Antecesoras de las Ágiles

- Desarrollo Rápido de Aplicaciones (RAD) (iterativo/incremental) (IBM 1977)
- Rational Unified Process – RUP (iterativo/incremental)

Ágiles: son iterativas e incrementales pero además profesan a los principios ágiles definidos en el Manifiesto Ágil.

Metodologías Agiles

- Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.) aparecen como pesados y poco eficientes.
- La crítica más frecuente a estas metodologías “clásicas” es que son demasiado burocráticas. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de las “Metodologías Ágiles”.
- Estos nuevos métodos buscan un punto medio entre la ausencia de procesos y el abuso de los mismos, proponiendo un proceso cuyo esfuerzo valga la pena.

Metodologías Agiles

- Los métodos ágiles son adaptables en lugar de predictivos.
- Los métodos “clásicos” tienden a intentar planear una gran parte del proceso del software en gran detalle para un plazo largo de tiempo. Esto funciona bien hasta que las cosas cambian. Su naturaleza es resistirse al cambio.
- Para los métodos ágiles el cambio es bienvenido. Intentan ser procesos que se adaptan y crecen en el cambio.

Énfasis de las Metodologías Agiles

- Los métodos ágiles son orientados a las personas y no orientados al proceso.
- El objetivo de los métodos “clásicos” es definir un proceso que funcionará bien independientemente de quien lo utilice.
- Los métodos ágiles afirman que ningún proceso podrá nunca “maquillar” las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo.

El Manifiesto Ágil

- En febrero de 2001, en las montañas Wasatch de Utah, se reunieron 17 desarrolladores convencidos de que era necesario un cambio en las metodologías “clásicas” de desarrollo de software.
- Entre ellos se encontraban los creadores de XP, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development y Pragmatic Programming.
- Juntos proclamaron lo que se ha dado a conocer como el “Manifesto for Agile Software Development”

Manifiesto por el Desarrollo Ágil de Software

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.

http://agilemanifesto.org

12 Principios del Manifiesto

1. Satisfacer al cliente a través de entregas continuas y tempranas es la mayor prioridad.
2. Los cambios a los requerimientos son bienvenidos, aún en fases tardías del desarrollo.
3. Entregar frecuentemente software que funciona, desde un par de semanas a un par de meses, prefiriendo los periodos más cortos.
4. Desarrolladores, gerentes y clientes deben trabajar juntos diariamente, a lo largo del proyecto.
5. Construir proyectos alrededor de personas motivadas, dándoles el entorno y soporte que necesitan, y confiando en que realizarán el trabajo.

12 Principios del Manifiesto

6. El método más eficiente y efectivo de transmitir información entre un equipo de desarrolladores es la conversación frontal (cara a cara).
7. Tener sw que funciona es la medida primaria del progreso.
8. El proceso ágil promueve el desarrollo sostenible. Los sponsors, desarrolladores y usuarios deben ser capaces de mantener un ritmo de trabajo constante en forma permanente a lo largo del proyecto.
9. La atención continua a la excelencia técnica y el buen diseño mejoran la agilidad.

12 Principios del Manifiesto

10. Simplicidad – el arte de maximizar el trabajo que no se debe hacer – es esencial.
11. Las mejores arquitecturas, requerimientos y diseños surgen de los equipos auto-organizados.
12. A intervalos regulares, el equipo debe reflexionar sobre como ser más efectivos, y ajustar su comportamiento de acuerdo a ello.

Metodología Ágil

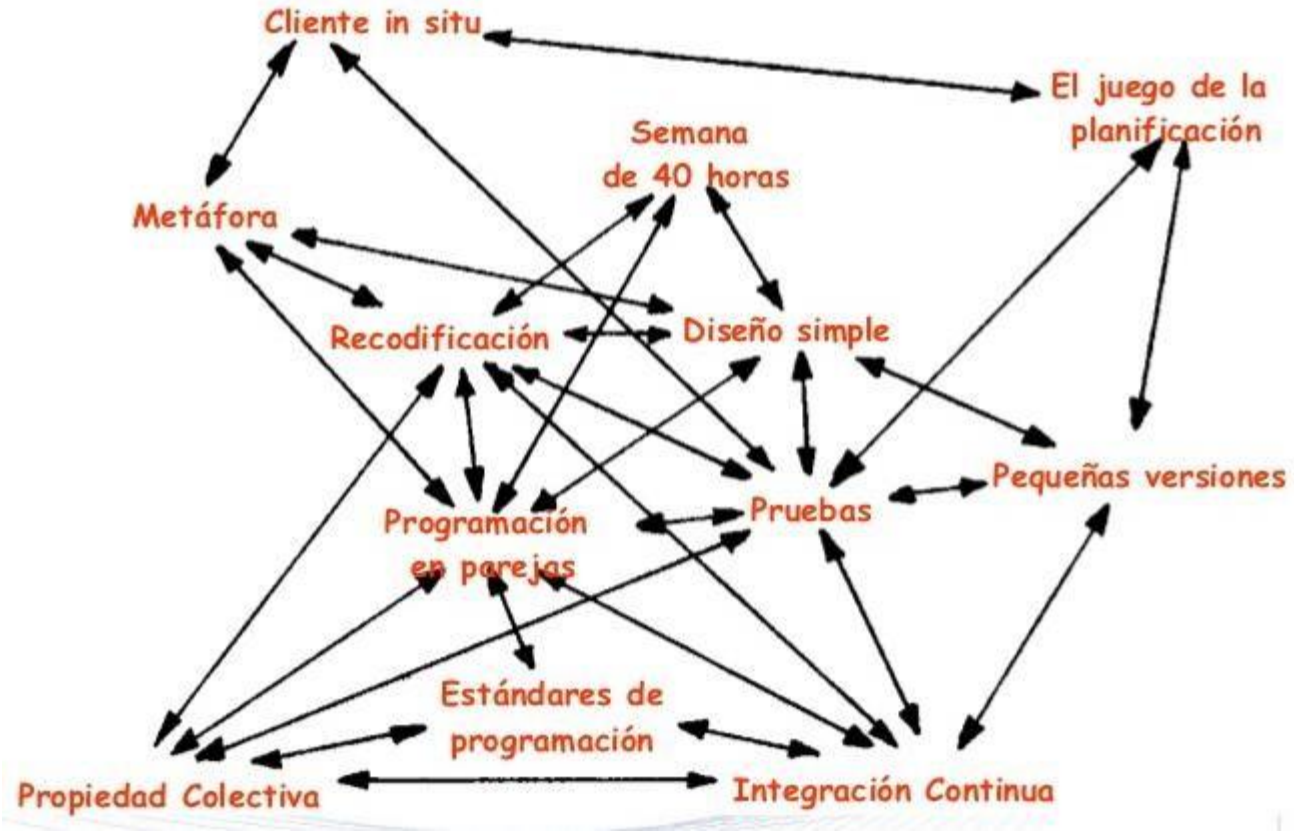
- Los desarrollos de software ágil que adoptan los principios del “Agile Manifesto”, no son anti-metodológicos. Por el contrario, siguen su metodología, diferente a la de los métodos clásicos de desarrollo.

