Ingeniería del Software

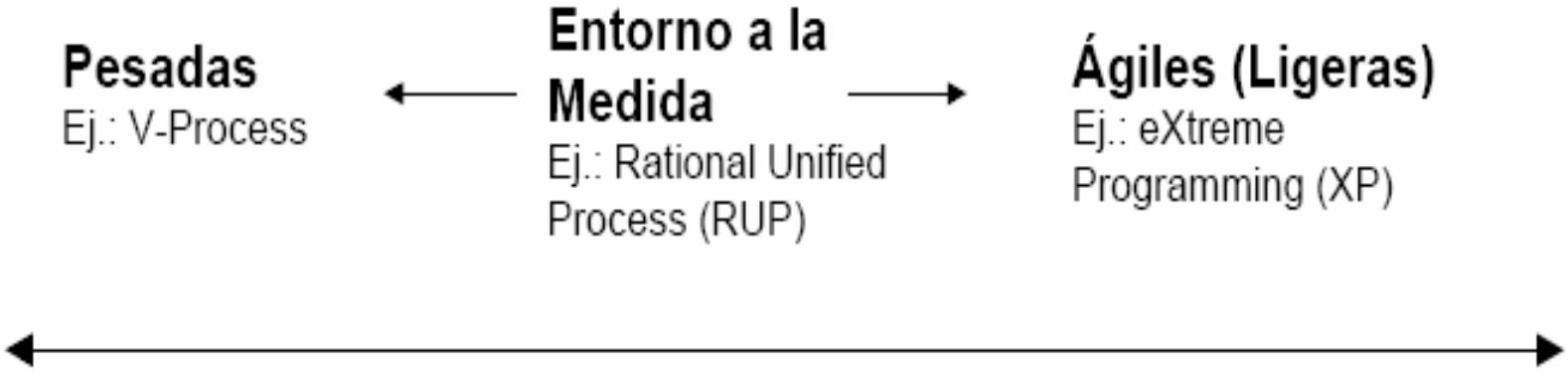
Unidad IV. Metodologías Agiles

Gabriela Arévalo
gabriela.arevalo@lifia.info.unlp.edu.ar

Contenido

- Procesos pesados y ligeros
- Introducción a las metodologías ágiles.
- Ubicación en el contexto de los metodologías disponibles.
- Principios de las metodologías ágiles.
- Metodologías ágiles: XP y SCRUM

Procesos Pesados y Ligeros



- Basado en Documentos
- Elaborar definiciones Flujo de Trabajo
- Muchos roles diferentes
- Muchos puntos de control
- Alto sobrecosto de gestión
- Mucha burocracia

- Foco en el código sobre el que se trabaja más que en la documentación
- Foco en la comunicación directa (entre desarrolladores y entre desarrolladores y el cliente).
- Bajo sobrecosto de gestión

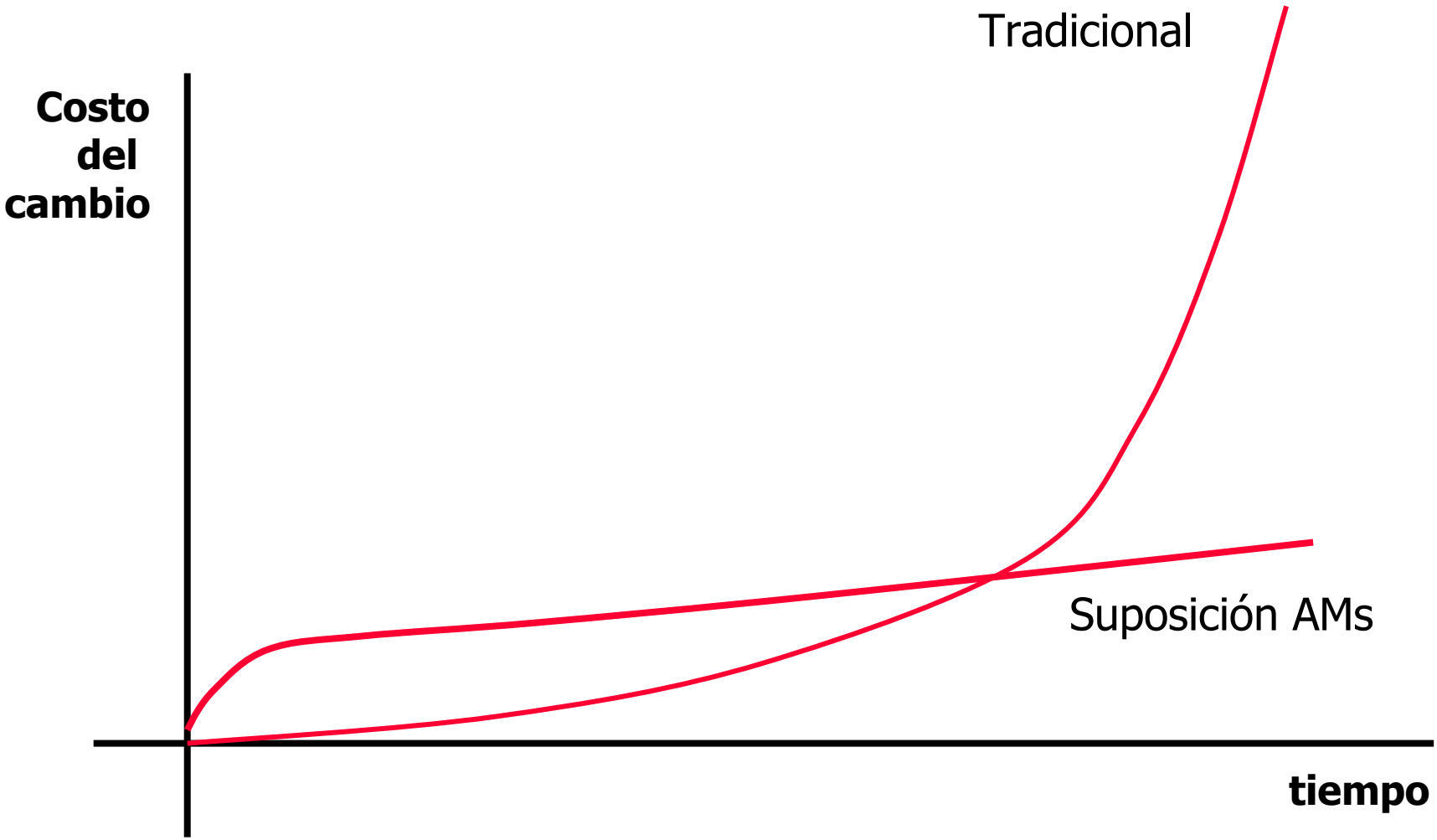
¿Qué es una Metodología Ágil?

