Metodologías de Desarrollo de Software

Medina Lopez, Jahir Gilberth Saavedra Jimenez, Lenin Sleyter

Definición

Las metodologías nos indican un plan adecuado de gestión y control del proyecto de software : definición de etapas, ingresos y salidas, restricciones, comunicaciones, tareas ordenadas y distribución de recursos.

De manera más formal:

> [1] *Un modelo de procesos es una representación del mundo real, que captura el estado de actual de las actividades para guiar, reforzar o automatizar partes de la producción de los procesos.*

[1] Derniame, 1999

Metodologías Clásicas

Secuencial

Representado por metodologías tan famosas como Waterfall. Se inicia con un completo análisis de los requisitos de los usuarios. En el siguiente paso, los programadores implementan el diseño y finalmente, el completado y perfecto sistema es probado y enviado.

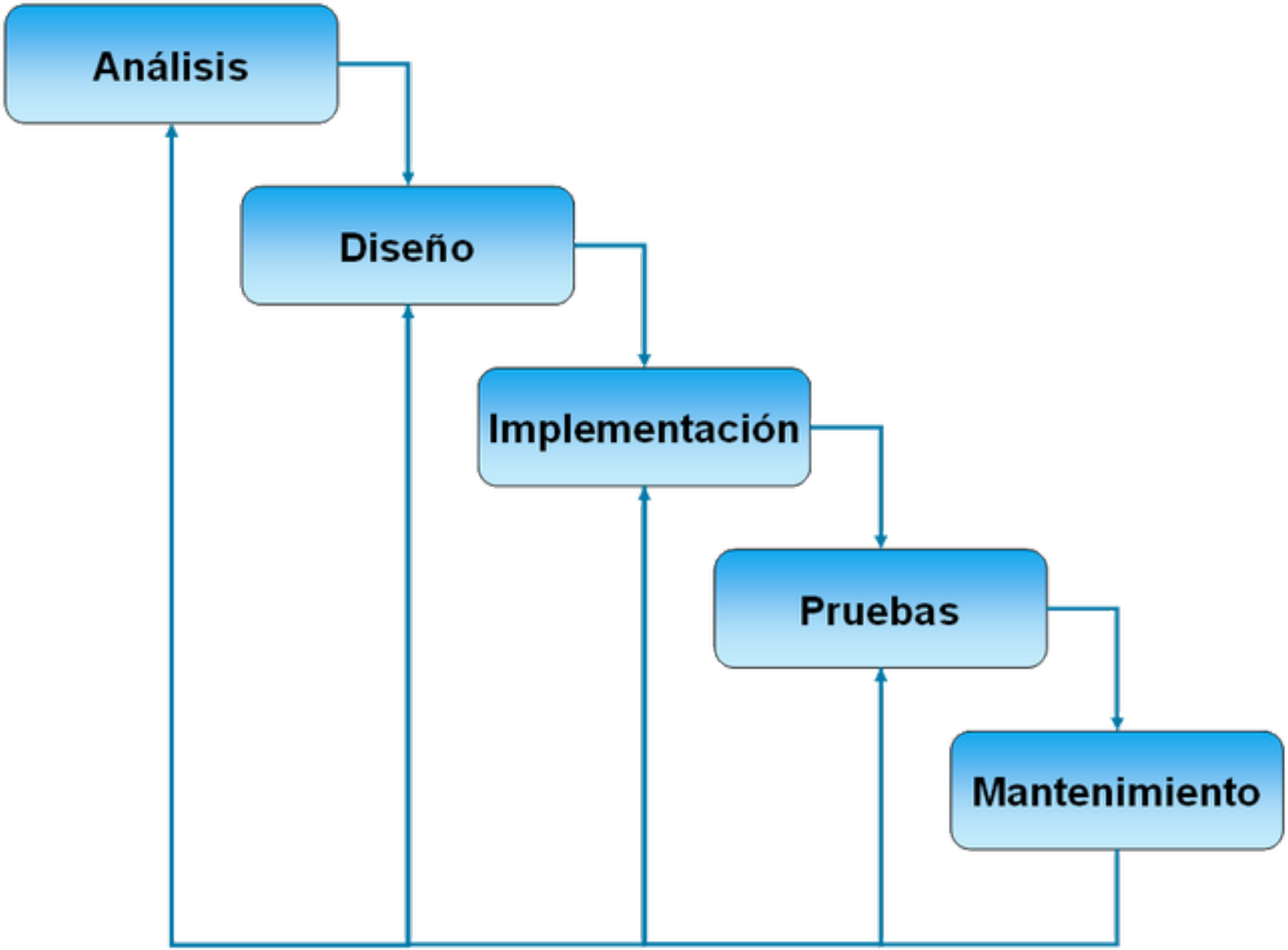


Figure 1: Flujograma de la Metodología

Incremental

Su principal objetivo es reducir el tiempo de desarrollo, dividiendo el proyecto en intervalos incrementales superpuestos. Del mismo modo que con el modelo waterfall, todos los requisitos se analizan antes de empezar a desarrollar, sin embargo, los requisitos se dividen en "incrementos" independientemente funcionales.

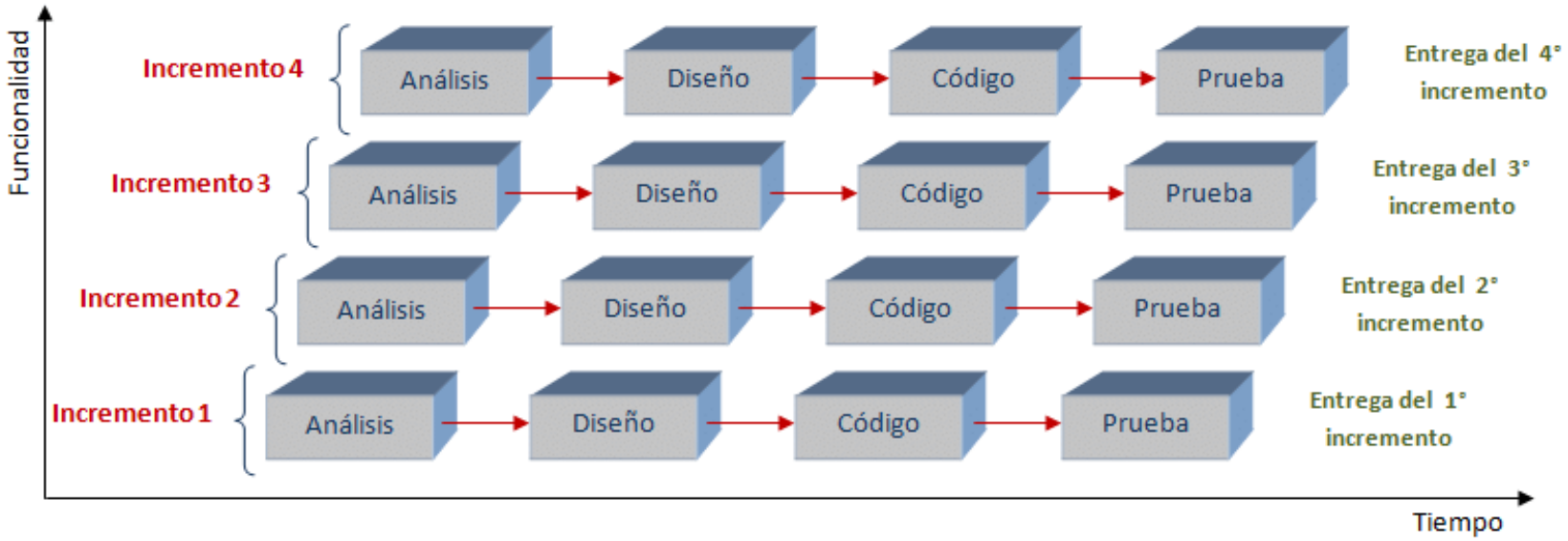


Figure 2: Flujograma de la Metodología

Iterativo

A diferencia del modelo incremental se centra más en capturar mejor los requisitos cambiantes y la gestión de los riesgos. En el desarrollo iterativo se rompe el proyecto en iteraciones de diferente longitud, cada una de ellas produciendo un producto completo y entregable.