- Se trata de lograr un equilibrio, en el que, por ejemplo, la documentación es concreta y útil, y no burocrática. Los planes existen, pero reconociendo sus limitaciones en el actual mundo en permanente cambio.

Algunas Metodologías Ágiles

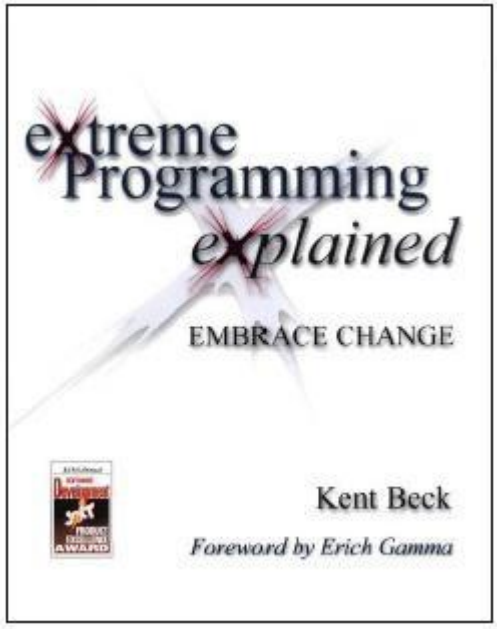
- XP
- SCRUM
- Método de Desarrollo de Sistemas Dinámicos (DSDM)
- Cristal Clear
- Lean Software Development (LSD)
- Adaptive Software Decelopment (ASD)
- Feature Driven Development (FDD)
- Kanban
- Open Unified Process (Open Up)

Prácticas Ágiles



E**x**treme **P**rogramming (XP)

- La programación extrema o eXtreme Programming (XP) es una metodología de desarrollo de la Ing. de software formulada por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999).



- Se originó en el proyecto C3 (Payroll System) de Chrysler, el cual culminó exitosamente en 1997

E**x**treme **P**rogramming (XP)

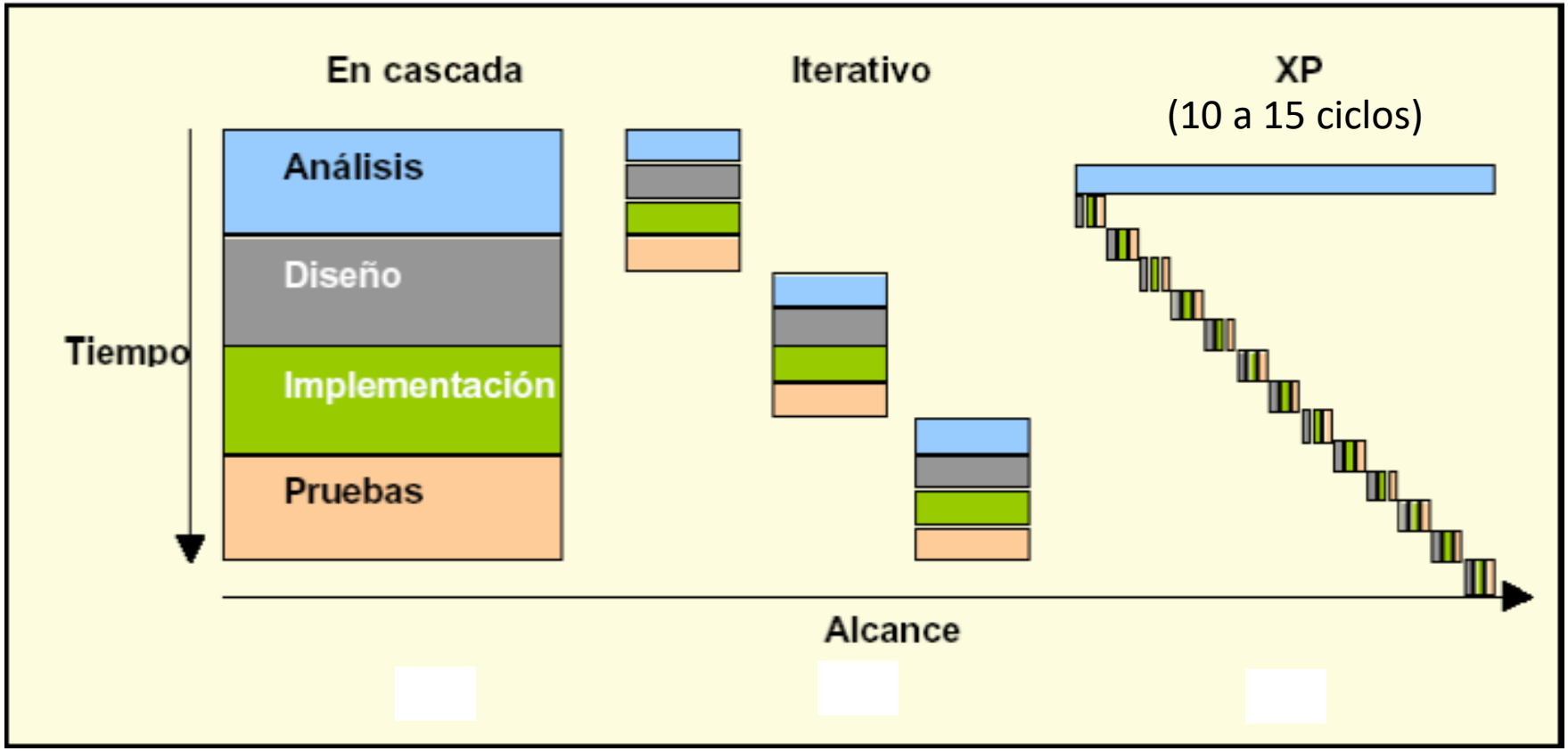
- Es el más destacado de los procesos ágiles de desarrollo de software.
- Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.
- Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.

