo mínimo e indispensable como para cumplir con los requisitos actuales y no más.
- **Desarrollo “test first”:** Se escriben pruebas para una funcionalidad antes de que la misma se implemente.
- **Refactorización:** Se espera que todos los desarrolladores refaccionen el código continuamente, manteniendo el código simple.

Prácticas de la XP

- **Programación de pares** Los desarrolladores trabajan en pares, revisando el trabajo de los demás y brindando soporte.
- **Propiedad colectiva** Los pares de desarrolladores trabajan en todas las áreas del sistema. Todos se hacen responsables de todo el código.
- **Integración continua** Tan pronto como se completa el trabajo en una tarea, se integra en todo el sistema. Después de tal integración, todas las pruebas unitarias en el sistema deben pasar.
- **Ritmo sostenible** No se consideran aceptables grandes cantidades de horas extra, ya que puede reducir la calidad del código y la productividad a medio plazo.
- **Cliente en el sitio** El cliente debe estar disponible a tiempo completo, es considerado un miembro del equipo de desarrollo.

Prácticas de la XP

¿Cómo reflejan estas prácticas los principios de la programación ágil?

1. Participación del Cliente.
2. Entrega Incremental.
3. Personas, no procesos.
4. Aceptar el cambio.
5. Mantener la simplicidad.

Degradación de código en XP

- Un problema general con el desarrollo incremental es que tiende a degradar el estructura del software, los cambios en el software se vuelven cada vez más difíciles de implementar (código duplicado, reutilización inapropiada, etc.)
- La programación extrema aborda este problema sugiriendo que el software debe ser constantemente **refactorizado**.
- El equipo de programación busca posibles mejoras al software y las pone en práctica inmediatamente.
- **El software siempre debe ser fácil de entender y cambiar**. En la práctica, este no es siempre el caso.

Testing en XP

- Con el desarrollo incremental, no hay especificaciones del sistema que puedan ser utilizadas por un equipo de prueba externo.
- Las características claves del testing en XP son:
  1. Desarrollo “test-first",
  2. desarrollo de prueba incremental a partir de escenarios,
  3. participación del usuario en el desarrollo y validación de la prueba,
  4. el uso de marcos de prueba automatizados.

Testing en XP

- El desarrollo “test first” y las pruebas automatizadas **no necesariamente conducen a pruebas exhaustivas.**
- Los programadores prefieren programar que probar y a veces toman atajos al escribir pruebas
- Algunas pruebas pueden ser muy difíciles de escribir de forma incremental.
- Es difícil juzgar la integridad de un conjunto de pruebas. Aunque puede tener un muchas pruebas del sistema, es posible que su conjunto de pruebas no brinde una cobertura completa.

Story Card: Prescripción de Medicamentos

Kate es un médico que desea recetar medicamentos para un paciente. El registro del paciente ya se muestra en su computadora por lo que hace clic en el campo de medicación y puede seleccionar la **medicación actual**, **nueva medicación** o **formulario** .

Al seleccionar:

- **Medicación actual** , el sistema le pide que verifique la dosis. Si ella quiere cambiar la dosis, ingresa la dosis y luego confirma la prescripción.
- **Nueva medicación**, ella escribe las primeras letras del nombre del medicamento. El sistema muestra una lista de posibles medicamentos comenzando con estas letras. Ella elige el medicamento y el sistema responde pidiéndole que verifique que el medicamento seleccionado es correcto. Ella ingresa la dosis y luego confirma la prescripción.

Story Card: Prescripción de Medicamentos

- **Formulario**, el sistema muestra un cuadro de búsqueda para buscar el medicamento requerido. Ella selecciona un medicamento y se le solicita verificar que el medicamento sea correcto. Ella ingresa la dosis y luego confirma la prescripción.

El sistema siempre verifica que la dosis esté dentro del rango aprobado. Si no es así, se le pide que cambie la dosis.

Después de que Kate confirme la prescripción, se solicita verificación .