Figure 3: Flujograma de la Metodología

Metodologías Orientadas a Objetos

ICONIX

En este contexto el proceso ICONIX[a] se define como un "proceso" de desarrollo de software práctico. ICONIX está entre la complejidad del RUP[b] y la simplicidad y pragmatismo del XP[c], sin eliminar las tareas de análisis y de diseño que XP no contempla. ICONIX es un proceso simplificado en comparación con otros procesos más tradicionales, que unifican un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

---

[a] Creado por Rosenberg & Scott, 1993
[b] Rational Unified Processes
[c] Extreme Programming

Características

Las tres características fundamentales de ICONIX son:

- Iterativo e incremental: Varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.
- Trazabilidad: Cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos producidos.
- Dinámica del UML: La metodología ofrece un uso "dinámico del UML" como los diagramas del caso de uso, diagramas de secuencia y de colaboración.

Etapas

a

1. Definición de Requerimientos
   - Modelado del Dominio
   - Modelado de los casos de uso
   - Revisión de los Requerimientos
2. Análisis, Diseño Conceptual y Arquitectura Técnica
   - Análisis de Robustez
   - Revisión del diseño preliminar
   - Arquitectura Técnica
3. Diseño y Programación
   - Diagrama Secuencia
   - Revisión del Diseño Crítico
   - Implementación
   - Revisión de Código y Actualización del modelo
4. Pruebas y Seguimiento de Requerimientos
   - Pruebas Basadas en Diseño
   - Cumplir (Validar el Cumplimiento) de Requerimientos

---

[a] Doug Rosenberg and Matt Stephens Use Case Driven Object Modeling with UML Theory and Practice

RUP

El *Proceso Unificado de Rational*[2] es un Proceso de Desarrollo de software iterativo, siendo un framework y metodología al mismo tiempo. Creado, Desarrollado y Mantenido por IBM desde 2003 , RUP no es un sistema rígido de pasos sin embargo un marco de trabajo adaptable (es por esto que puede ser visto como metodología). RUP, al igual que ICONIX emplea una variante propia de UML.