E**x**treme **P**rogramming (XP)

- El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente.

- Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.

Iteraciones XP



Fase de Exploración

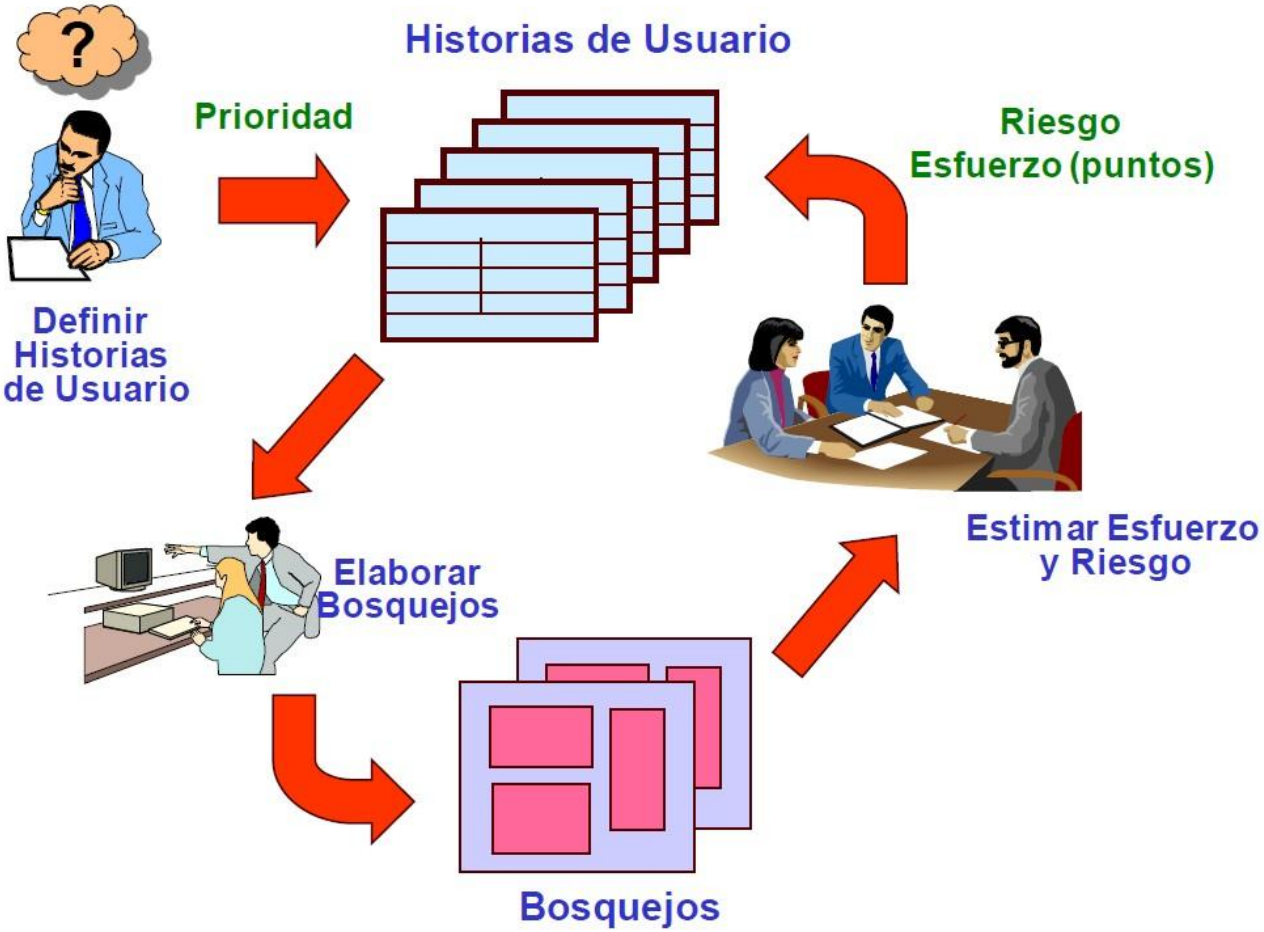
- Se define el alcance general del proyecto.
- El cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”.
- Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración.
- Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Fase de Exploración

