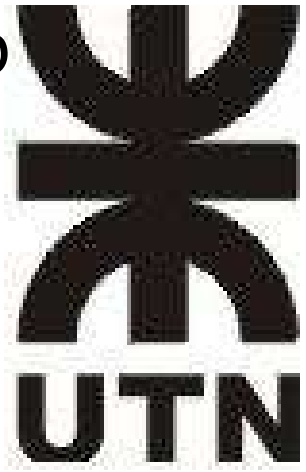
Arquitectura de Proyectos de IT

Metodologías Iterativas de Desarrollo



Lic. Carlos Leone (MBA)
Ing. Nicolás Passerini
Ing. Gustavo A. Brey

Agenda

#	Tema
1	Introducción a Metodologías de Desarrollo
2	Tipos de Metodología
3	Metodologías orientadas a Iteraciones
4	Metodologías Agiles
5	Rol del arquitecto en un proyecto iterativo

Introducción a Metodologías de Desarrollo

- Una metodología define:
  - Estados, etapas o fases de un desarrollo, junto con los criterios de transición entre elllos.
  - Tareas, actividades, etc.
  - Roles, con sus skills necesarios y las interacciones entre ellos.
  - Artefactos o entregables.
  - Herramientas de control, seguimiento, medición y perfeccionamiento.
  - Principios, criterios para tomar decisiones, estrategias para manejar distintos tipos de situaciones, herramientas de manejo de riesgos, etc.

- No debe confundirse metogología con ciclo de vida.

Agenda

#	Tema
1	Introducción a Metodologías de Desarrollo
2	Tipos de Metodología
3	Metodologías orientadas a Iteraciones
4	Metodologías Agiles
5	Rol del arquitecto en un proyecto iterativo

Diferentes tipos de Metodología

- **Cascada** vs. **Iterativo**.
  - Decision Momentum
- **Procesos** bien definidos vs. aprovechamiento de las habilidades y las interacciones entre las **personas**.
  - Brain Work vs. No Brain Work
  - Procesos definidos vs. Procesos intuitivos o internalizados
- Procesos **repetibles** vs. adaptativos, **eficiencia**, mejora continua.
- Cuánto y qué tan formalmente **documentar**.

Agenda

#	Tema
1	Introducción a Metodologías de Desarrollo
2	Tipos de Metodología
3	Metodologías orientadas a Iteraciones
4	Metodologías Agiles
5	Rol del arquitecto en un proyecto iterativo

Metodologías Iterativas - Desripción

- El proyecto se divide en "iteraciones", cuyo entregable es una versión del sistema.
- **Todo está sujeto a ser modificado** en las iteraciones posteriores (planificación, análisis, diseño, código, etc).
- En lugar de poner el énfasis en la eliminación de los **errores**, se procura **minimizar** su **impacto**.
- Se intenta **aprovechar el aprendizaje** durante el desarrollo.
- Ideales para cuando los **requerimientos** no están del todo claros en un comienzo o pueden sufrir modificaciones.
- No confundir iterativo e incremental. Lo iterativo no presupone o incremental, ni viceversa.

Metodologías Iterativas - Motivaciones

- En la actualidad, con frecuencia la extracción de requerimientos es más costosa que la construcción.
- *Dinamismo* o *"apuro"*. Muchas veces los requerimientos no están definidos desde el comienzo, cambian durante el desarrollo, pueden cometerse errores.
- La complejidad en muchos sistemas hoy en día pasa más por las interfaces de usuario que por la lógica de negocio.
- No es tan sencillo manejar los requerimientos independientemente de las posibilidades o restricciones tecnológicas.

