



Introducción a las Metodologías Ágiles

CC62V Taller de Metodologías Ágiles de Desarrollo de Software

Agustín Villena

agustin.villena@gmail.com

agustin.villena@gmail.com

Motivación

*Software fails to deliver...
and fails to deliver value!*

Kent Beck, Extreme Programming Explained

o ¿Problemas comunes al desarrollar software?...

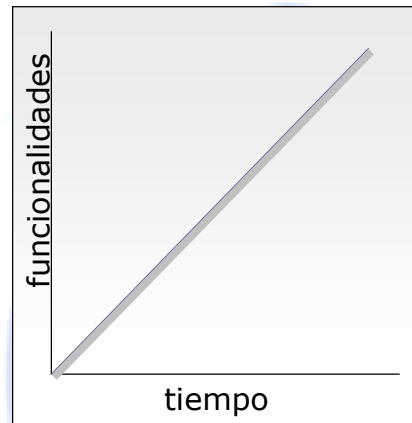
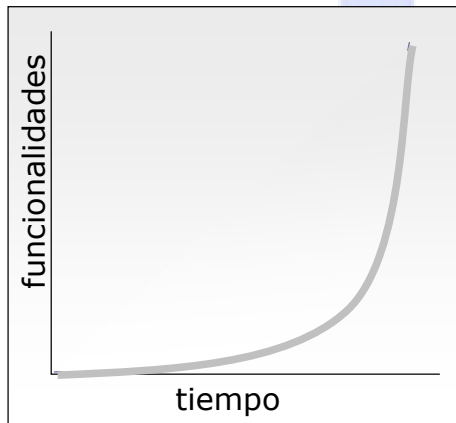
- Caos,
- Falta de un método sistemático de trabajo, que se sigue **a veces**
- La dura realidad: **Code like hell**
- Entregas llenas de **cool features** que el cliente nunca utiliza (tiempo y dinero desperdiciados)
- Software lleno de bugs, o de código inmantenible

agustin.villena@gmail.com

Fundamentos

El valor entregado del proyecto
en su tiempo de desarrollo

- Tradicionalmente
- ¿Y si fuera así?



agustin.villena@gmail.com

Problema fundamental del desarrollo de innovaciones

Lidiar con el **Riesgo**

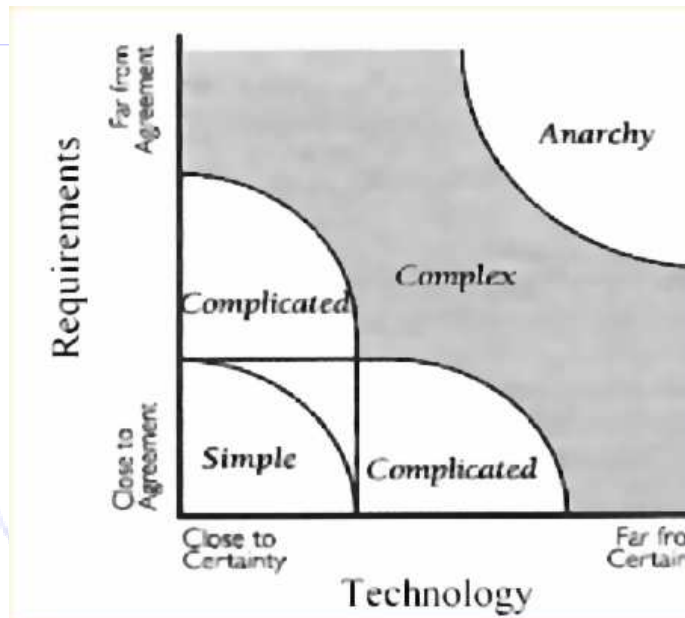
- ¿Causa del Riesgo?

La **incertidumbre**



agustin.villena@gmail.com

Las dimensiones de la incertidumbre



Ogunnaik and Ray:
Process Dynamics,
Modeling, and Control.