HISTORIA DE USUARIO	
Numero: 6	**Nombre:** Ingresar Herramientas
Usuario: Encargado de Bodega	
Modificación de Historia Numero:	**Iteración asignada (asociada a la entrega):** 2
Prioridad en Negocio: Media **(Alta/ Media /Baja)**	
Riesgo en Desarrollo: Bajo **(Alto/ Medio/Bajo)**	
Descripción: Se registra una herramienta cuando ésta llega a bodega indicando Nombre, Marca, Descripción, Código, Precio y Stock	
Observaciones:	

Ejemplo: Sistema de Gestión de Stocks e Inventario – Módulo de Bodega

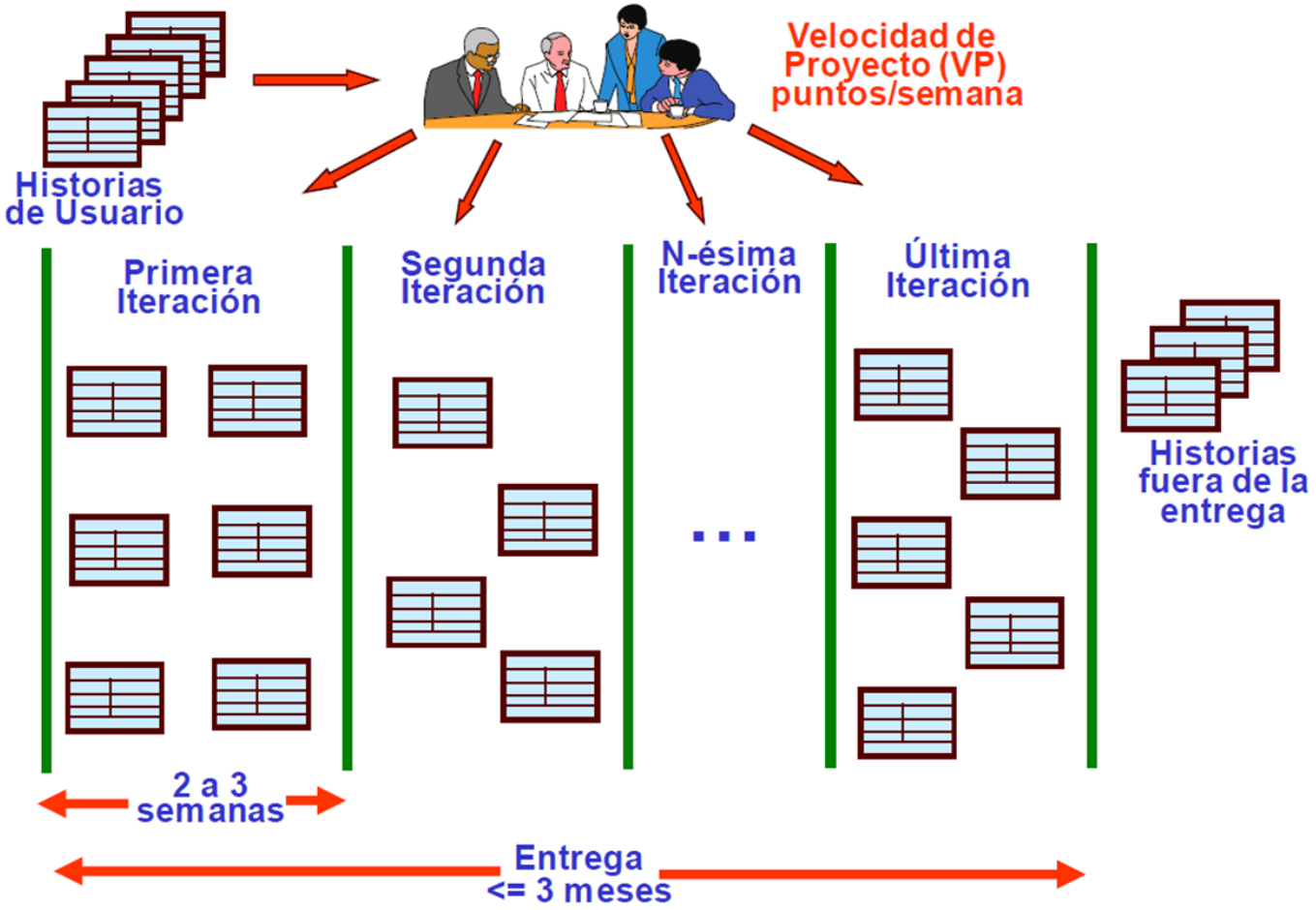
Fase de Exploración



Fase de Exploración

- La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas.
- Típicamente esta fase consiste en una o varias reuniones grupales de planificación.
- El resultado de esta fase es un Plan de Entregas, o “Release Plan”