- Las Metodologías Ágiles (AMs) valoran:
  - Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas
  - Desarrollar software que funciona más que conseguir una buena documentación $\Rightarrow$ Minimalismo respecto del modelado y la documentación del sistema
  - La colaboración con el cliente más que la negociación de un contrato
  - Responder a los cambios más que seguir estrictamente una planificación

¿Por qué surgen las Metodologías Ágiles?

- Dificultad para implantar metodologías tradicionales. Sofisticadas herramientas CASE y notaciones (UML)
- Una solución a medida para un segmento importante de proyectos de desarrollo de software
- Pugna entre comunidades/gurús
- "Aceptar el cambio" ...

Costo de los Cambios en SW



Manifiesto de las AMs

Principios

1. La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software que le reporte un valor

2. Dar la bienvenida a los cambios. Los AMs capturan los cambios para que el cliente tenga una ventaja competitiva

3. Entregar frecuentemente software que funcione, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente

Manifiesto de las AMs

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto

5. Construir proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir el trabajo

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo

7. El software que funciona es la medida principal de progreso

Manifiesto de las AMs

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad
10. La simplicidad es esencial
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos
12. En intervalos regulares, el equipo reflexiona respecto de cómo llegar a ser más efectivo, y según esto ajusta su comportamiento

Principales AMs

- Crystal Methodologies, Alistarir Cockburn, www.crystalmethodologies.org
- SCRUM, Ken Schwaber & Jeff Sutherland, www.controlchaos.com
- DSDM (Dynamic Systems Development Method), www.dsdm.org
- Lean Programming, Mary Poppendieck, www.poppendieck.com
- FDD (Feature-Driven Development), Peter Coad & Jeff De Luca, www.nebulon.com/fdd, www.coad.com/peter/#fdd
- Extreme Programming, Kent Beck www.extremeprogramming.org, www.xprogramming.com
- Adaptative Software Development, Jim Highsmith www.adaptivesd.com

eXtreme Programming

¿Qué es XP?

- Es una metodología ágil
  - Diseñada para entornos dinámicos
  - Pensada para equipos pequeños (hasta 10 programadores)
  - Orientada fuertemente hacia la codificación
  - Énfasis en la comunicación informal, verbal

Historia de XP

- Creado por Kent Beck para el grupo de desarrollo del proyecto C3 en Chrysler
  - Kent fue contratado para dirigir el proyecto
  - Durante el proceso nació una nueva metodología: eXtreme Programming (XP)
  - C3 concluyó exitosamente en 1997

Valores que fomenta XP

- Comunicación
- Simplicidad
- Retroalimentación
- Coraje

Roles XP

- Programador *(Programmer)*
  - Responsable de decisiones técnicas
  - Responsable de construir el sistema
  - Sin distinción entre analistas, diseñadores o codificadores
  - En XP, los programadores diseñan, programan y realizan las pruebas
- Jefe de Proyecto *(Manager)*
  - Organiza y guía las reuniones
  - Asegura condiciones adecuadas para el proyecto
- Cliente *(Customer)*
  - Es parte del equipo
  - Determina qué construir y cuándo
  - Establece las pruebas funcionales

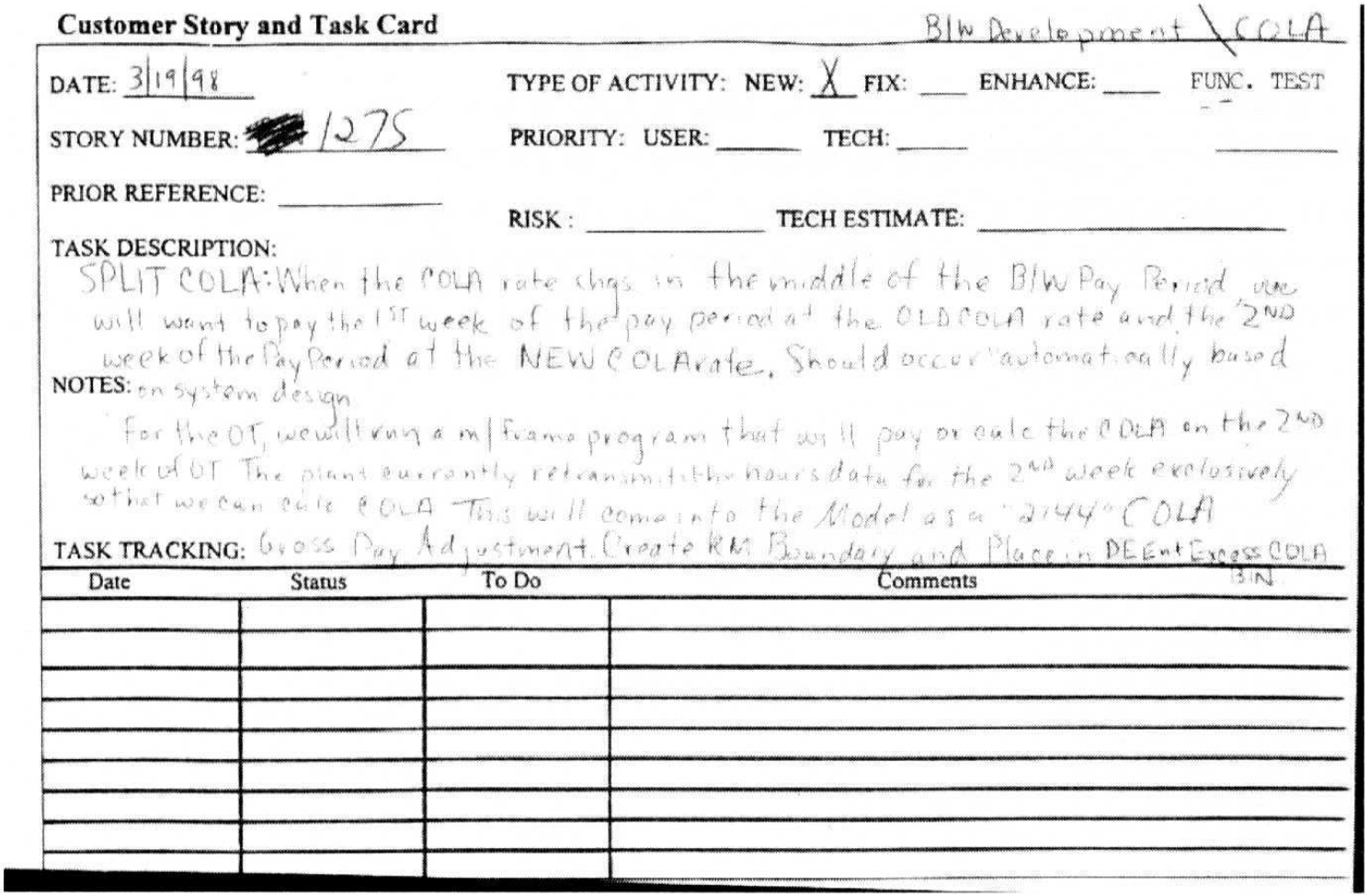
Roles XP

- Encargado de Pruebas (*Tester*)
  - Ayuda al cliente con las pruebas funcionales
  - Se asegura de que las pruebas funcionales se superan
- Rastreador (*Tracker*)
  - *Metric Man*
  - Observa sin molestar
  - Conserva datos históricos
- Entrenador *(Coach)*
  - Responsable del proceso
  - Tiende a estar en un segundo plano a medida que el equipo madura