agustin.villena@gmail.com

Porqué de la curva tradicional de valor (cont.) Big Design Upfront

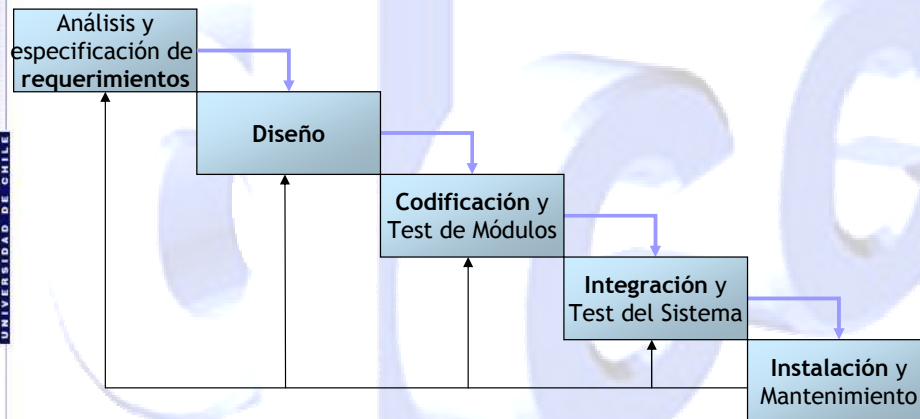
- Tradicionalmente, para atacar la incertidumbre,
 - se trata de **definir todo desde un principio**
 - generando un **diseño y plan detallado**, que luego el desarrollador debe **ejecutar**
 - Conformado por requerimientos **abstractos y detallados**
 - En este modelo, el cliente **delega** su responsabilidad en el desarrollador
 - Y si este no cumple lo definido es **castigado** por multas en el contrato

agustin.villena@gmail.com

Porqué de la curva tradicional de valor (cont.)

Modelo de Cascada

- Esto da origen al Modelo de Cascada de desarrollo



agustin.villena@gmail.com

Porqué de la curva tradicional de valor (cont.)

Variables que definen un proyecto

- **Tiempo** disponible para el proyecto
- **Costo (Recursos)** que se dispondrán
- **Alcance**, es decir conjunto de funcionalidades que se desarrollarán
- **Calidad** (Resistencia a fallas, eficiencia, etc.)
- Por lo regular se fijan las 3 primeras variables:
 - *"Hazme un administrador de estas tablas de la base de datos, en una semana, con dos programadores"*

agustin.villena@gmail.com

Porqué de la curva tradicional de valor (cont.) El problema del cambio

- En un proyecto que implique innovar existe una sola certeza: **el cambio**
 - En las condiciones **internas** o **externas**
 - Porque al **adentrarse en definir la solución** ahora se **conoce mejor el problema a resolver**
 - Etc



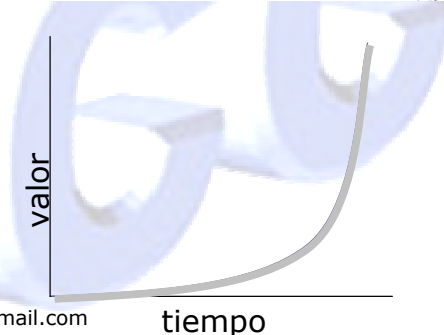
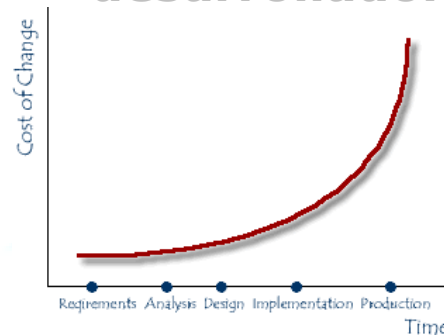
agustin.villena@gmail.com

Porqué de la curva tradicional de valor (cont.) El cambio y las variables de un proyecto

- Al suceder el cambio, ¿qué pasa con las variables?:
 - **Tiempo** y **Costo** no son en realidad muy flexibles
 - **El Alcance está fijo** por el diseño y el plan definido
 - Entonces, para acomodar los cambios el desarrollador está forzado a bajar la **Calidad**, lo que afecta el **valor generado** para el cliente
 - Esto genera **conflicto**
 - El cliente siempre querrá el mayor valor por sus recursos invertidos
 - Y el desarrollador la mayor calidad
 - ¿Qué persona sana desea hacer un trabajo mediocre?

Porqué de la curva tradicional de valor (cont.) Paradigmas tradicionales del desarrollador

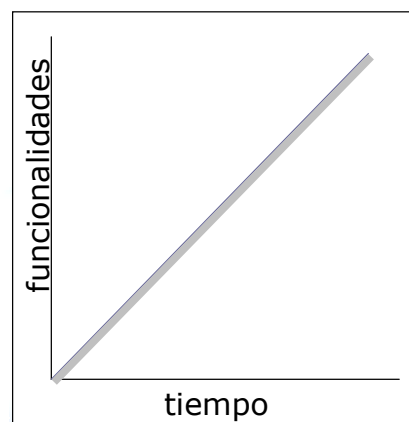
- Cree que a medida que avanza un proyecto, los cambios son más caros
 - Se tiende a desarrollar cosas que **se cree** que alguna vez se necesitarán
 - Tiende a priorizar según su propia visión, no la del negocio
 - *Primero diseñamos **toda** la BD y sus mantenedores, y entonces implementamos las aplicaciones cliente*



agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto

- Ventajas de la nueva curva de valor
 - Lo desarrollado tiene una utilidad promedio superior
 - Al fin de cada iteración del desarrollo, se tiene un producto funcional y útil
 - ¡El cliente obtendrá mayor valor!



agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto ¿Cómo se logra?