Metodologías Iterativas

Ventajas

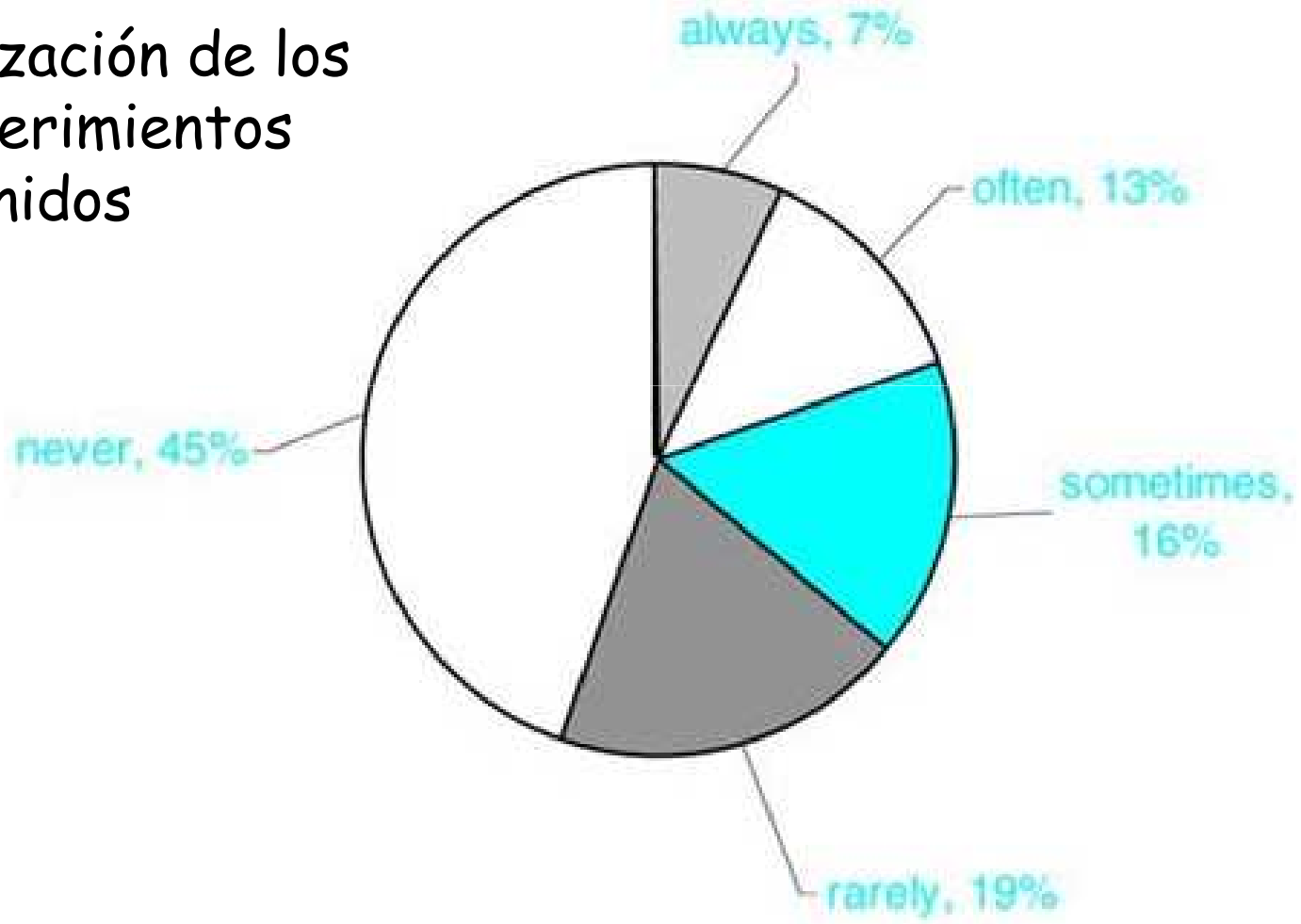
- Más "realista"
- Se adaptan mejor a los cambios
- Permiten administrar mejor los riesgos
- Mayor visibilidad para el cliente
- Mejor feedback para el equipo de desarrollo

Desventajas

- Falta de experiencia
- Reacción al cambio de enfoque
- Requieren de distintos modelos de contrato.

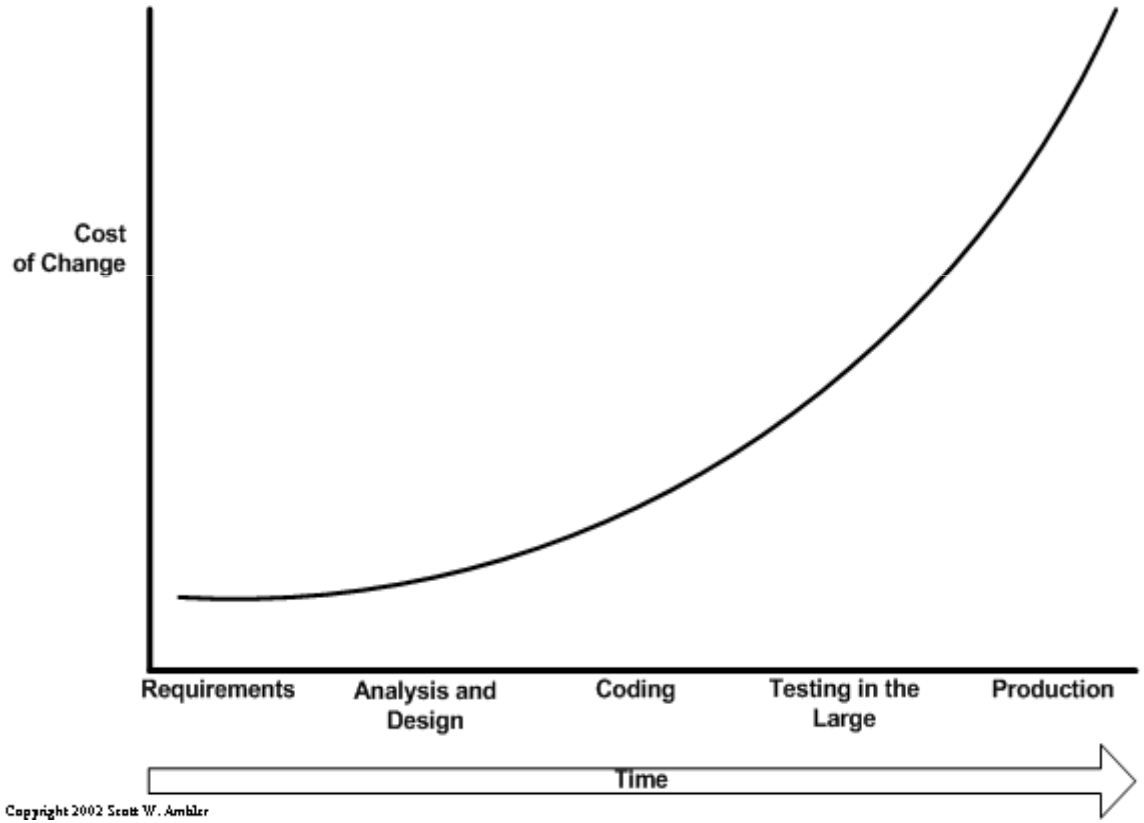
Problemas de las metodologías secuenciales