Captura de Requisitos en XP

- **Historias del Usuario (*User-Stories*)**
  - Establecen los requisitos del cliente
  - Trozos de funcionalidad que aportan valor
  - Se les asignan tareas de programación con un nº de horas de desarrollo
  - Las establece el cliente
  - Son la base para las pruebas funcionales

Captura de Requisitos en XP



Customer Story and Task Card B/W Development / COLA

DATE: 3/19/98 TYPE OF ACTIVITY: NEW: X FIX: ___ ENHANCE: ___ FUNC. TEST

STORY NUMBER: [redacted] 1275 PRIORITY: USER: ___ TECH: ___

PRIOR REFERENCE: ___ RISK : ___ TECH ESTIMATE: ___

TASK DESCRIPTION:

SPLIT COLA: When the COLA rate chgs in the middle of the B/W Pay Period we will want to pay the 1ST week of the pay period at the OLD COLA rate and the 2ND week of the Pay Period at the NEW COLA rate. Should occur automatically based on system design.

NOTES: For the OT, we will run a mainframe program that will pay or calc the COLA on the 2ND week of OT. The plant currently retains in the hours data for the 2ND week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA

TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEEnt Excess COLA B/N

Date	Status	To Do	Comments

FIGURE 6. A story card

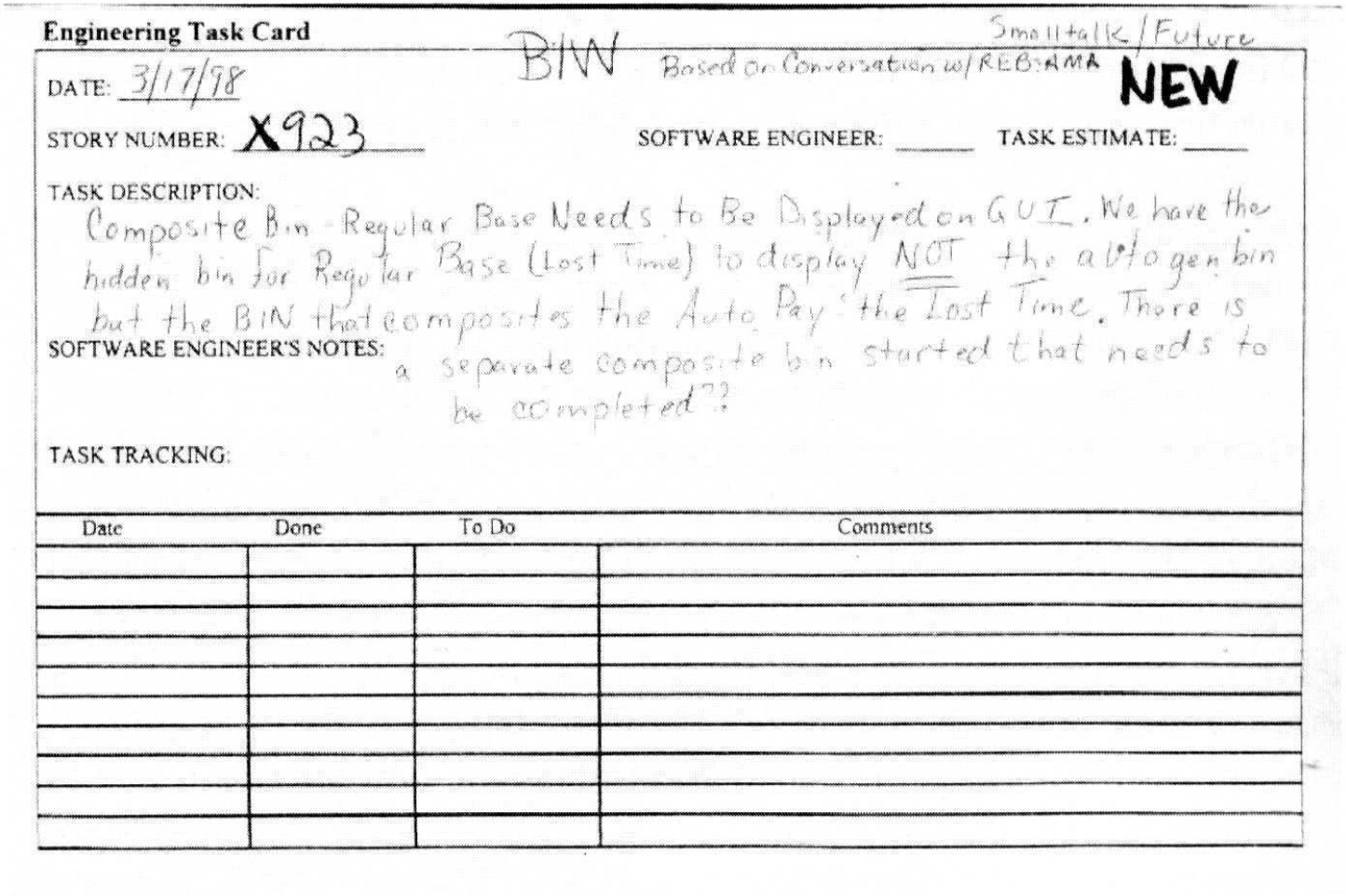
Planificación en XP

- Planificación por entregas (*releases*)
- Se priorizan aquellas user-stories que el cliente selecciona porque son más importantes para el negocio
- Entregas:
  - Son lo más pequeñas posibles
  - Se dividen en iteraciones (iteración = 2 o 3 semanas)
  - Están compuestas por historias
- A cada programador se le asigna una tarea de la user-story

Programación en XP

- La programación de tareas se realiza por parejas
- La pareja diseña, prueba, implementa e integra el código de la tarea
- Código dirigido por las pruebas
- Código modular, intentando refactorizar siempre que se pueda

Programación en XP



Engineering Task Card

BIW — Based on Conversation w/ REBECCA — Smalltalk / Future — NEW

DATE: 3/17/98

STORY NUMBER: X923 — SOFTWARE ENGINEER: _____ — TASK ESTIMATE: _____

TASK DESCRIPTION:
Composite Bin - Regular Base Needs to Be Displayed on GUI. We have the hidden bin for Regular Base (Last Time) to display NOT the auto gen bin but the BIN that composites the Auto Pay the Last Time. There is a separate composite bin started that needs to be completed?