[2] Rational Unified Process

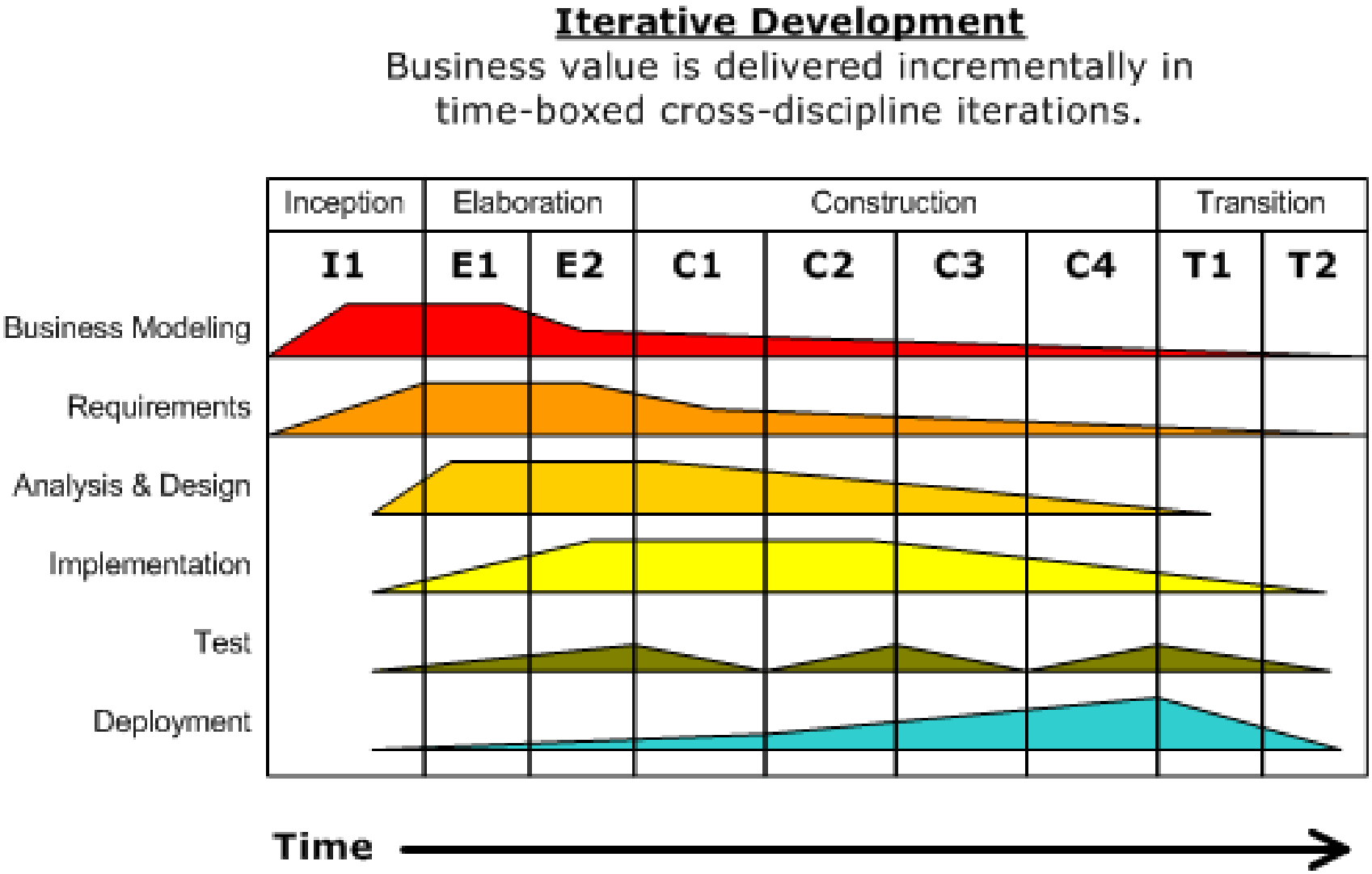


Figure 4: Vista General

Componentes Importantes

- Roles. Definir quién hace qué.
- Productos a Trabajar/Obtener. Definir lo que genera cada rol.
- Tareas. Definir la forma en la que cada rol genera su producto.

Etapas

- Creación o Planeamiento (Inception).
  - *Stakeholder Concurrence* [a]
  - Entendimiento de los Requerimientos
  - Veracidad de las estimaciones
    - Costo/tiempo
    - Prioridades
    - Riesgos
    - Procesos de Desarrollo
  - Alcance y Detalle de la Arquitectura Prototipo
  - Definir el estado actual del sistema.

[a] Se define como la identificación de los puntos en común de las áreas más importantes.

Etapas

- Elaboración
  - Modelo de Casos de Uso
  - Descripción de la Arquitectura
  - Planteamiento del plan de Desarrollo general
  - Prototipos Ejecutables o Pruebas de Concepto
  - Validación de Suficiencia
- Construcción
  - Si el proyecto es pequeño puede bastar con una sola iteración, de lo contrario el número de iteraciones se deberá haber previsto en la etapa anterior.

Etapas

- Transición.
  - Despliegue en producción.
  - Pruebas de Validación
  - Comparación con el sistema previo
  - Validación del cumplimiento de expectativas.
  - Revisión de la calidad del Producto