Utilización de los requerimientos definidos



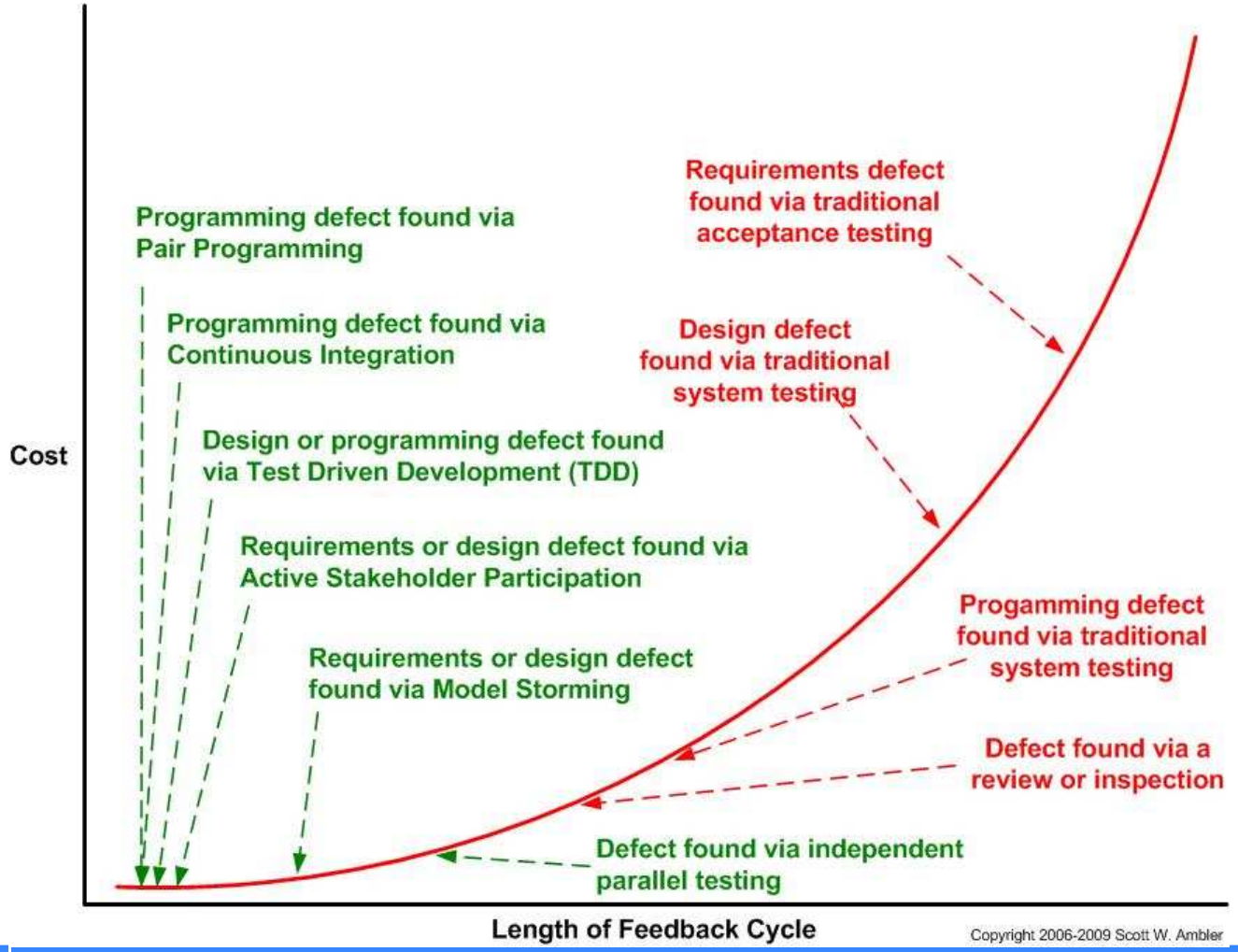
Problemas de las metodologías secuenciales

Relación del costo de un cambio con metodologías secuenciales



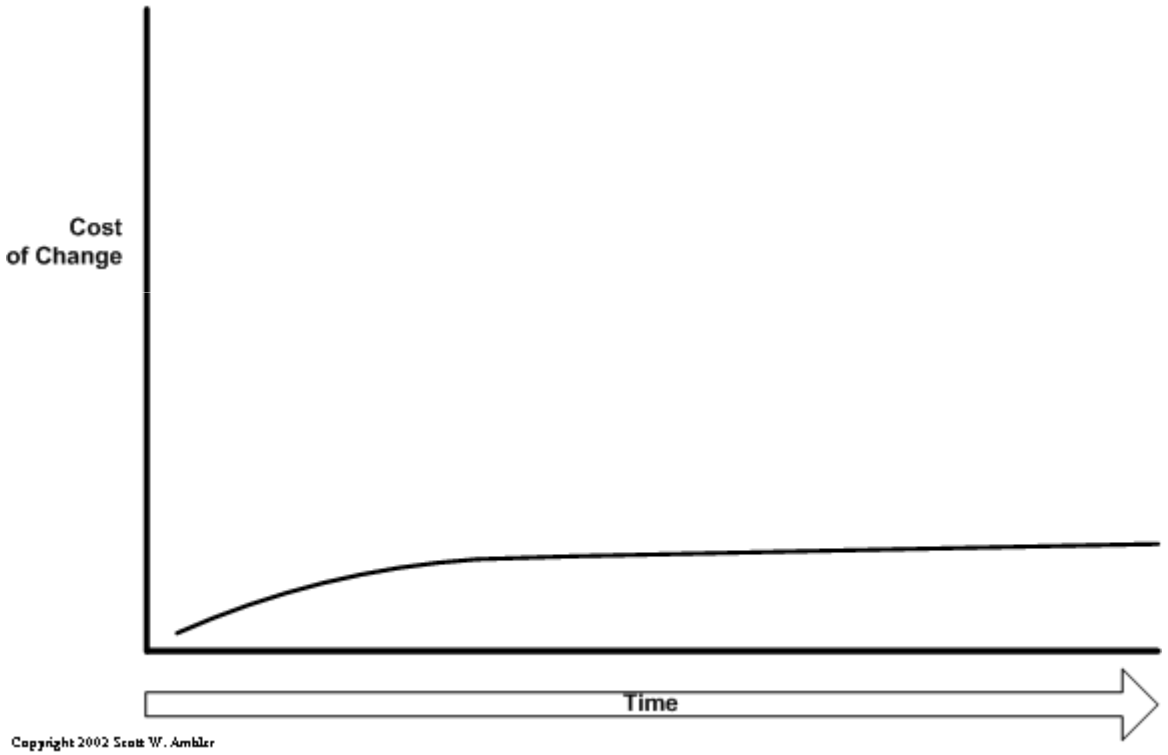
Problemas de las metodologías secuenciales