- Definamos un nuevo modelo, donde la incertidumbre se ataca
 - Definiendo un **plan preliminar**
 - Conformado por requerimientos, que son **ambiguos** pero **útiles**, es decir **estimables** y **verificables**
 - Obteniendo **retroalimentación exacta y precisa** del **avance y estado** de un proyecto
 - Invirtiendo los **menos recursos posibles** al comienzo
 - se elimina **todo aquello que no genere valor**.
 - Por ejemplo, el sobrediseño
 - Ofreciendo la oportunidad de **ir más rápido**
 - Permitiendo **cambiar drásticamente** los requerimientos (**alcance variable**)

agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto Reglas para una relación sinérgica

	Cliente	Desarrollador
Desea maximizar	Valor recibido por cada semana de desarrollo	Calidad del trabajo realizado
Puede definir	Qué será implementado, y en qué prioridad, según las necesidades de su negocio	Cuánto se estima que demorará una tarea (idealmente)
Puede cambiar	Funcionalidades solicitadas por otras no implementadas de costo equivalente (canjear)	Sus estimaciones en base a nuevos descubrimientos

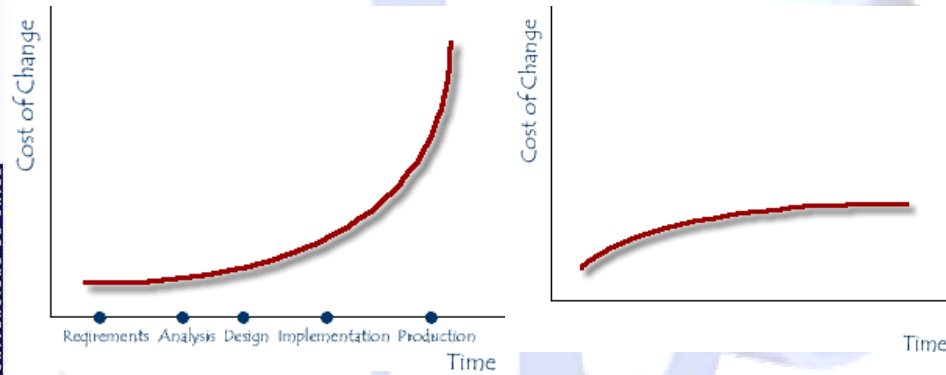
La tecnología es una decisión del cliente

- Aquí el cliente **comparte** su responsabilidad sobre el proyecto con el desarrollador, en una relación de **sinergia**
 - (inspirado en el modelo Toyota de gestión)

agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto Rediseñando la curva de costo del cambio

- Visión tradicional
- ¿Y si fuera así?



agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto Rediseñando la curva de costo del cambio

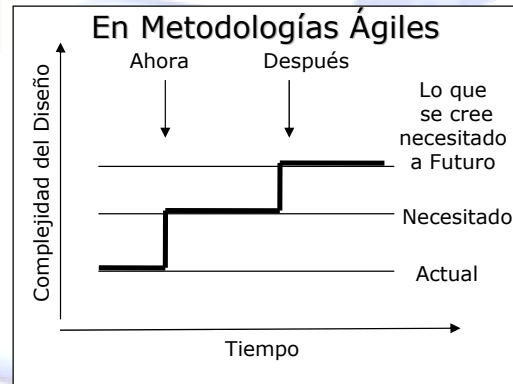
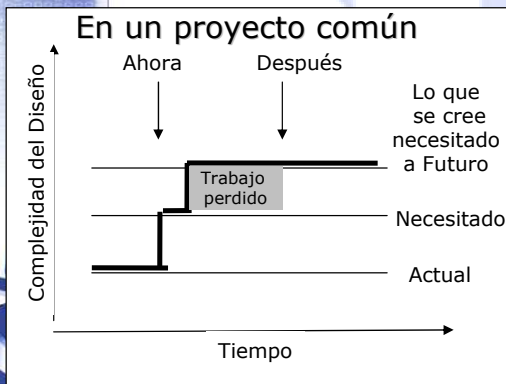
- No sucede mágicamente...
- En el ámbito del software, surgen las metodologías ágiles
 - Hay tecnologías que son más "amigables" con el cambio
 - Programación Orientada a Objetos
 - Bases de Datos Orientadas a Objetos
 - Pero la tecnología sola no sirve...
 - Prácticas útiles
 - Un **diseño simple del sistema**, sin "áreas desiertas" (código que no se usa)
 - Un **sistema automático de Pruebas**
 - Permite hacer cambios y comprobar la coherencia del sistema
 - **Mucha práctica en cambiar** diseños de sistemas
 - **Definir y generar entregables evolutivos**, en incrementos pequeños

agustin.villena@gmail.com

Aumentando el valor entregado por un proyecto Rediseñando la curva de costo del cambio (continuación)