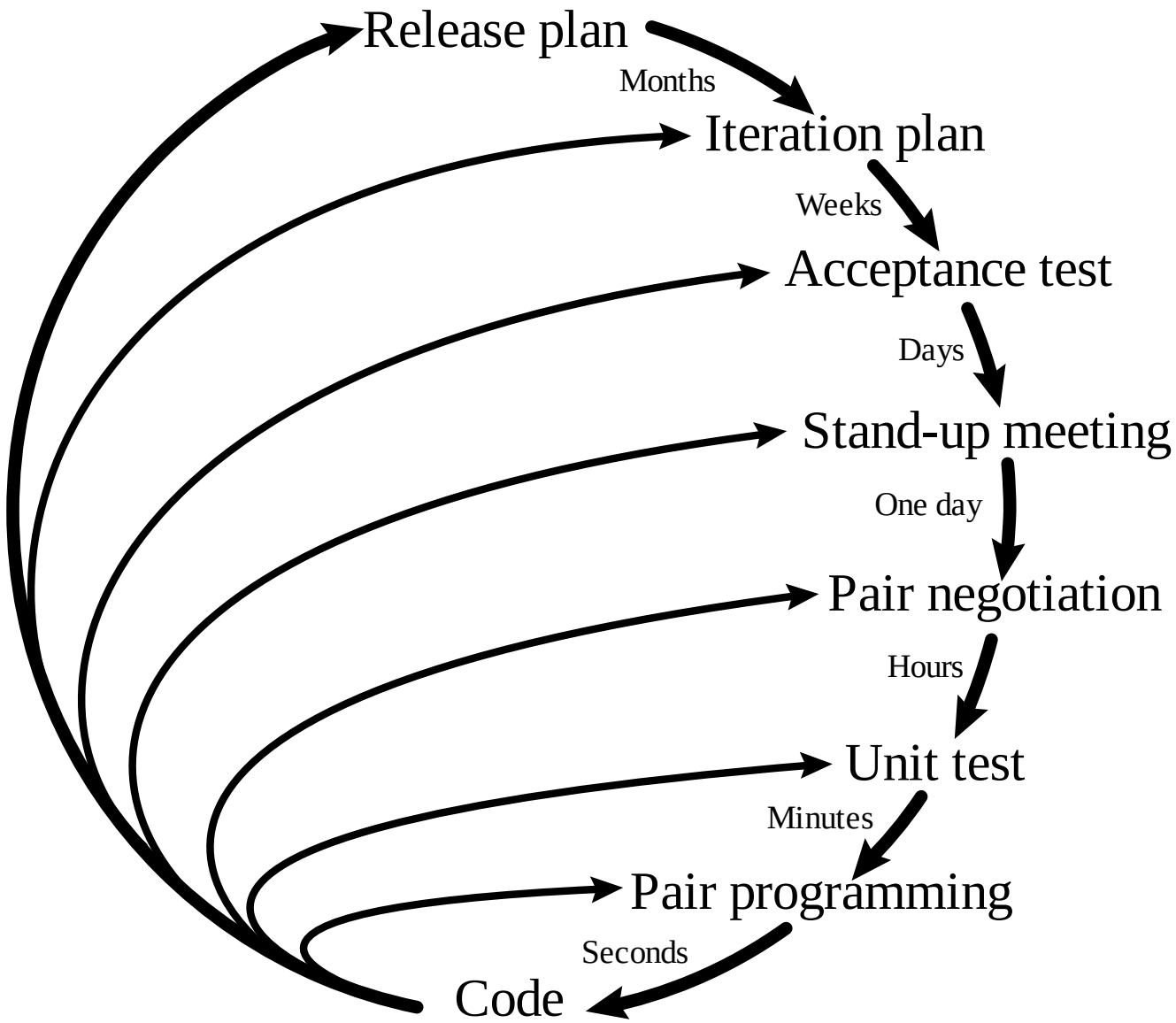
XP (Extreme Programming)

Esta metodología busca mejorar la calidad y responsividad al cambio de requerimientos. Es un tipo de metodología ágil. Desarrollado por Kent Beck en marzo de 1996.

Influencia

- Internamente, la Programación Orientada a Objetos.
- Externamente, El internet y los negocios online

Planning/feedback loops



Características

En el caso de XP , es una metodología orientada a priorizar las personas implicadas en el desarrollo y no el producto en sí. Teniendo así , que en el libro original menciona como prácticas primarias:

- Sentarse Juntos, Reunir a todo el equipo y hablar con cada uno de ser necesario
- Convertirse en un Equipo Unido
- Espacios de Trabajo con Información a mano.
- Trabajo Motivado y Eficiente
- Programación en pares.
- Emparejar Correctamente y Respetar el Espacio Personal

Características

- Historias como forma de control de avance
- Ciclos Semanales
- Trabajo Modular
- Descansos
- Integración/Compilación cada 10 minutos
- Integración Continua
- Programación que prioriza el `testeo`
- Diseño Incremental