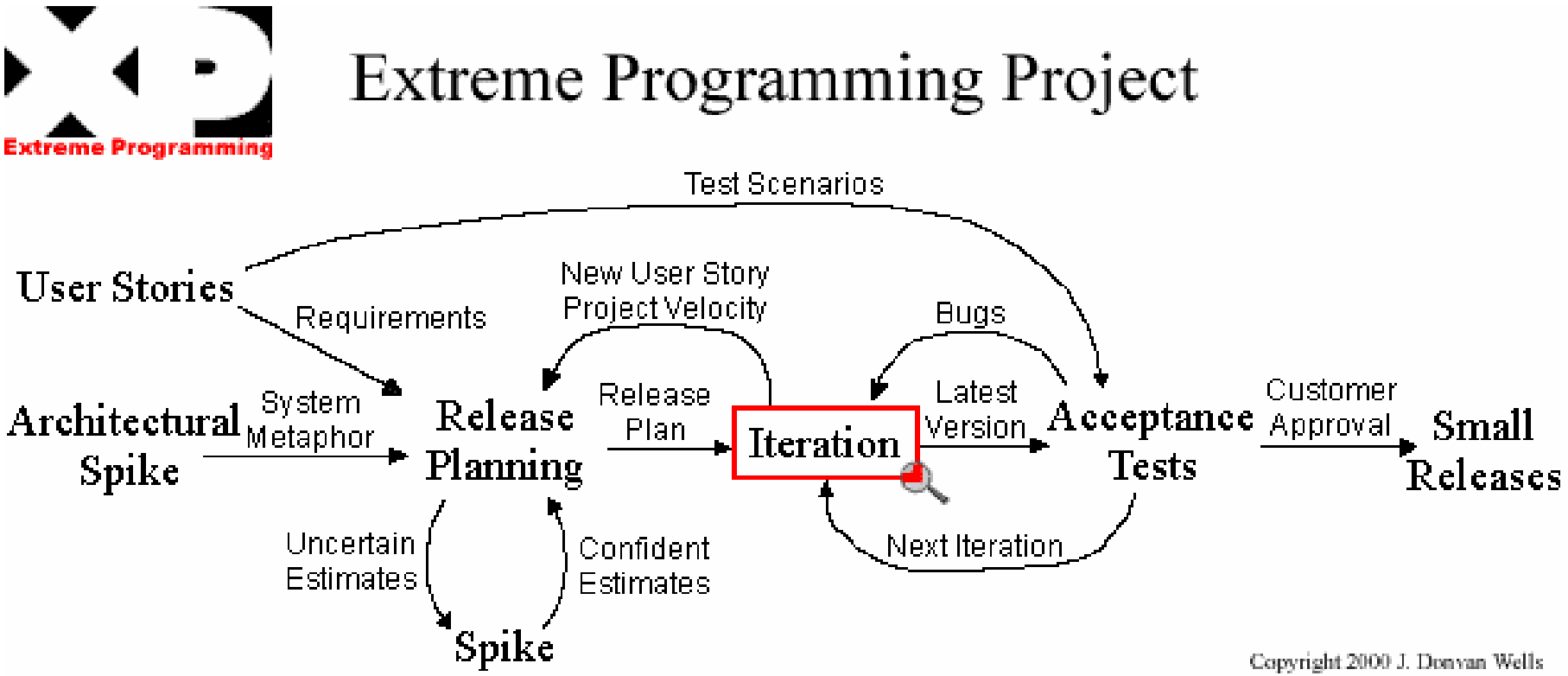
SOFTWARE ENGINEER'S NOTES:

TASK TRACKING:

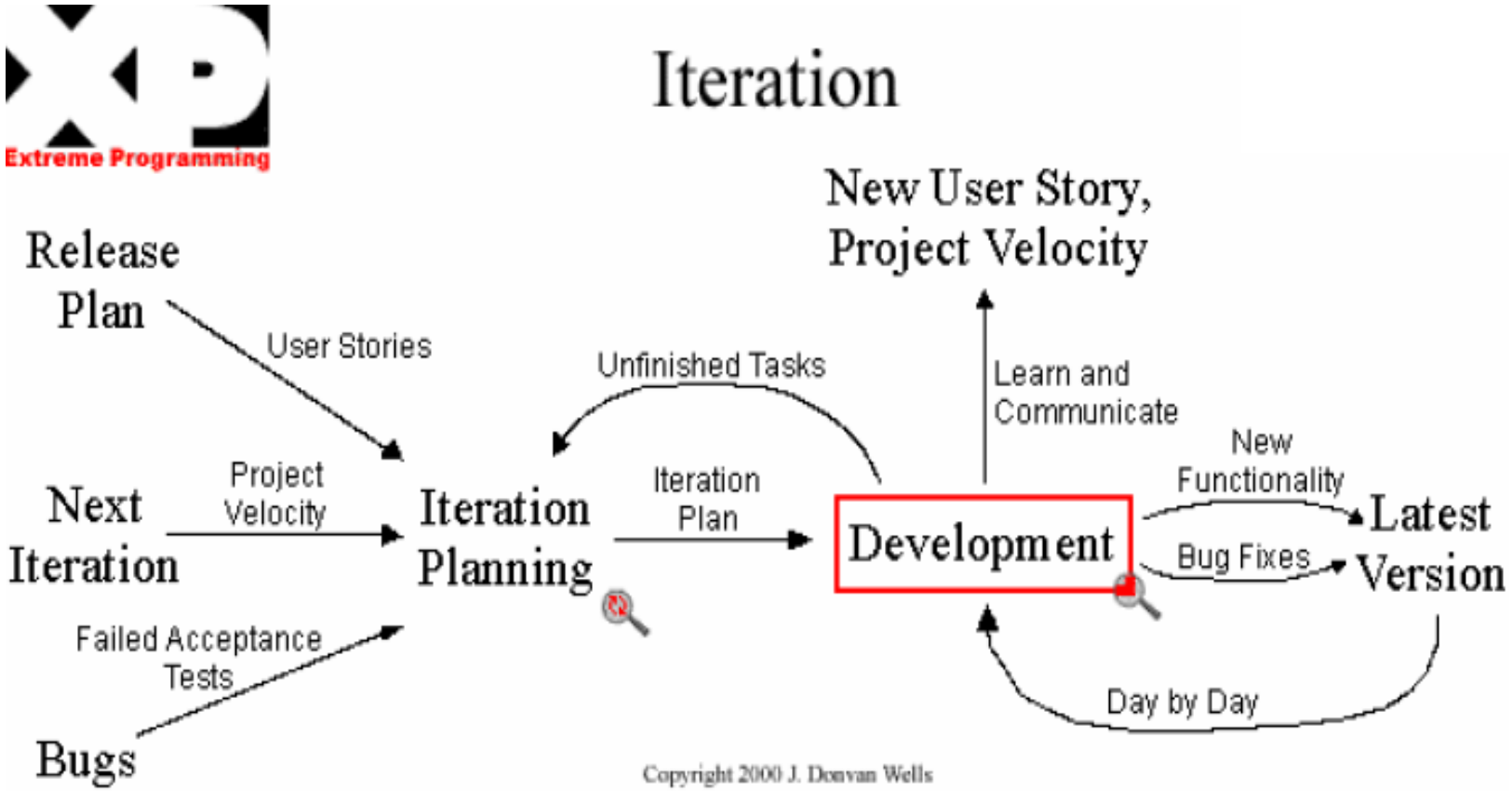
Date	Done	To Do	Comments

FIGURE 7. A task card

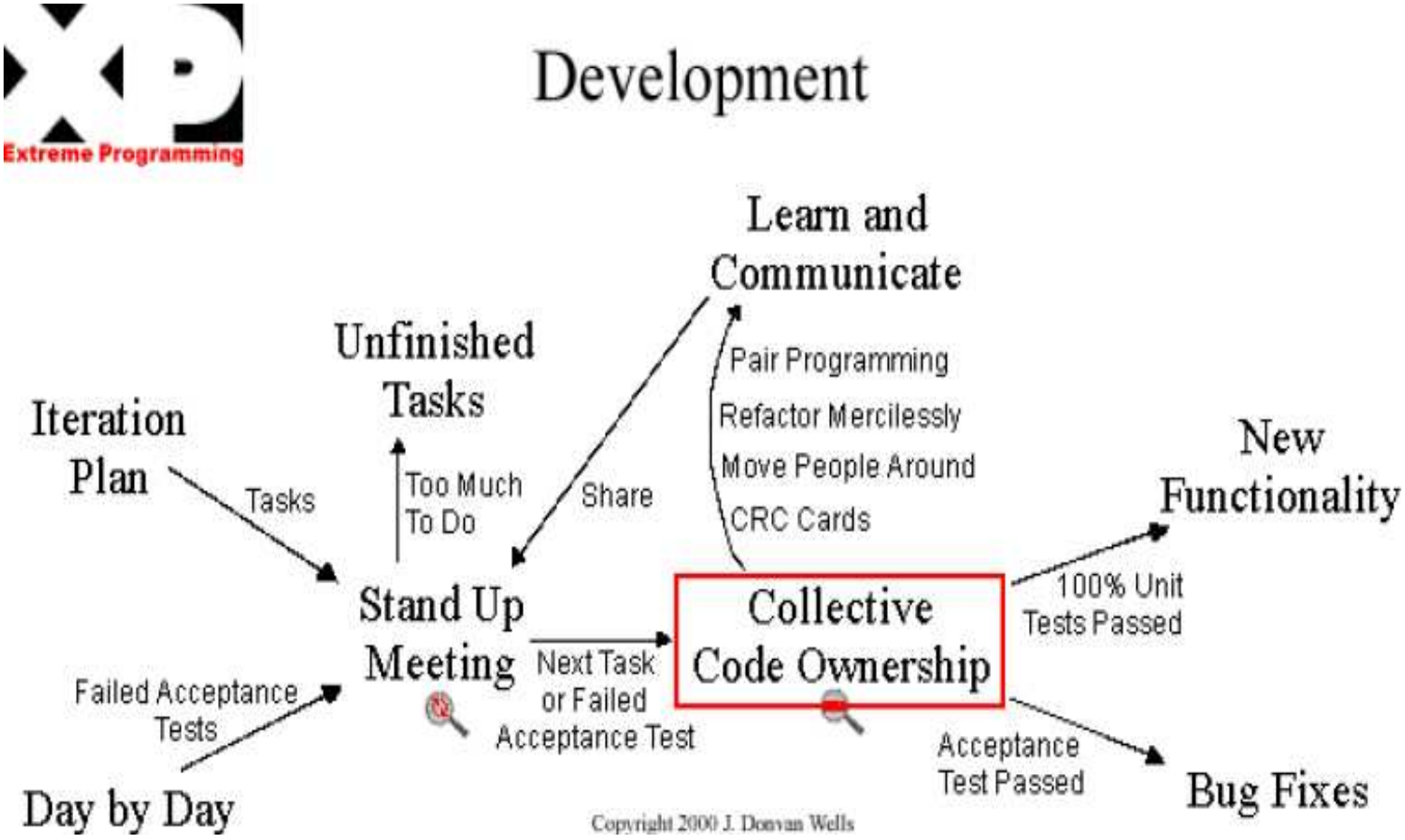
El Ciclo XP


Extreme Programming
Extreme Programming Project
Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural Spike
System Metaphor
Release Planning
Release Plan
Iteration
Latest Version
Acceptance Tests
Customer Approval
Small Releases
Uncertain Estimates
Confident Estimates
Spike
Next Iteration
Copyright 2000 J. Donvan Wells