○ Ejemplo: **Diseño Simple**

- Lo más simple que pueda funcionar
- Hacer ahora lo que se necesita ahora, no lo que se cree que necesitará a futuro



agustin.villena@gmail.com

Entra eXtreme Programming

- Enunciada por **Kent Beck, Ward Cunningham, and Ron Jeffries.**
- Concebida en el proyecto C3 (Chrysler Comprehensive Compensation)
- Documentada en el WikiWiki original
 - <http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap>
- Formalizada en 1999 en el libro **"Extreme Programming Explained"**
 - Que va en su segunda versión

agustin.villena@gmail.com



Entra eXtreme Programming (cont)

- Para lograr las metas ágiles antes planteadas, XP define un framework interrelacionado de
 - Valores (5)
 - Principios (+ de 12)
 - Prácticas (+ de 20)

agustin.villena@gmail.com



Fundamentos de XP Valores de XP

- **Valores Comunes:** son los que permiten que las personas **trabajen por el beneficio común antes que el propio**
- Y en definitiva, el desarrollo de software es una **actividad humana**
 - Es afectada por la motivación, creencias y los instintos de las personas

agustin.villena@gmail.com

Fundamentos de XP

Valores de XP

Comunicación 😊 (Centro de todo problema humano) <ul style="list-style-type: none">• Abierta y honesta• Enseñar a aprender• Trabajar con los instintos de las personas	Simplicidad 🐉 <ul style="list-style-type: none">• Asumirla siempre• Viajar con equipaje: poco, simple y valioso• Cambios paso a paso• Adaptación local
Coraje ⚡ <ul style="list-style-type: none">• Jugar a ganar• Responsabilidad aceptada (antes que asumida)• Trabajo de Calidad• Atacar problema urgente, dejando la mayor cantidad de opciones	Retroalimentación ↻ <ul style="list-style-type: none">• Rápida (favorece el aprendizaje)• Medir honestamente• Experimentos concretos

Y un 5º valor implícito : el **Respeto**

agustin.villena@gmail.com

Fundamentos de XP

Prácticas : Lo "extremo" de XP

- **Kent Beck:**
 - "Llevar buenas prácticas de Ingeniería de Software al extremo"
- Algunos ejemplos:
 - Revisiones de código => Programación de a pares
 - Sistema de Pruebas Estructurado => Desarrollo Guiado por Pruebas
 - Software funcionando => Entregas Incrementales e Integración Continua