Como aplanar la curva utilizando técnicas ágiles



Problemas de las metodologías secuenciales

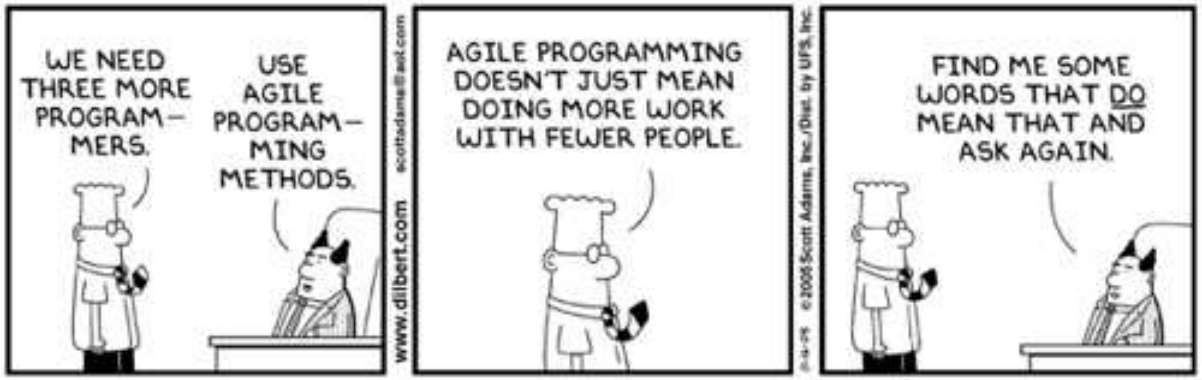
Curva según Ken Beck (Creador de la metodologías ágil XP)



Agenda

#	Tema
1	Introducción a Metodologías de Desarrollo
2	Tipos de Metodología
3	Metodologías orientadas a Iteraciones
4	Metodologías Agiles
5	Rol del arquitecto en un proyecto iterativo

Que no es agile!



© Scott Adams, Inc./Dist. by UFS, Inc.

© Scott Adams, Inc./Dist. by UFS, Inc.

Principios de las metodologías ágiles

A los **individuos y su interacción**, por encima de los procesos y las herramientas.

El **software que funciona**, por encima de la documentación exhaustiva.

La **colaboración con el cliente**, por encima de la negociación contractual.

La **respuesta al cambio**, por encima del seguimiento de un plan.

Características de las metodologías ágiles

- Son **iterativas**. No intentan minimizar los cambios, sino estar preparados para aceptarlos.
- Son **adaptativas** en lugar de repetibles.
- Priorizan a los **individuos** y a las **interacciones** por sobre los procesos.
- Incorporan **feedback** sobre el proceso. Priorizan la **colaboración** con el cliente.
- Buscan **minimizar el overhead** metodológico.

Manejo de Riesgos en Metodologías Ágiles

- **Retrasos y cancelaciones**
  - Realizar releases frecuentemente, tomar el más pequeño release posible.
  - Priorizar las tareas que agregan mayor valor al cliente (vs. reducir riesgos).
- **Altos costos de mantenimiento**
  - Énfasis en la calidad, simplicidad y modificabilidad del diseño.
  - Los tests unitarios automáticos aseguran una base de calidad y facilitan los tests de regresión.
- **Gran cantidad de defectos**
  - Test unitarios y de aceptación.
  - Feedback del usuario en todas las etapas.