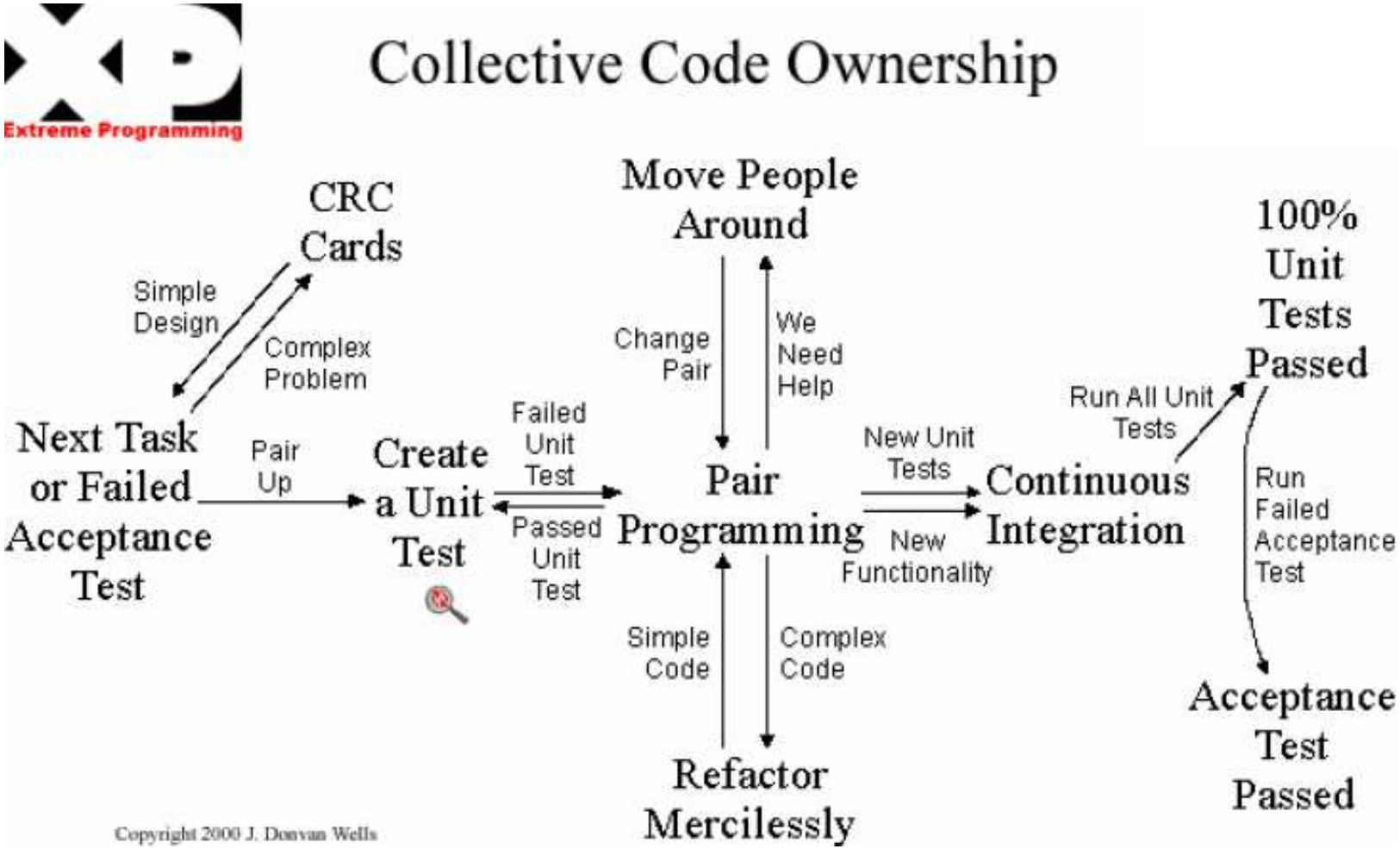
El Ciclo XP


Extreme Programming
Iteration
New User Story,
Project Velocity
Release
Plan
User Stories
Unfinished Tasks
Learn and
Communicate
New
Functionality
Project
Velocity
Iteration
Plan
Next
Iteration
Iteration
Planning
Development
Latest
Version
Bug Fixes
Failed Acceptance
Tests
Day by Day
Bugs
Copyright 2000 J. Donvan Wells

El Ciclo XP


Extreme Programming
Development
Learn and Communicate
Unfinished Tasks
Pair Programming
Refactor Mercilessly
Move People Around
CRC Cards
Iteration Plan
Tasks
Too Much To Do
Share
New Functionality
Stand Up Meeting
Collective Code Ownership
100% Unit Tests Passed
Next Task or Failed Acceptance Test
Failed Acceptance Tests
Acceptance Test Passed
Day by Day
Bug Fixes
Copyright 2000 J. Donvan Wells

El Ciclo XP


Extreme Programming
Collective Code Ownership
CRC Cards
Simple Design
Complex Problem
Next Task or Failed Acceptance Test
Pair Up
Create a Unit Test
Failed Unit Test
Passed Unit Test
Move People Around
Change Pair
We Need Help
Pair Programming
Simple Code
Complex Code
Refactor Mercilessly
New Unit Tests
New Functionality
Continuous Integration
Run All Unit Tests
100% Unit Tests Passed
Run Failed Acceptance Test
Acceptance Test Passed
Copyright 2000 J. Donvan Wells

Prácticas XP

- El juego de la planificación
- Entregas pequeñas
- Metáfora
- Diseño simple
- Pruebas
- Refactoring
- Programación en parejas
- Propiedad colectiva
- Integración contínua
- Semana de 40 horas
- Cliente in situ
- Estándares de programación

El Juego de la Planificación

- Decisiones de negocio (cliente):
  - **Alcance** → ¿Cuándo debe estar listo el producto para que sea valioso en producción?
  - **Prioridad** → Prioriza la incorporación de las user-stories
  - **Composición de entregas** → ¿Qué se necesita para que el negocio sea mejor antes de tener el sw?
  - **Fechas de entrega** → Fechas cuando el software funcionando causaría una gran diferencia

El Juego de la Planificación

- Decisiones técnicas (programadores y otros):
  - **Estimaciones** → ¿Cuánto tiempo tardará en implementarse una user-story?
  - **Consecuencias** → Tener en cuenta las consecuencias técnicas de determinadas decisiones de negocio
  - **Proceso** → Organización del proceso y el equipo
  - **Planificación detallada** → Dentro de una entrega, qué user-stories se realizan primero. Intentar trasladar los segmentos de desarrollo más arriesgados al principio, intentando respetar las prioridades del negocio

El Juego de la Planificación: Reunión diaria XP

- Reunión diaria "Stand-up Meeting"
  - Todo el equipo
    - Problemas
    - Soluciones
  - De pie en un círculo
    - Evitar discusiones largas
    - Sin conversaciones separadas

Entregas Pequeñas

- Cada entrega es lo más corta posible:
  - Contenga requisitos más valiosos del sistema (básicos)
  - Reducen el riesgo → mayor retroalimentación desde el cliente, y más frecuente
- Minimizar el nº de user-stories que componen una entrega → No realizar user-stories a medias

Metáfora

- Cada proyecto XP es guiado por una metáfora global
- Da un contexto al equipo para entender los elementos básicos y sus relaciones
- Proporciona integridad conceptual

Diseño Simple

- Se diseña “la cosa más simple que pueda funcionar”
- Uso de tarjetas CRC
- Diseño de software correcto, es aquel que:
  - Supera todas las pruebas
  - No tiene lógica duplicada
  - Pone de manifiesto las intenciones importantes de los programadores
  - Tiene el mínimo número de clases y métodos

Pruebas

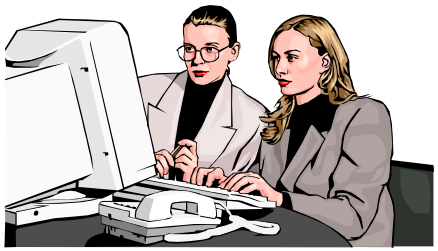
- Las pruebas unitarias se escriben ANTE código
- Pruebas automatizadas
- Permiten el desarrollo de proyectos de forma rápida y segura
- Pruebas unitarias → programadores
- Pruebas funcionales → cliente
- Resultado → Un programa cada vez más seguro

Refactoring

- Refactorización = Mejora del código
- Intentar eliminar complejidad
- Código duplicado → Refactorización
- Se plantea su aplicación después de implementar cada user-story

Programación en Parejas

- Toda el código se escribe en parejas
  - Se produce código de mayor calidad
  - Extiende el conocimiento
- "Se realiza el trabajo de 1 persona en casi la mitad del tiempo y mejor" (cuestionable)