agustin.villena@gmail.com

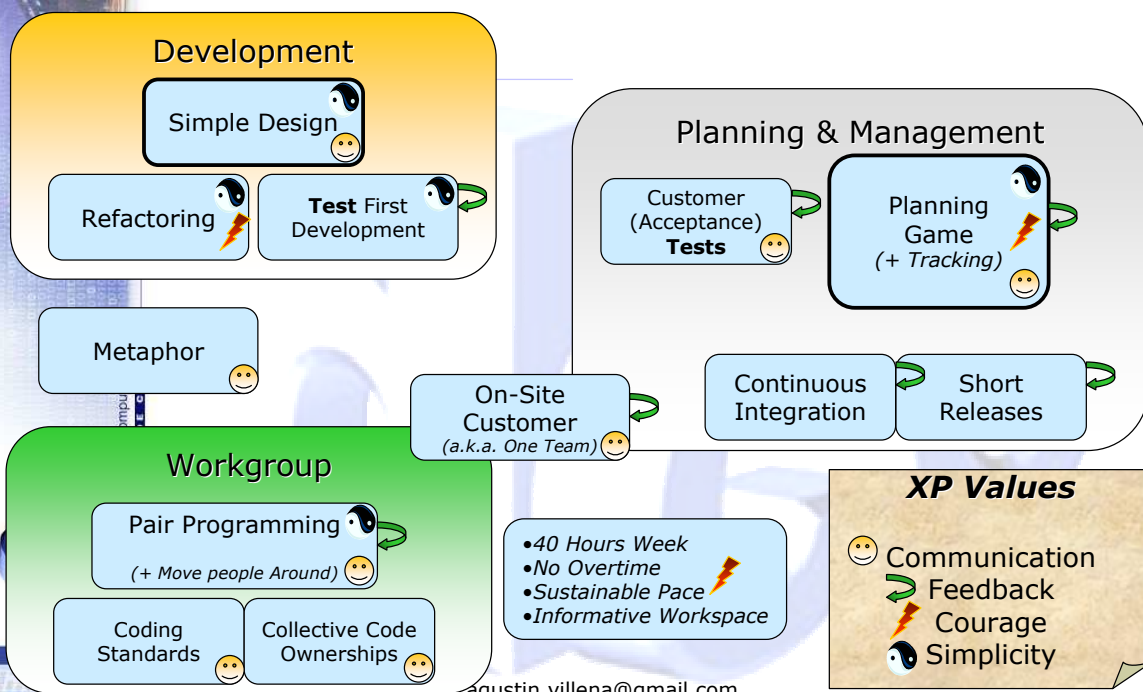
Fundamentos de XP

Prácticas : Lo "extremo" de XP (cont)

- Kent Beck:
 - "Llevar buenas prácticas de Ingeniería de Software al extremo"
- En vez de
 - diseñar-programar-probar-depurar-armar-entregar, ***pasar a***
 - Diseñar continuamente, Probar continuamente, Armar continuamente, Revisar continuamente y Entregar temprana y frecuentemente.