Manejo de Riesgos en Metodologías Ágiles

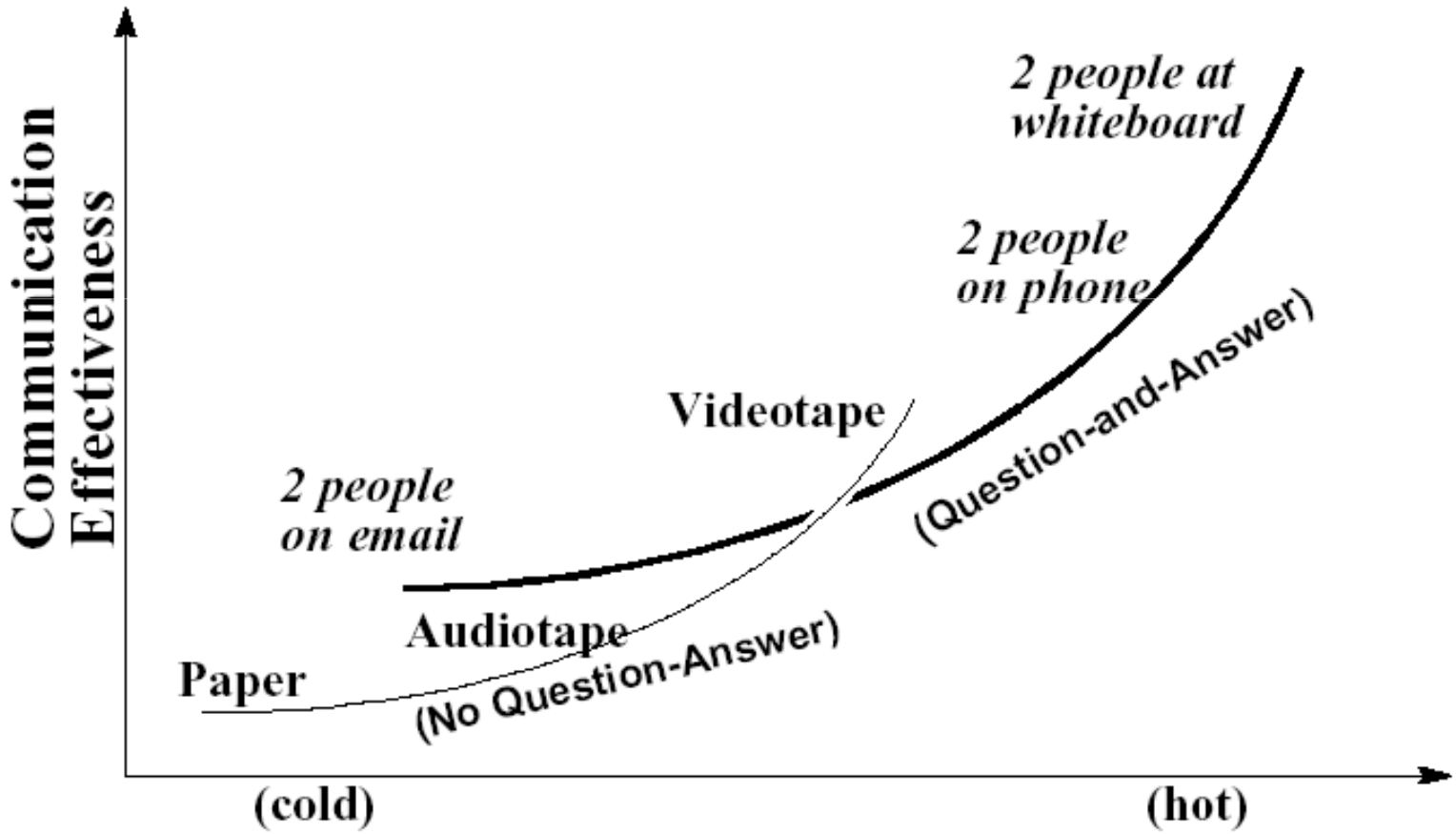
- **Falta de entendimiento del negocio**
  - El cliente integra el equipo de desarrollo en comunicación directa con los programadores.
  - Refinación continua de los requerimientos.
- **Cambios en el negocio**
  - Acortar el ciclo de releases.
  - Diseño flexible.
- **Requerimientos "volados"**
  - Sólo las funcionalidades más importantes se implementan.
  - Contacto contínuo con el cliente, que determina las prioridades.

Manejo de Riesgos en Metodologías Ágiles

- **Arquitectura y diseño "volados"**
  - Utilizar siempre el diseño más simple posible.
  - Refactoring.

- **Inestabilidad de personal**
  - Reglas claras.
  - Responsabilidades compartidas.
  - Incremento del contacto humano.
  - Feedback sobre las estimaciones.
  - "Sustainable Pace".
  - Pair programming

Comunicación



Especificación y verificación

- **Documento word**
  - La verificación se hace a mano
- **Lenguajes formales de especificación**
  - Verificados mediante razonamiento ecuacional
  - En algunos lenguajes estas especificaciones pueden introducirse como parte del lenguaje
  - Algunas pruebas pueden automatizarse
  - Contract Driven Development
- **El test como especificación**
  - Se verifica automáticamente al ejecutarlo.

¿Cuándo Elegir las Metodologías Ágiles?

- Los requerimientos son poco claros o altamente volátiles.
- El cliente entiende el proceso y está involucrado en el proyecto.
- Se cuenta con profesionales capacitados y competentes
- Se tienen canales ricos de comunicación
- El grupo de trabajo no es demasiado grande (~ 50)
- Se desea fomentar la mejora continua del proceso

Ejemplos

- Extreme Programming (XP)
- Scrum
- Crystal
- Dynamic Systems Development Methodology (DSDM)
- Rapid Application Development (RAD)
- Feature-driven development

Agenda

#	Tema
1	Introducción a Metodologías de Desarrollo
2	Importancia de la metodología, las personas y lo ingenieril
3	Metodologías orientadas a Iteraciones
4	Metodologías Ágiles
5	Rol del arquitecto en un proyecto iterativo