- Si hace clic en **Aceptar**, la receta se registra en la auditoría base de datos.
- Si hace clic en **Cambiar**, vuelve a ingresar al proceso de "Prescripción de medicamentos".

Tareas: Prescripción de Medicamentos

Tarea 1: Selección de Formulario

Tarea 2: Control de Dosis

La verificación de la dosis es una medida de seguridad para verificar que el médico no haya recetado una dosis peligrosamente pequeña o grande.

Utilizando el ID del formulario para el nombre del medicamento genérico, busque el formulario y obtenga la dosis máxima y mínima recomendada.

Verifique la dosis prescrita contra el mínimo y máximo.
Si está fuera del rango, emita un mensaje de error que diga que la dosis es demasiado alta o demasiado baja.
Si está dentro del rango, habilite el botón 'Confirmar'.

Caso de Test: Control de Dosis

Test 3: Control de Dosis

Entrada:
1. Un número en mg que representa una dosis única de la droga.
2. Un número que representa el número de dosis únicas por día.

Pruebas:
1. La dosis única es correcta pero la frecuencia es demasiado alta.
2. La dosis única es demasiado alta y muy baja.
3. La dosis única x la frecuencia es demasiado alta y muy baja.
4. La dosis única X frecuencia está en el rango permitido.

Salida:
OK o mensaje de error que indica que la dosis está fuera del rango seguro.

Enfoque Scrum

- El desarrollo ágil requiere un enfoque diferente para la gestión de proyectos, que es adaptado al desarrollo incremental y las fortalezas particulares de los métodos ágiles.

- El enfoque de **Scrum** es un método ágil que se centra en la administración de desarrollo iterativo en lugar del específico para enfoques técnicos desarrollo de software ágil.

Enfoque Scrum

- Fases del Enfoque Scrum:
  1. **Planificación**: se establecen los objetivos generales para el proyecto y se diseña la arquitectura del software.
  2. **Ciclos de Sprints**: cada ciclo desarrolla un incremento del sistema
  3. **Cierre del Proyecto:** finaliza el proyecto, se completa la documentación requerida y se evalúan las lecciones aprendidas del proyecto.

Enfoque Scrum: Sprints

- La característica innovadora de Scrum es su fase central, es decir, **los ciclos de sprint**.
- Un sprint es una unidad de planificación en la que:
  - se evalúa el trabajo a realizar,
  - se seleccionan las funcionalidades para el desarrollo,
  - se implementa el software.
- Al final de un sprint, la funcionalidad completa se entrega a los interesados