Fase de Exploración

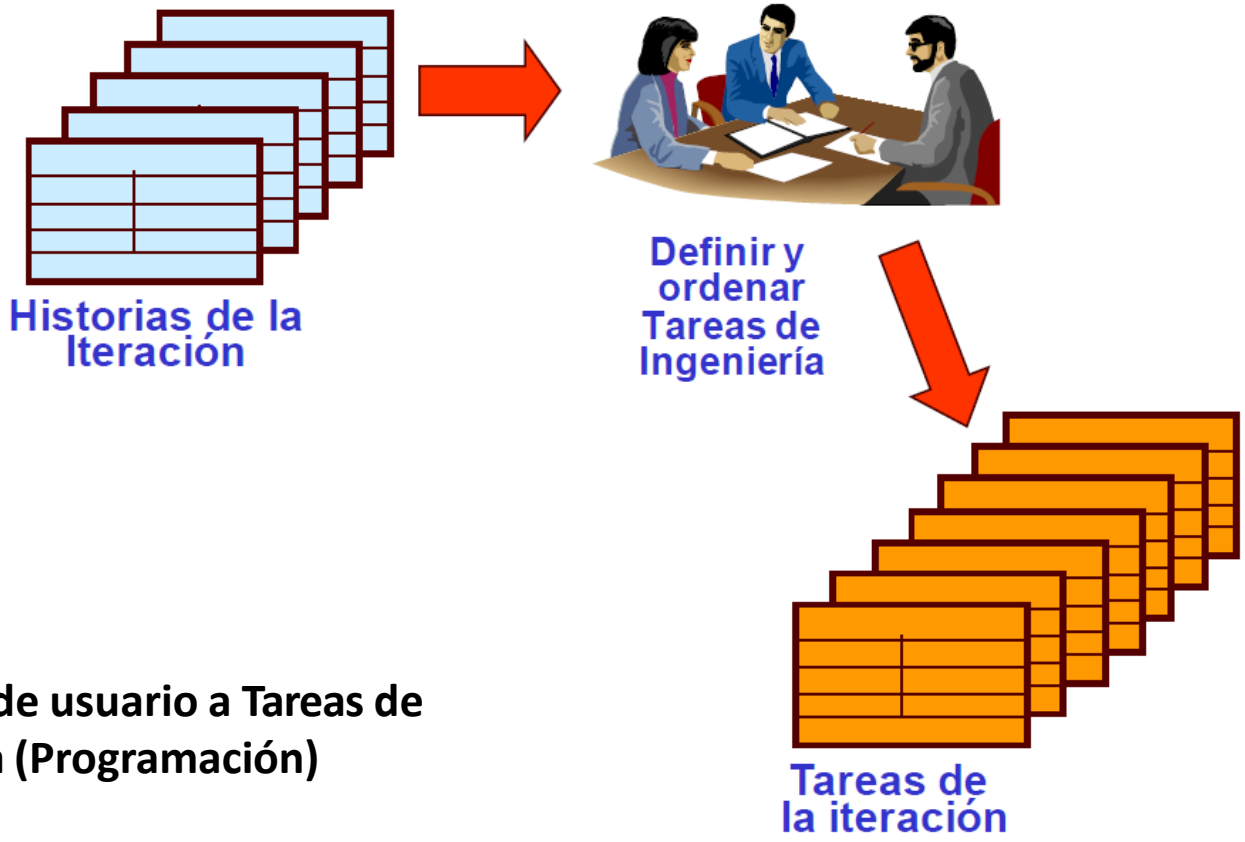


Planificación de Entregas

Fase de Iteraciones

- Esta es la fase principal en el ciclo de desarrollo de XP.
- Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.
- Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios.
- El cliente también debe participar activamente durante esta fase del ciclo.
- Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

Fase de Iteraciones



Historias de usuario a Tareas de Ingeniería (Programación)

Fase de Iteraciones – Programación

Fase de Puesta en Producción

- Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa.
- En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste (“fine tuning”)

Reglas y Prácticas en la Planificación

- La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes.
- El proyecto comienza recopilando “Historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”.
- Una vez obtenidas las “historias de usuarios”, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.
- Si alguna de las “historias de usuario“ tiene “riesgos” que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (“spikes”), para reducir estos riesgos.
- Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes) para establecer un plan o cronograma de entregas en los que todos estén de acuerdo.
- Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas “historias de usuarios”.

Reglas y Prácticas en la Planificación

- Según Martín Fowler (uno de los firmantes del “Agile Manifesto”), los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:
  - Simplicidad del plan. No se espera que un plan requiera de un “gurú” con complicados sistemas de gerenciamiento de proyectos.
  - Los planes son realizados por las mismas personas que realizarán el trabajo.
  - Los planes no son predicciones del futuro, sino simplemente la mejor estimación de cómo saldrán las cosas. Los planes son útiles, pero necesitan ser cambiados cuando las circunstancias lo requieren. De otra manera, se termina en situaciones en las que el plan y la realidad no coinciden, y en estos casos, el plan es totalmente inútil.

Historias de Usuario

- Las “Historias de usuarios” (“User stories”) sustituyen a los documentos de especificación funcional, y a los “casos de uso”.
- Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido.
- Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo.
- Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.
- Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

Historias de Usuario

HISTORIA DE USUARIO	
Numero: 6	**Nombre:** Ingresar Herramientas
Usuario: Encargado de Bodega	
Modificación de Historia Numero:	**Iteración asignada (asociada a la entrega):** 2
Prioridad en Negocio: Media **(Alta/ Media /Baja)**	
Riesgo en Desarrollo: Bajo **(Alto/ Medio/Bajo)**	
Descripción: Se registra una herramienta cuando ésta llega a bodega indicando Nombre, Marca, Descripción, Código, Precio y Stock	
Observaciones:	

Ejemplo: Sistema de Gestión de Stocks e Inventario – Módulo de Bodega

Plan de Entrega (“Release Plan”)

- El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas.
- Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.).
- XP denomina a esta reunión “Juego de planeamiento” (“Planning game”), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, “Planning meeting” o “Planning workshop”)
- Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.
- Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.
- Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.
- Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación.
- Para cada historia de usuario se establecen las **pruebas de aceptación**. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.
- Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

Plan de Entrega (“Release Plan”)

TAREA	
Número tarea: 8	**Número historia: 6**
Nombre tarea: Diseño interfaz módulo Bodega	
Tipo de tarea (nueva, corrección, Mejora)**:** Nueva (desarrollo)	**Puntos estimados (complejidad): 1**
Fecha inicio: 3/8/21	**Fecha fin:** **4**/8/21
Programador responsable: Jean Carvajal – Olivia Coñuepan	
Descripción: Se diseñara una ventana que esté compuesta por dos pestañas {Materiales, Herramientas}. Cada una de las cuales contendrá 3 botones, uno para el ingreso de materiales/herramientas, otro para el registro de salida de materiales/herramientas y otro para ver el stock actual de materiales/herramientas. Al apretar algún botón se abrirá una ventana aparte correspondiente al botón presionado.	