Rol del Arquitecto en un Proyecto Iterativo

- Toma de decisiones en todo momento
- Necesidad de construir arquitecturas que evolucionen
- Tiene sus propios pasos en la iteración
  - Entender/seleccionar los requerimientos
  - Crear/Actualizar la arquitectura
  - Comunicarla constantemente
  - Evaluarla con los stakeholders
  - Verificar su implementación
- Foco en la comunicación

Arquitectura de Proyectos de IT

Metodologías Iterativas de Desarrollo

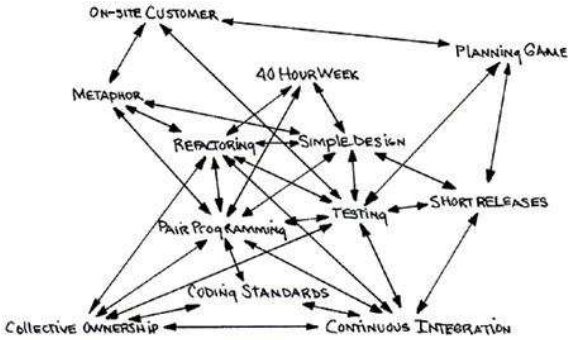
--Material Complementario--

Lic. Carlos Leone (MBA)
Ing. Nicolas Passerini
Ing. Gustavo A. Brey

Material Complementario

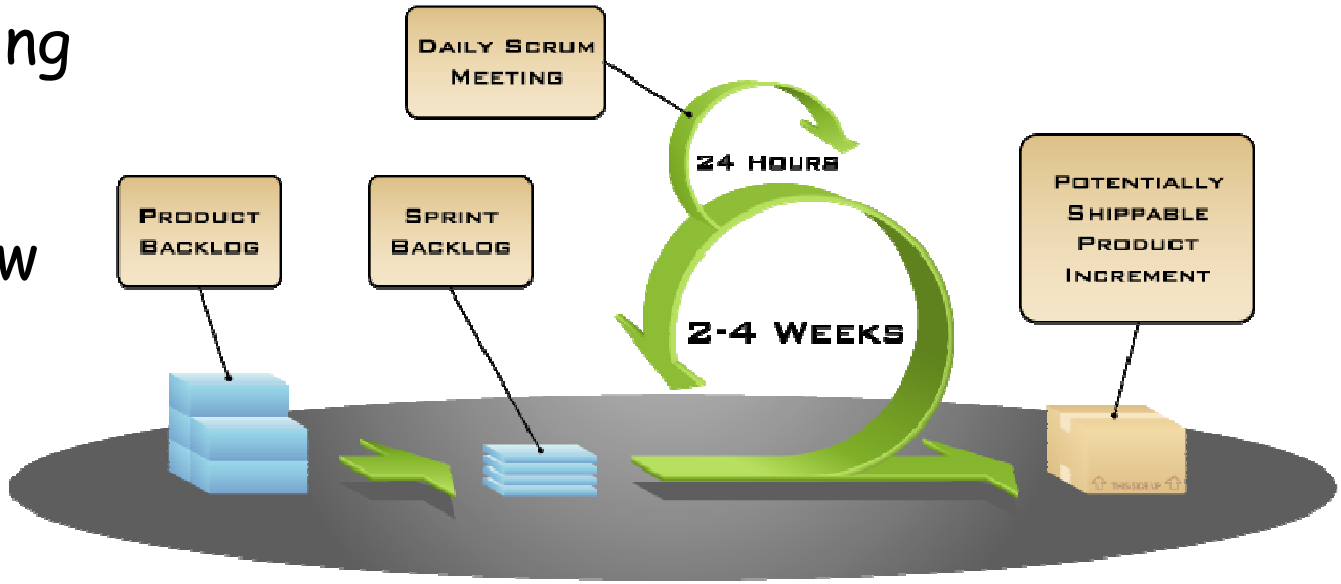
- Ejemplos de Metodologías Agiles
  - Scum

  - eXtreme Programming

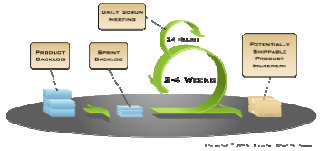


Scrum – Fases de un proyecto

- Planeamiento
- Arquitectura o diseño de alto nivel
- Desarrollo (sprints)
  - Sprint Planning
  - Daily Work
  - Sprint Review
- Cierre

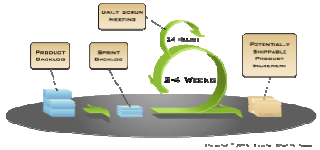


Scrum – Planeamiento



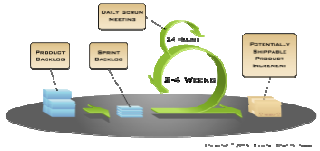
- Definir el backlog del producto.
- Determinar los próximos releases (objetivos y fechas).
- Determinar los objetivos y el equipo para el primer release.
- Analizar los riesgos.
- Revalidar o ajustar las herramientas e infraestructura.
- Estimar el costo del release.
- Verificar la aprobación del management.

Scrum - Arquitectura o diseño de alto nivel



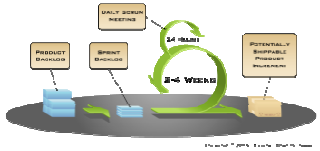
- Identificar los cambios necesarios para implementar el backlog.
- Extender o actualizar el modelo de dominio.
- Refinar la arquitectura para soportar los nuevos requerimientos.
- Plantear la estrategia para encarar cada item del backlog.
- Revisar cada diseño y reasignar tareas si es necesario.