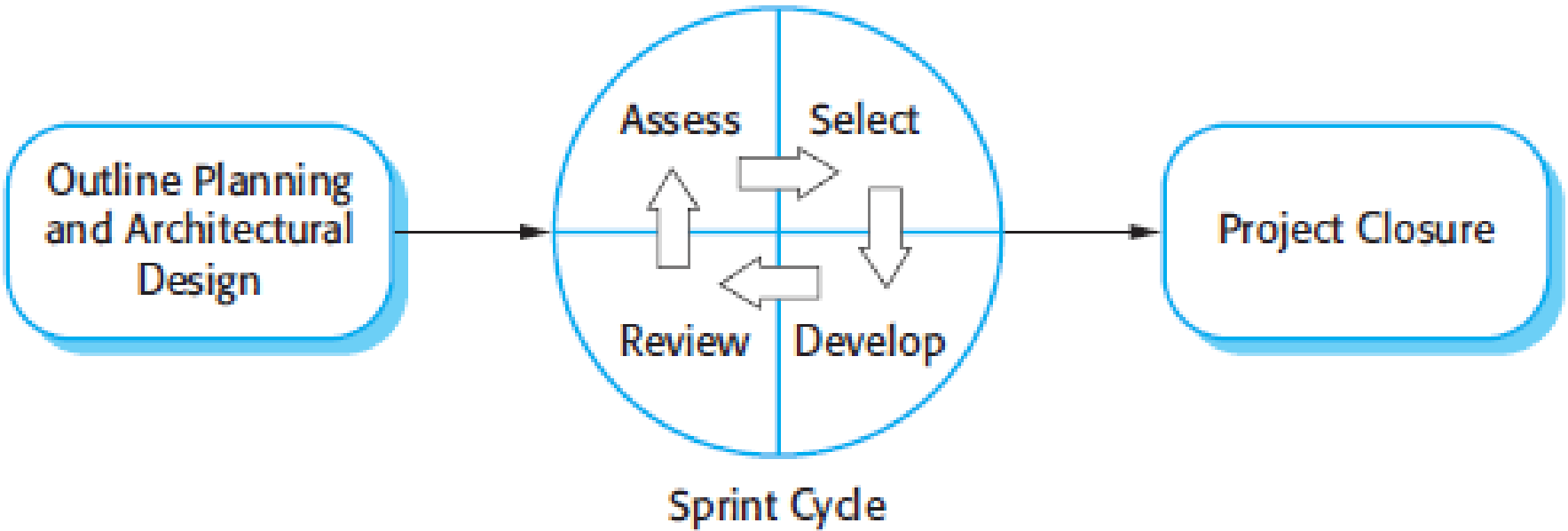
Enfoque Scrum: Características claves

1. Los sprints son de longitud fija, normalmente 2-4 semanas..
2. El punto de partida para la planificación es la acumulación de tareas pendientes (backlogs).
   - Durante la fase de evaluación del sprint, se revisa la lista de backlogs y se asignan prioridades y riesgos.
   - El cliente está muy involucrado en este proceso y puede introducir nuevos requisitos o tareas al comienzo de cada sprint.
3. La fase de selección involucra a todo el equipo del proyecto que trabaja con el cliente para seleccionar las características y la funcionalidades que se desarrollarán durante el sprint.

Enfoque Scrum: Características claves

4. Una vez que se logra un acuerdo, el equipo se organiza para desarrollar el software.
   - Se realizan breves reuniones diarias que involucran a todos los miembros del equipo para revisar el progreso y si es necesario, volver a priorizar el trabajo.
   - Durante esta etapa, el equipo está aislado del cliente y la comunicación se canaliza a través del **'Scrum master**‘, que es el encargado de proteger el equipo de desarrollo de distracciones externas.
5. Al final del sprint, el trabajo realizado se revisa y se presenta a los interesados. El siguiente ciclo de sprint comienza.

Enfoque Scrum



Enfoque Scrum

- Todo el equipo debe estar facultado para tomar decisiones.
- El "Scrum master" es quien que organiza reuniones diarias, rastrea la acumulación de trabajo por hacer, registra las decisiones, mide el progreso contra la acumulación y se comunica con clientes y administración fuera del equipo.
- Todo el equipo asiste a las reuniones diarias, donde todos comparten información, describen su progreso, los problemas, y lo que está planeado para el día siguiente**. Todos saben lo que está pasando**.

Ventajas del Enfoque Scrum

1. El producto se divide en un conjunto manejable y comprensible de partes.
2. Los requisitos inestables no retrasan el progreso.
3. Todo el equipo tiene visibilidad de todo, lo que mejora la comunicación del equipo.
4. Los clientes reciben la entrega a tiempo de incrementos y obtienen retroalimentación sobre cómo funciona el producto.
5. Se establece la confianza entre clientes y desarrolladores y se crea una cultura positiva en la que todos esperan que el proyecto tenga éxito.

Material Bibliográfico

- Ian Sommerville. 2010. Software Engineering (9th ed.). Addison-Wesley Publishing Company, USA.

- Cadle, J., & Yeates, D. (Eds.). 2004. *Project management for information systems*. Pearson education.

- Epstein, D., & Maltzman, R. 2013. *Project workflow management: a business process approach*. J. Ross Publishing.

Recordemos que…

- Fecha límite de conformación de comisiones y definición de ideas del proyecto:

Martes 27/08

- Presentación de la idea del proyecto:

Jueves 29/08

PRÓXIMA CLASE

Ingeniería de Software (parte 1)