Ejemplo de Tarea asociada a Historia de Usuario “Ingresar Herramientas”

Reuniones diarias de seguimiento (“Stand-up meeting”)

- El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones.
- En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar.
- Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

Reuniones diarias de seguimiento ("Stand-up meeting")

Todo el equipo

- Problemas
- Soluciones

De pie en un círculo

- Evitar discusiones largas
- Sin conversaciones separadas

Reglas y prácticas en el Diseño

- La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:
- **Simplicidad:** Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

Reglas y prácticas en el Diseño

- Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento

Reglas y prácticas en el Diseño

- **Soluciones “spike”:**
  - Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados “spike”), para explorar diferentes soluciones.
  - Estos programas son únicamente para probar o evaluar una solución y suelen ser desechados luego de su evaluación (son como prototipos desechables).
- **Recodificación (“refactoring”):**
  - Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, para hacerlo más simple, conciso y/o entendible.
  - Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, mas clara y eficientemente. Sin embargo, como ya está listo y “funciona”, rara vez es reescrito.
  - Las metodologías de XP sugieren recodificar cada vez que sea necesario. Si bien, puede parecer una pérdida de tiempo innecesaria en el plazo inmediato, los resultados de esta práctica tienen sus frutos en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad.
  - La filosofía que se persigue es tratar de mantener el código más simple posible que implemente la funcionalidad deseada.

Reglas y prácticas en el Diseño

- **Metáforas:**
  - Una “metáfora” es algo que todos entienden, sin necesidad de mayores explicaciones.
  - La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto y guiar la estructura y arquitectura del mismo.
  - Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo.
  - Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan las “metáfora”, para que puedan dialogar en un “mismo idioma”.
  - Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.
  - Sin embargo, esta práctica resulta, muchas veces, difícil de realizar.

Reglas y prácticas en la Codificación

- **Disponibilidad del cliente:**
  - Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo.
  - El cliente debe proporcionar las historias de usuarios, pero dado que estas historias son cortas y de “alto nivel”, no contienen los detalles necesarios para realizar el desarrollo del código.
  - Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo.
  - No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, “cara a cara”.
  - Si bien esto parece demandar del cliente recursos por un tiempo prolongado, debe tenerse en cuenta que en otras metodologías este tiempo es empleado por el cliente en realizar los documentos detallados de especificación.
  - Al estar el cliente en todo el proceso, puede prevenir a tiempo de situaciones no deseables, o de funcionamientos que no eran los que en realidad se deseaban. En otras metodologías, estas situaciones son detectadas en forma muy tardía del ciclo de desarrollo, y su corrección puede llegar a ser muy complicada.

Reglas y prácticas en la Codificación

- **Disponibilidad del cliente:**
  - Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo.
  - El cliente debe proporcionar las historias de usuarios, pero dado que estas historias son cortas y de “alto nivel”, no contienen los detalles necesarios para realizar el desarrollo del código.
  - Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo.
  - No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, “cara a cara”.
  - Si bien esto parece demandar del cliente recursos por un tiempo prolongado, debe tenerse en cuenta que en otras metodologías este tiempo es empleado por el cliente en realizar los documentos detallados de especificación.
  - Al estar el cliente en todo el proceso, puede prevenir a tiempo de situaciones no deseables, o de funcionamientos que no eran los que en realidad se deseaban. En otras metodologías, estas situaciones son detectadas en forma muy tardía del ciclo de desarrollo, y su corrección puede llegar a ser muy complicada.

Reglas y prácticas en la Codificación

- **Uso de estándares:**
  - Si bien esto no es una idea nueva, XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo y que facilite la recodificación.
- **Programación dirigida por las pruebas ("Test-driven programming"):**
  - En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada al final del proyecto, o al final del desarrollo de cada módulo.
  - La metodología XP propone un modelo inverso, en el que lo primero que se escribe son los test que el sistema debe pasar.
  - Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas (pruebas unitarias, realizados por los desarrolladores).
  - La definición de estos test al comienzo, condiciona o "dirige" el desarrollo.

Reglas y prácticas en la Codificación

- **Programación en pares:**
  - XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo computador.
  - Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas.
  - El producto obtenido es, por lo general, de mejor calidad que cuando el desarrollo se realiza por programadores individuales.
  - En un estudio realizado por Cockburn y Williams (2000), se concluye que la programación en pares tiene un sobre costo aproximado de 15%, y no de un 100% como se puede pensar a priori. Este sobre costo es rápidamente pagado por la mejor calidad obtenida en el producto final.
- **La programación en pares tiene las siguientes ventajas:**
  - La mayoría de los errores se descubren en el momento en que se codifican, ya que el código es permanentemente revisado por dos personas.
  - La cantidad de defectos encontrados en las pruebas es estadísticamente menor.
  - Los diseños son mejores y el código más corto. El equipo resuelve problemas en forma más rápida.
  - Las personas aprenden significativamente más, acerca del sistema y de desarrollo de software.
  - El proyecto termina con más personas que conocen los detallas de cada parte del código.
  - Las personas aprenden a trabajar juntas, generando mejor dinámica de grupo y haciendo que la información fluya rápidamente.
  - Las personas disfrutan más de su trabajo.