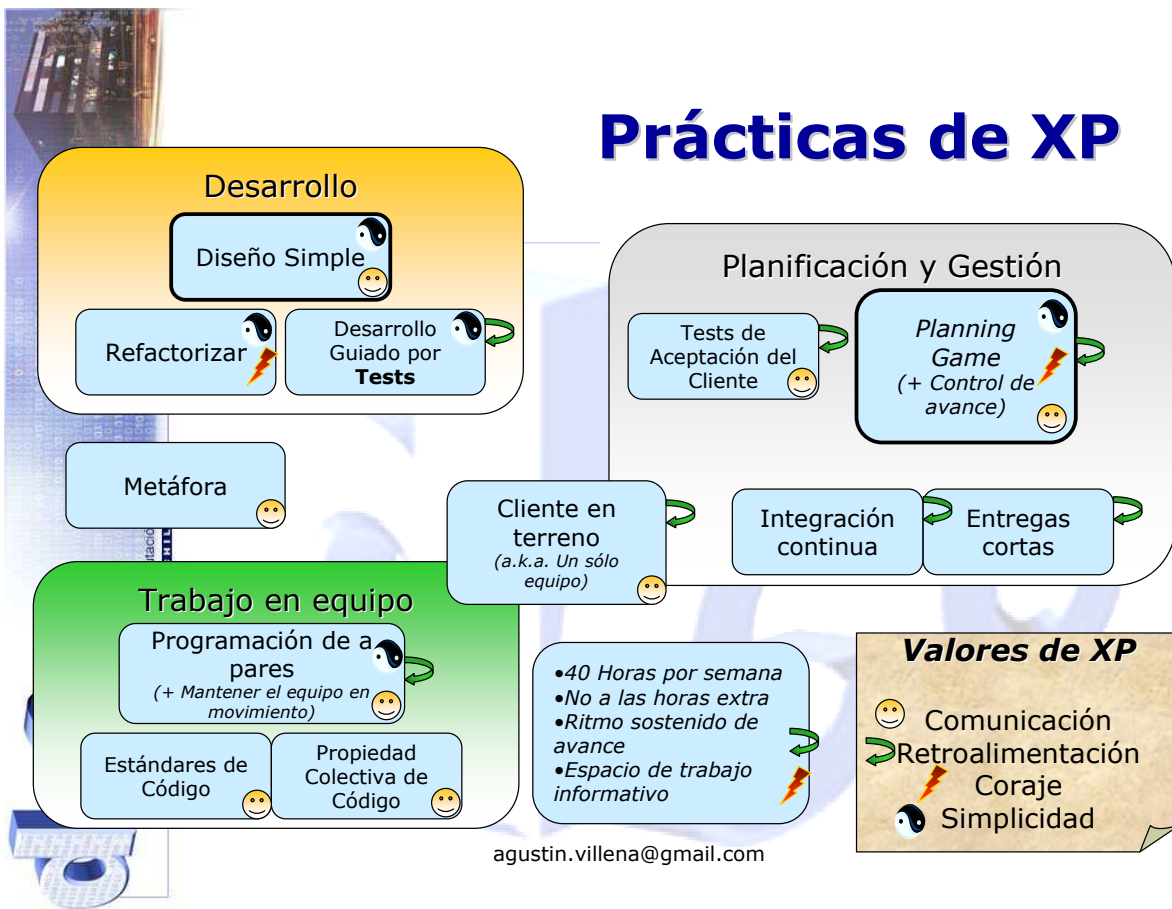
agustin.villena@gmail.com

XP Practices

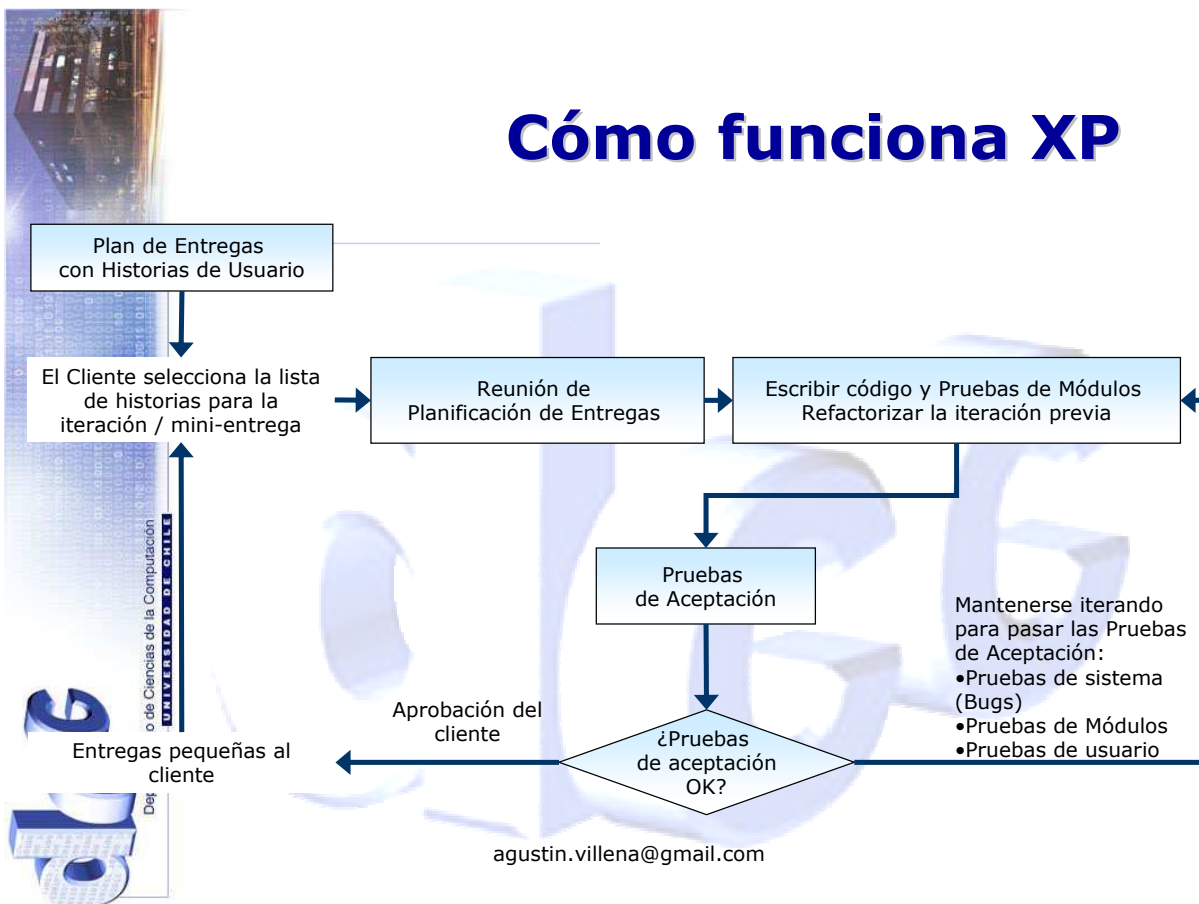


agustin.villena@gmail.com

Prácticas de XP



Cómo funciona XP





Prácticas de XP: Desarrollo

- **SimpleDesign**
 - **DoSimpleThings:** Implementar la solución más simple posible que pase el test
 - **YouArentGonnaNeedIt:** NO implementar para lo que "podría pasar"
 - **OnceAndOnlyOnce:** una funcionalidad debe estar implementada en un solo segmento de código
 - **Regla de oro:**
 - Lograr la implementación más simple que corra la suite de pruebas actual

agustin.villena@gmail.com



Prácticas de XP: Desarrollo: SimpleDesign

- ¿Qué define el mejor y más simple diseño? Se siguen 4 guías en orden de prioridad
 1. El sistema (código y pruebas incluidos) debe **comunicar todo lo que se desea comunicar**
 2. **No** debe existir **código duplicado**
 3. El sistema debe tener el **menor número de clases**
 4. El sistema debe tener el **menor número de métodos**