Propiedad Colectiva

- Cualquiera puede modificar el código en cualquier momento → Se evitan cuellos de botella en la codificación
- Todos asume las responsabilidades sobre el conjunto del sistema
- Todos conocen algo sobre todas las partes y conocen muy bien aquéllas en las que trabajan

Integración Contínua

- El código se integra y se prueba después de pocas horas
- Existe una ordenador dedicado para la integración
- Cada pareja integra su código en dicho ordenador

Semana de 40 horas

- Filosofía: “Los programadores que descansan son más productivos”
- El exceso de trabajo es un serio problema en un proyecto
- La gente está más fresca y tiene mejores ideas

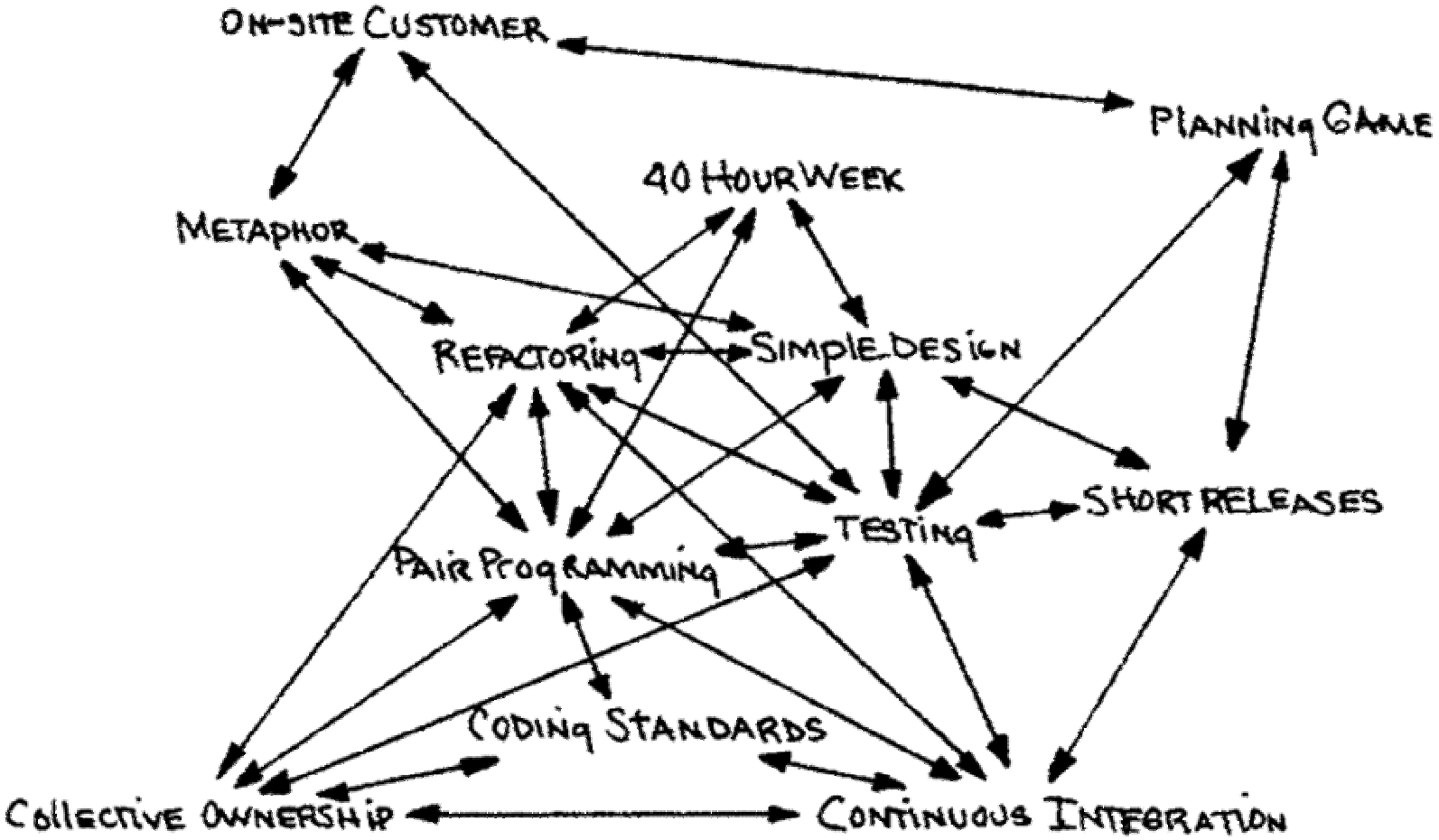
Cliente in situ

- Cliente real = Aquel que usará el sistema cuando esté en producción
- El cliente real debe estar con el equipo de trabajo:
  - Responder preguntas
  - Resolver disputas
  - Establecer prioridades
  - Discutir mejoras

Estándares de Programación

- Son fundamentales cuando los programadores cambian de pareja o hacen *refactoring* del código de otros
- Se consigue un código con el mismo estilo, homogéneo, legible

Interacción entre Prácticas



SCRUM

Gestión Ágil de Proyectos

¿Qué Es SCRUM?

- SCRUM es una metodología ágil de gestión de proyectos
- Busca máximo la productividad de un equipo.
- Reduce al máximo la burocracia y actividades no orientadas a producir software que funcione.
- Produce resultados en periodos muy breves de tiempo (cada 30 días), por medio de iteraciones o **Sprints**.
- Ideal para proyectos con un rápido cambio de requerimientos.

Contexto SCRUM

- Sólo abarca prácticas de gestión sin entrar en las prácticas de desarrollo como puede hacer XP.
- Delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.
- Le da gran protagonismo a las reuniones que realicen a lo largo del proyecto.
- Sus raíces teóricas están en las teorías de la auto-organización.

Actores SCRUM

Propietario del producto

Representa a todos los interesados en el producto final.
Sus áreas de responsabilidad son:

- **Financiación del proyecto.**
- **Retorno de la inversión del proyecto.**
- **Lanzamiento del proyecto.**

Actores SCRUM

Equipo

Responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software.

- **Auto-gestionado.**
- **Auto-organizado.**
- **Multi-funcional.**

Actores SCRUM

Scrum Master

Responsable del proceso Scrum.

- **Formación y entrenamiento del proceso.**
- **Incorporación de Scrum en la cultura de la empresa.**
- **Garantía de cumplimiento de roles y responsabilidad.**

Metodología De Trabajo

- Equipos de entre 6 y 10 personas revisan los requisitos, la tecnología disponible y evalúan los conocimientos para colectivamente determinar como **incrementar la funcionalidad.**
- Reuniones diarias, antes de empezar a trabajar, con una duración máxima de 4 hrs.
- Se llevan a cabo hasta que el proyecto este listo para ser puesto en producción o ser lanzado al mercado.