Reglas y prácticas en la Codificación

- **Integraciones permanentes:**
  - Todos los desarrolladores necesitan trabajar siempre con la “última versión”.
  - Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores.
  - Idealmente, todos los días deben existir nuevas versiones publicadas.
  - Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.
- **Propiedad colectiva del código:**
  - En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto.
  - Cualquier pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar.
  - En otras metodologías, este concepto puede parecer extraño. Muchas veces se asume que, si hay algo de propiedad colectiva, la responsabilidad también es colectiva. Y que “todos sean responsables”, muchas veces significa que “nadie es responsable”. Sin embargo, en XP quienes encuentran un problema, o necesitan desarrollar una nueva función, pueden resolverlo directamente, sin necesidad de “negociar” con el “dueño” o autor del módulo (ya que, de hecho, este concepto no existe en XP).
  - Muchas veces, una solución pasa por la recodificación de varios módulos, que atraviesan de forma horizontal una determinada jerarquía vertical. Si es necesario dialogar y convencer al encargado de cada módulo, posiblemente la solución no se pueda implementar, por lo menos en tiempos razonables.
  - En XP, se promueve la recodificación para generar códigos mas simples y adaptados a las realidades cambiantes. Cualquier pareja de programadores puede tomar la responsabilidad de este cambio.
  - Los testeos permanentes deberían asegurar que los cambios realizados cumplen con lo requerido, y además, no afectan al resto de las funcionalidades.

Reglas y prácticas en la Codificación

- **Ritmo sostenido (“Semana de 40 hrs”):**
  - La metodología XP indica que debe llevarse un ritmo sostenido de trabajo. Lo importante no es si se trabajan, 35, 40 o 42 horas por semana. El concepto que se desea establecer con esta práctica es el de planificar el trabajo de manera de mantener un ritmo constante y razonable, sin sobrecargar al equipo.
  - Cuando un proyecto se retrasa, trabajar tiempo extra puede ser más perjudicial que beneficioso. El trabajo extra desmotiva inmediatamente al grupo e impacta en la calidad del producto.
  - En la medida de lo posible, se debería renegociar el plan de entregas, realizando una nueva reunión de planificación con el cliente, los desarrolladores y los gerentes.
  - Agregar más desarrolladores en proyectos ya avanzados no siempre resuelve el problema.

Reglas y prácticas en las Pruebas

- **Pruebas unitarias:**
  - Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben pasar las pruebas unitarias antes de ser liberados o publicados.
  - Las pruebas deben ser definidas antes de realizar el código. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código.
  - El sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.
- **Detección y corrección de errores:**
  - Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir.
  - Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

Reglas y prácticas en las Pruebas

- **Pruebas de aceptación:**
  - Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.
  - Las pruebas de aceptación son consideradas como “pruebas de caja negra” (“**Black box system tests**”). Los clientes son responsables de verificar que los resultados de éstas pruebas sean correctos.
  - En caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución.
  - Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación.
  - Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

Reglas y prácticas en las Pruebas

Prueba	
Numero caso de prueba: 1	**Número historia de usuario: 6**
Nombre caso de prueba: Ingreso herramientas	
Descripción: Se llenará el formulario para el ingreso de herramientas, dejando algún campo en blanco.	
Condiciones de ejecución: Debe estar creada y configurada la base de datos	
Entradas: Nombre: Esmeril Angular; Marca: Bosch; Código: 3400; Precio: $35.000; Stock: (null); Descripción: (null);	
Resultado esperado: Debiera aparecer algún mensaje de error	
Evaluación: Aparece mensaje de error. *Correcto*	

Roles de XP

De acuerdo a la propuesta original de Beck son:

- **Programador:** El programador escribe las pruebas unitarias y produce el código del sistema. Sin distinción entre analistas, diseñadores y programadores.
- **Cliente:** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (Tester):** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker):** Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador (Coach):** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor:** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor – Jefe de Proyecto (Big boss):** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Valores XP

- La **simplicidad** es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente.
- Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado.
- La **comunicación** se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo entendible.
- El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado.
- Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método.
- Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de como utilizar su funcionalidad.
- Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas.

Valores XP

- **(Retroalimentación)** Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante.
- En desarrollos con ciclos largos, meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo.
- El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código.
- Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

Valores XP

- Muchas de las prácticas implican **valentía**. Una de ellas es siempre diseñar y programar para hoy y no para mañana. Esto es un esfuerzo para evitar "empantanarse" en el diseño y requerir demasiado tiempo y trabajo para implementar todo lo demás del proyecto.
- La valentía le permite a los desarrolladores que se sientan cómodos con reconstruir su código cuando sea necesario. Esto significa revisar el sistema existente y modificarlo si con ello los cambios futuros se implementarán mas fácilmente.
- Otro ejemplo de valentía es saber cuando desechar un código: valentía para quitar código fuente obsoleto, sin importar cuanto esfuerzo y tiempo se invirtió en crear ese código.
- Además, valentía significa persistencia: un programador puede permanecer sin avanzar en un problema complejo por un día entero, y luego lo resolverá rápidamente al día siguiente, sólo si es persistente.
- Los miembros del equipo **respetan** su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo o más eficiente para la solución a través de la refactorización del código.
- Los miembros del equipo respetan el trabajo del resto no haciendo menos a otros, una mejor autoestima en el equipo y elevando el ritmo de producción en el equipo.

Aplicabilidad de XP

- Para proyectos medianos, y en los que las especificaciones no se puedan obtener hasta luego de comenzado el proyecto
- La metodología XP aplica a equipos relativamente pequeños. Si bien no hay un consenso en el número máximo de desarrolladores, todos parecen coincidir en números no mayores a 20. Sus propias prácticas así lo requieren, ya que, por ejemplo, mantener las “Stand-up meeting” con más de 20 personas parece poco razonable.
- El entorno físico en el que se realizan los desarrollos deben ser adecuados a la metodología. Escritorios amplios, con un computador y dos sillas, mesas redondas para trabajo en equipo, ambientes que permitan la permanente comunicación y colaboración, son algunos de los requerimientos de infraestructura para poder implementar esta metodología.
- La participación e involucramiento del cliente en el proceso es fundamental. El cliente debe conocer y aprobar la metodología, y destinar los recursos necesarios.
- Los participantes del equipo de desarrollo deben estar compenetrados con el proyecto y su metodología. Deben aceptar compartir sus códigos y ser responsables por el código que escribieron otros.