Scrum – Desarrollo



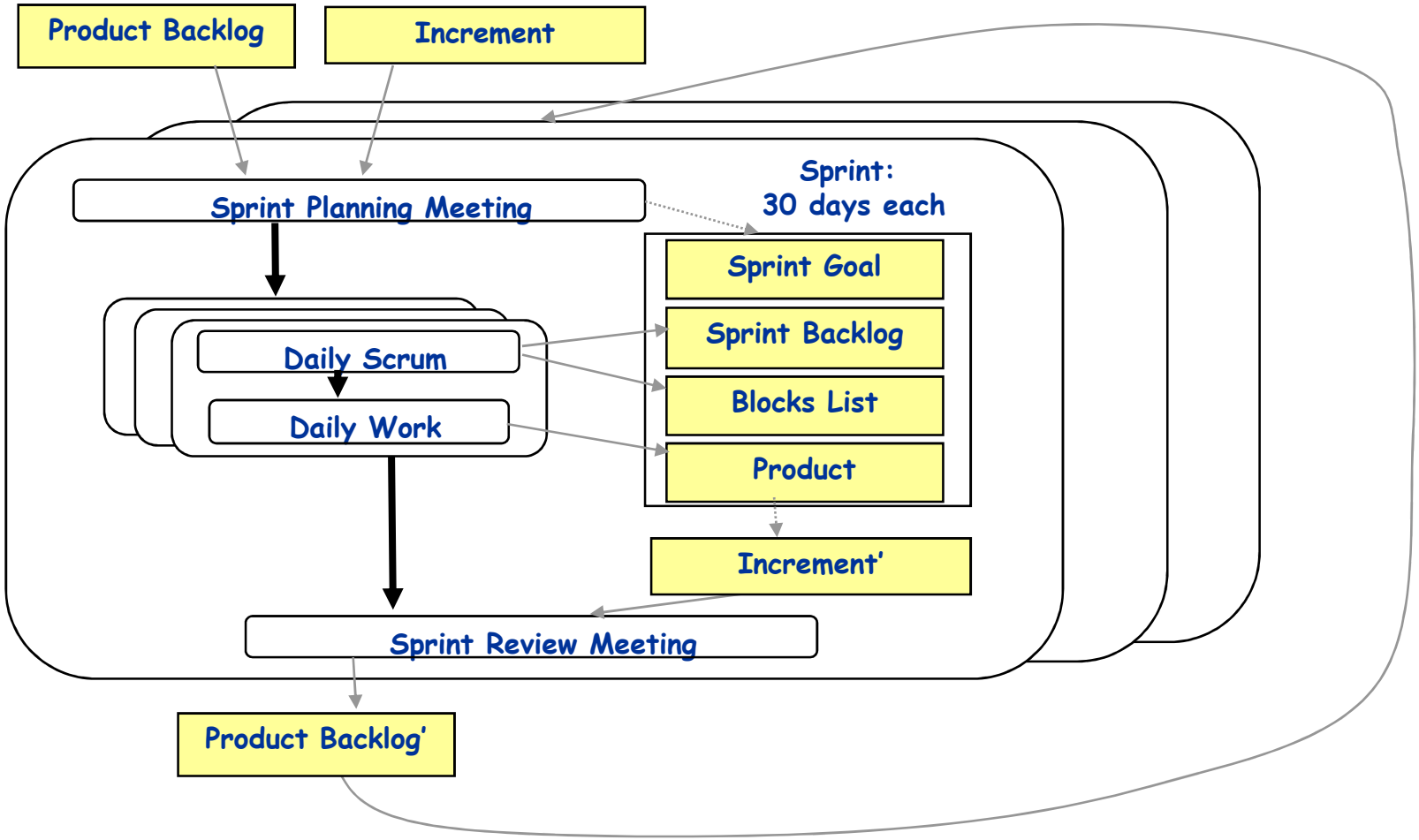
- Ciclo de desarrollo iterativo
- "Concurrent Engineering"
- Sprint Planning
- En cada sprint se realizan las siguientes tareas:
  - Desarrollo (análisis, diseño, programación, testing y documentación)
  - Empaquetado y despliegue
  - Revisión
  - Ajustes
- Sprint Review

Scrum – Cierre



- Se cierra el release.
- Pruebas de integración
- Pruebas de sistema
- Documentación para el usuario
- Preparación del material de entrenamiento

Scrum – Vista global del proceso



Scrum

Artefactos clave

- Product Backlog
- Sprint Goal
- Sprint Backlog
- Blocks List
- Increment

Roles

- Product Owner
- Scrum Master
- Team
- StakeHolders

Reuniones clave

- Sprint Planning
- Daily Scrum
- Sprint Review

Scrum – Reuniones

- Sprint Planning
  - Seleccionar los items de mayor prioridad en el Product Backlog
  - Determinar el Sprint Goal
  - El equipo determina el Sprint Backlog
- Scrum
  - Habla cada desarrollador del equipo:
    - ¿Qué hiciste ayer?
    - ¿Qué vas a hacer hoy?
    - ¿Qué cosas te traban?
  - Se actualiza el Sprint Backlog y el Blocks List
- Sprint Review
  - Demostración y discusión del incremento
- Las tres reuniones son manejadas por el ScrumMaster

Scrum – Artefactos

- **Backlog**: Funcionalidades aún no cubiertas en el release actual. Bugs, defectos, mejoras, etc.
- **Release**: Conjunto de items del backlog que representan un entregable con fecha.
- **Packet**: Conjunto de componentes u objetos que deben se modificados para implementar un item del backlog.
- **Changes**: Cambios que deben ocurrir para implementar un item del backlog.
- **Problems**: Problemas técnicos que deben ser resueltos para implementar un cambio.
- **Risks**.
- **Solutions**: Respuestas a los problemas y riesgos, generalmente resultando en cambios.
- **Issues**: Cualquier otra cuestión del general al proyecto no descripta en términos de paquetes, cambios y problemas.