agustin.villena@gmail.com

Prácticas de XP: Desarrollo: SimpleDesign

- El rol de los diagramas
 - Si aplicamos principios XP
 - **Menor inversión inicial:** solo dibujar pocos diagramas a la vez
 - **Viajar ligero:** si no se puede sincronizar automáticamente con el código, el bosquejo se pasa a código y luego se bota
 - **Trabajar con los instintos:** No es algo obligatorio, pero debe estimularse en quienes se comunican bien con ellos

agustin.villena@gmail.com

Prácticas de XP: Desarrollo: Test First Development

- Tests son programados **antes** de programar la implementación. Luego de programar, se verifica la correctitud automáticamente
- Aumenta la confianza del desarrollador en su código
- Facilita el mejorar el código sin provocar errores en cascada
- Herramientas:
 - JUnit : Java
 - NUnit : .NET
 - PyUnit : Python
 - CPPUnit : C++
 - Etc.

```

30 import unittest
31
32 def calculateVerifier(rutNumber):
33     invertedRUT = rutNumber[::-1]
34     return (range(10) + ['K', 0]) [11 - sum(
35         [int(digit) * factor for digit, factor in zip(invertedRUT, 2
36             ) % 11]
37     )]
38
39 class rutTestCase(unittest.TestCase):
40     def testIsValid(self):
41         self.assertTrue(isValid("1-9"))
42         self.assertTrue(isValid("11.111.111-1"))
43         self.assertTrue(isValid("8.687.797-K"))
44         self.assertFalse(isValid("1-8"))
45
46     def testStandardize(self):
47         self.assertEqual(standardize("11.111.111-1"), "11111111-1")
48         self.assertEqual(standardize("111111111"), "11111111-1")
49         self.assertRaises(ValueError, standardize, "1-")
50
51 if __name__ == "__main__":
52     unittest.main()
53

```

agustin.villena@gmail.com

Prácticas de XP: Desarrollo: Refactoring

● RefactorMercilessly

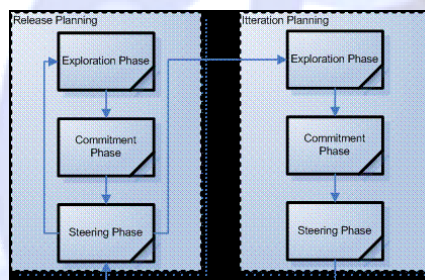
- Cuando dos segmentos de código hagan los mismo, se refactoriza para combinarlos y simplificar el código resultante... SIEMPRE
- Permite el llamado "Diseño Emergente", en contraste con "Diseño detallado"

agustin.villena@gmail.com

Prácticas de XP: Planificación y Gestión

● PlanningGame

- Con reglas claras y simples, permite definir qué se hará, y cuanto costará hacerlo
- Involucra a Clientes y Desarrolladores
- Permite llevar un registro claro de lo desarrollado



agustin.villena@gmail.com



Prácticas de XP: Planificación y Gestión

- **CustomerTests** (aka **AcceptanceTests**)
 - El Cliente define un conjunto de pruebas que el producto debe ser capaz de pasar, para ser aprobado
- **SmallReleases**
 - Cada iteración es muy corta, y sólo se crea el código para un muy pequeño conjunto de funcionalidades
- **ContinuousIntegration**
 - Generar ejecutables funcionales periódicamente (diariamente)
 - Depende de tener un sistema de pruebas continuo
- **StandUpMeeting**
 - Mini-reunión al comenzar la jornada, en que todos, de pie, definen lo que harán

agustin.villena@gmail.com



Prácticas de XP: Trabajo en equipo

- **PairProgramming**
 - Un computador, un teclado, un mouse, y dos programadores
 - Empírico
 - código un 20% más simple
 - 15% adicional de productividad sobre el trabajo separado
- **CollectiveCodeOwnership:** todos son propietarios (y responsables) de todo el código
- **CodingStandards:** Usar convenciones que permitan que cualquier desarrollador pueda entender, usar y modificar el código desarrollado por el equipo

agustin.villena@gmail.com



Prácticas de XP: *OnsiteCustomer*

- Tener todos los involucrados en el mismo lugar, en el mismo horario
 - ¡Incluyendo al cliente!
- El cliente estará disponible para
 - Responder dudas en el momento que ocurran
 - Entregar retroalimentación acerca del trabajo
 - "esta ok"
 - "hay que cambiar esto"
 - "necesito canjear esta funcionalidad por esta otra"
- Así nos aseguramos de que el cliente **no tenga sorpresas**
 - las que casi siempre son desagradable)