Metodología De Trabajo

- En la primera reunión se explica al equipo la forma de trabajo.
- Se establecen los criterios para **arreglar los errores por prioridades** (base del éxito del sistema).
- Al inicio de cada iteración se revisa el trabajo pendiente en el proyecto y se selecciona la parte a la cual se le incrementara funcionalidad, para al final de la iteración incorporarla al SW y presentársela a las partes involucradas.
- En cada reunión las preguntas claves a contestar son:
  - **¿Qué es lo que se hizo desde la última reunión?**
  - **¿Qué es lo que se va a hacer hasta la siguiente reunión?**
  - **¿Cómo se va a llevar a cabo?**

Artefactos SCRUM

Sprint

- Es la base del desarrollo Scrum.
- Su duración máxima es de 30 días.
- Se llevan a cabo las tareas pre-establecidas y no se puede modificar el trabajo acordado en el backlog.
- Sólo el ScrumMaster puede abortar un sprint si lo considera no viable por alguna de las sgtes. razones:
  - Las circunstancias del negocio han cambiado.
  - La tecnología acordada no funciona.
  - El equipo ha tenido interferencias.

Artefactos SCRUM

Product Backlog

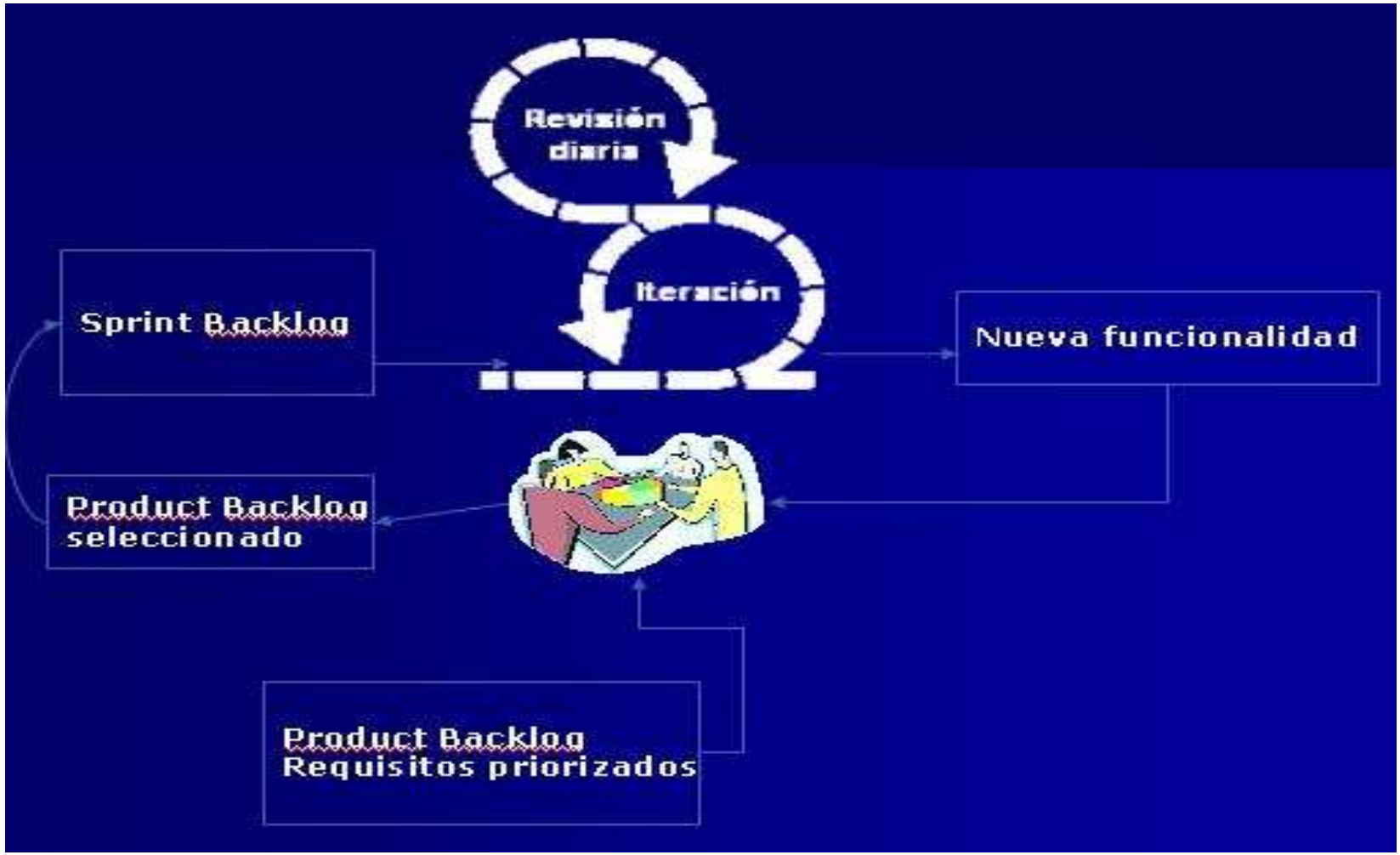
- Crea un listado con los requisitos de los usuarios o propietarios del sistema para planificar el proyecto.
- No es una lista completa y definitiva. Es sólo una estimación inicial de los requisitos.
- Es un documento dinámico que incorpora las constantes necesidades del sistema y se mantiene durante todo el ciclo de vida (hasta la retirada del Sist.).

Artefactos SCRUM

Sprint Backlog

- Especifica la serie de tareas que se van a desarrollar según los requisitos señalados.
- Estas tareas tienen una duración de entre 4 y 16 hrs. de trabajo.
- Las de mayor duración intentar descomponerlas en Sub-Tareas dentro de ese rango de tiempo.
- Al final del sprint se busca un incremento en la funcionalidad.

El Flujo De SCRUM



Conclusiones

¿Qué resultado proveen las AMs?

- Hay pocos datos concretos del índice de éxito de proyectos
- Está teniendo un gran auge
  - Aumento en el número de proyectos
  - ¿Por qué?
    - Tiene el apoyo de muchos gurús en ingeniería de sw
    - Es un proceso para gente que odia los procesos
    - Tiene sentido
    - ¿Política? ... Pugna entre comunidades

¿Cuándo utilizar una Metodología Ágil?

- ¿Existe ya un proceso? Si
- ¿Reacciona bien a los cambios? Si
- ¿Está el equipo contento con él? Si

⇒ Mejor esperar

- Se están recogiendo datos (red NAME) http://name.case.unibz.it/
- En un futuro se podrán hacer comparaciones sobre lo que es más conveniente

¿Cuándo utilizar una Metodología Ágil?

- ¿Existe ya un proceso? No
  - o existe pero no reacciona bien a los cambios
  - o existe pero el equipo no está contento con él

⇒ Una Metodología Ágil puede ser una buena forma de empezar

- Fácil de financiar
- A los programadores les gusta
- A los clientes les gusta el control añadido