Críticas a XP

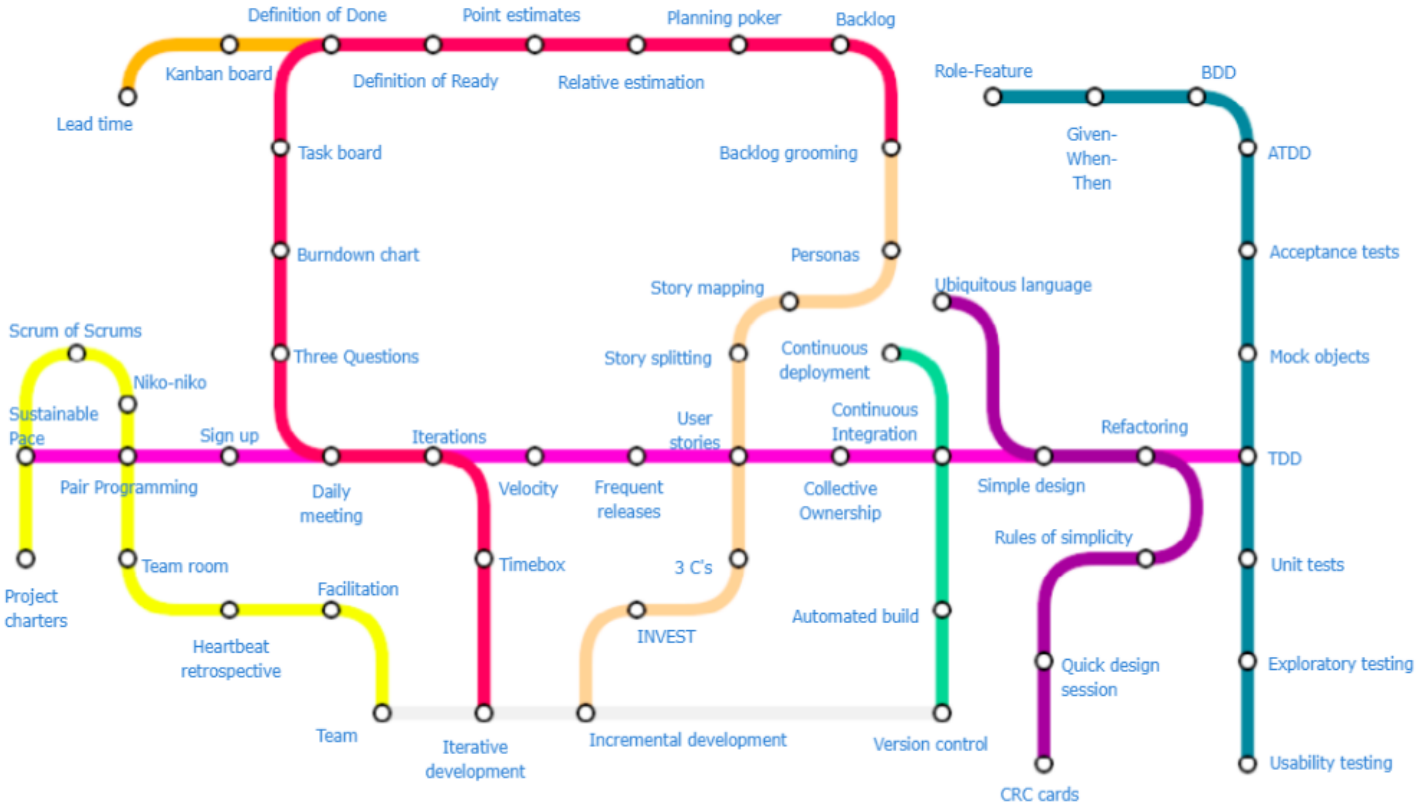
- Dificultad de estimar cuánto va a costar un proyecto. Dado que el alcance del mismo no está completamente definido al comienzo, y que la metodología XP es expresamente abierta a los cambios durante todo el proceso, se torna sumamente difícil poder realizar un presupuesto previo.
- Para desarrollos “in house” este punto puede no ser crítico, pero sí lo es especialmente para empresas desarrolladoras de software, donde deben presupuestar (y adjudicar) proyectos.
- Baja documentación, pensando en el posterior mantenimiento del sistema. Si bien durante el proyecto el equipo tiene en mente todas sus particularidades, hay que prever que pasará luego de entregado, cuando el equipo se disuelva, y sea necesario realizar algún cambio o mejora.
- XP propone la recodificación permanente, para asegurar la claridad y simplicidad del código. Sin embargo, es posible que aún un código simple y claro no baste cuando otro equipo de trabajo tenga que tomar el sistema y cambiarlo.
- Es posible que sea necesario mantener cierta documentación, aunque ésta debe ser la mínima necesaria.

Conclusión

En una encuesta realizada sobre 45 proyectos realizados con XP se concluye que:

- Casi todos los proyectos se categorizaron como exitosos. El 100% de los desarrolladores encuestados afirmaron que volvería a utilizar la metodología XP en el siguiente proyecto si fuera apropiado.
- Las frecuentes ausencias del cliente fueron identificadas como el mayor riesgo en los proyectos.
- Los problemas más comunes fueron “barreras psicológicas”, como por ejemplo el escepticismo de la línea gerencial, la filosofía de la empresa desarrolladora que no permitía tener al cliente en sitio, o que algunos desarrolladores se oponía al trabajo en parejas.
- Los elementos más útiles de XP fueron la propiedad colectiva del código, las pruebas y la integración continua. Las más críticas fueron la metáfora y el cliente en sitio.

Otras áreas agiles



Lines represent practices from the various Agile "tribes" or areas of concern:

- Extreme Programming
- Teams
- Lean
- Scrum
- Product management
- Devops
- Design
- Testing
- Fundamentals