agustin.villena@gmail.com



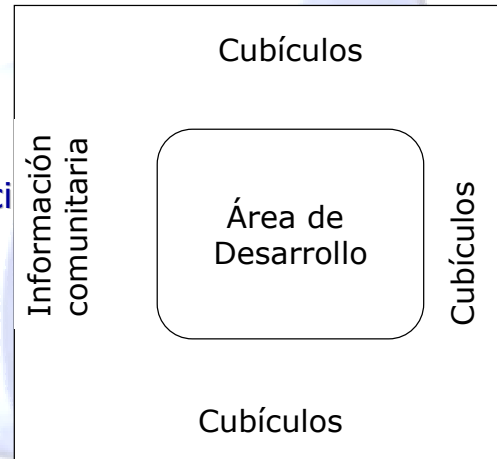
Prácticas de XP: **Productividad Sostenida en 40 hrs x semana**

- Los equipos XP trabajan duro, y a un ritmo que puede sostenerse indefinidamente. Esto significa:
 - Se trabaja sobretiempo sólo cuando es efectivo, pero se entiende como algo extraordinario
 - Un proceso ordenado saca el mejor provecho del tiempo ya asignado
 - Un desarrollador cansado no produce más que uno descansado
- También llamado "No Overtime"

agustin.villena@gmail.com

Prácticas de XP: Informative Workspace

- Planta Abierta, sin grandes muros
- En la periferia, mini-cubículos para trabajo individual y pizarras
- En el centro
 - PC's más poderosos
 - Dos puestos por PC
- "radiadores de información"
 - Historias de Usuario
 - Métricas de Avance



agustin.villena@gmail.com

Gestión de un proyecto XP

- ¿Cómo alcanzar el justo medio entre...?
 - Gestión centralizada
 - Caos
- Se definen dos roles
 - **Coach** (Entrenador)
 - **Tracker** (Medidor de avance)

... pueden ser cumplidos por la misma persona

agustin.villena@gmail.com



Gestión de un proyecto XP

Rol del Entrenador

- Ayudar al resto a hacer un **trabajo aún mejor**
- Que **todos** tomen buenas decisiones
- Estar disponible para ser **socio de desarrollo** (en especial de los novicios)
- Vislumbrar **objetivos de refactorización de gran escala**, y promover las refactorizaciones pequeñas que ayuden
- Ayudar a los programadores con destrezas individuales
 - Testing,
 - Formateo y estandarización de código
 - refactorización
- Explicar el proceso a gerentes de mayor nivel
- Proveer de juguetes y comida ☺

agustin.villena@gmail.com



Gestión de un proyecto XP

Rol del Medidor de Avance

- Usar métricas para medir el avance del proyecto (solo las indispensables)
 - Por ejemplo:
Tiempo de desarrollo / Tiempo calendario
- Actualizarlas mínimo 2 veces por semana
- Mostrar la información en una **Carta Grande y Visible** para todo el equipo
- Ayudar al Entrenador a
 - Motivar al cambio de una manera gentil y no coercitiva

agustin.villena@gmail.com



Gestión de un proyecto XP

Cómo interviene la gestión un proyecto XP

- Es importante reaccionar ante los problemas apenas aparecen
- Se reflexiona acerca de qué provocó el problema
- Si hay que tomar medidas impopulares, hacerlo firme pero humildemente
- Ejemplos de intervención
 - Cambios de personal
 - En las estrategias de desarrollo
 - *Matar* el proyecto

agustin.villena@gmail.com



¿Cómo adoptar XP?

- Se sugiere adoptar una práctica a la vez
- Seguir el siguiente algoritmo
 1. Elegir el peor problema
 2. Resolverlo al estilo XP
 3. Cuando no sea el peor problema, partir de nuevo
- Y puede refinarse
 - 1. Reacomodar todo para programar de pares y que el cliente pueda sentarse con uno
 0. Comprar algunos *snacks* ☺

agustin.villena@gmail.com



(Algunos) Recursos Enlaces de Internet

- WikiWiki entry point for XP
 - <http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap>
- Extreme programming
 - www.extremeprogramming.org
- Gestión Ágil de Software
 - www.agilemanagement.net
- Lean Software Development
 - www.poppendieck.com
- Agile Alliance
 - www.agilealliance.org

agustin.villena@gmail.com



(Algunos) Recursos Bibliografía

- **Lean Software Development**
Mary Poppendieck
- **Extreme Programming Explained: Embrace Change**
Kent Beck
- **Extreme Programming Installed**
Ron Jeffries, Ann Anderson, Chet Hendrickson
- **Extreme Programming Applied: *Playing to Win***
Ken Auer and Roy Miller
- **Extreme Programming in Practice**
Robert C. Martin, James W. Newkirk

agustin.villena@gmail.com