Scrum – Ventajas

- Evita estancamientos en el proyecto
- Seguimiento del proyecto
- Seguimiento del equipo
- SW que incrementa su funcionalidad en cada sprint
- Mecanismos de control para variables cambiantes con el entorno
- Progreso en el producto con requerimientos inestables
- Aumenta comunicación con el equipo
- Cliente obtiene feedback frecuente sobre el producto

XP – Sus Valores

- Comunicación
- Feedback
- Simplicidad
- Coraje

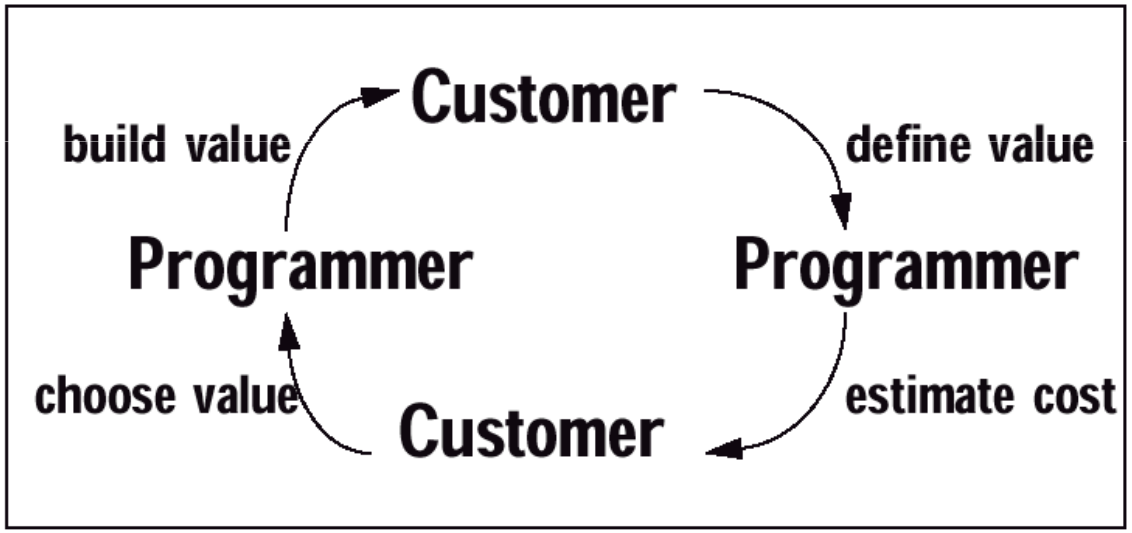


Figure 2.2 Programmer estimates, customer chooses.

XP – Estrategia de planificación

- **Planning Game**
  - El equipo de desarrollo estima
  - El cliente prioriza
- **Whole Team**
  - El cliente participa del equipo de desarrollo
  - Debe estar disponible siempre
- **Small Releases**
- **User stories**
  - El usuario…
  - Desea…
  - Para así poder…
- **Metrics**

XP – Estrategia de diseño

- **Principios**
  - Baja inversión inicial
  - Asumir simplicidad
  - Crecimiento gradual – pasos pequeños
  - No sobrediseñar ("travel light")

- **Prácticas**
  - System Metaphor
  - Simple Design
  - Refactoring
  - Test Driven Development

XP – Otras prácticas

- Pair Programming
- Collective Code Ownership
- Continuous Integration
- Coding Standards
- Sustainable Pace

XP – Fases del desarrollo

- Listening
- Testing
- Coding
- Refactoring

XP – Elementos de una iteración

- **Planning Game** – recolección de user stories
- **Estimación** – hecha por el desarrollador en base a analogía o prototipado
- **Metáfora** – vocabulario común del sistema
- **Diseño Simple** – en XP el código fuente es el diseño
- **Tests de aceptación**
- **Refactoring**
- **Integración**

XP - Críticas

- No escalable
- Altos riesgos si existen fallas en la arquitectura
- Altos riesgos si no hay capacidad/estabilidad en las personas
- La programación de a pares es un intento por solventar la falta de análisis
- Puede caer en el modelo de codificar y probar
- Vagas nociones de aseguramiento de la calidad
- Fuerte tendencia a no documentar

Bibliografía

- [PlanningXP] Kent Beck, Martin Fowler , Planning Extreme Programming. Addison Wesley, 2000, ISBN 0-201-71091-9.
- [XPExplained] Kent Beck, Extreme Programming Explained, Addison-Wesley, 1999, ISBN 0201616416
- [MythicalMM] Fred Brooks, The Mythical Man-Month, Addison-Wesley, 1995, ISBN 0201835959
- [PMMethodolgies] Jason Charvat, Project Management Methodologies, JOHN WILEY & SONS, INC., 2003, ISBN 0-471-22178-3
- [AbySoft] http://www.agilemodeling.com/essays/costOfChange.htm