Flujograma del Trabajo

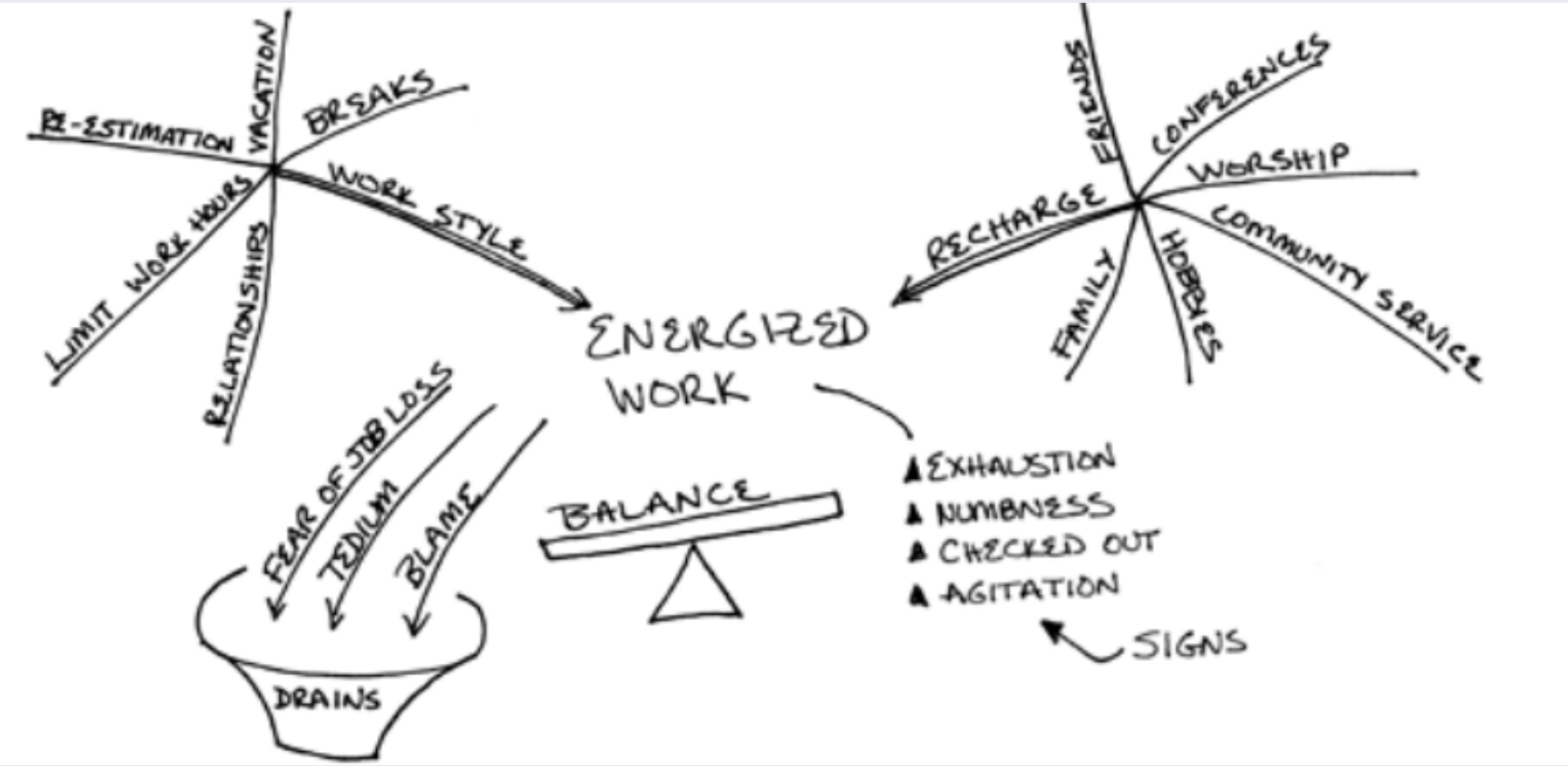


Figure 6: Flujograma Integrador

Flujograma del Trabajo

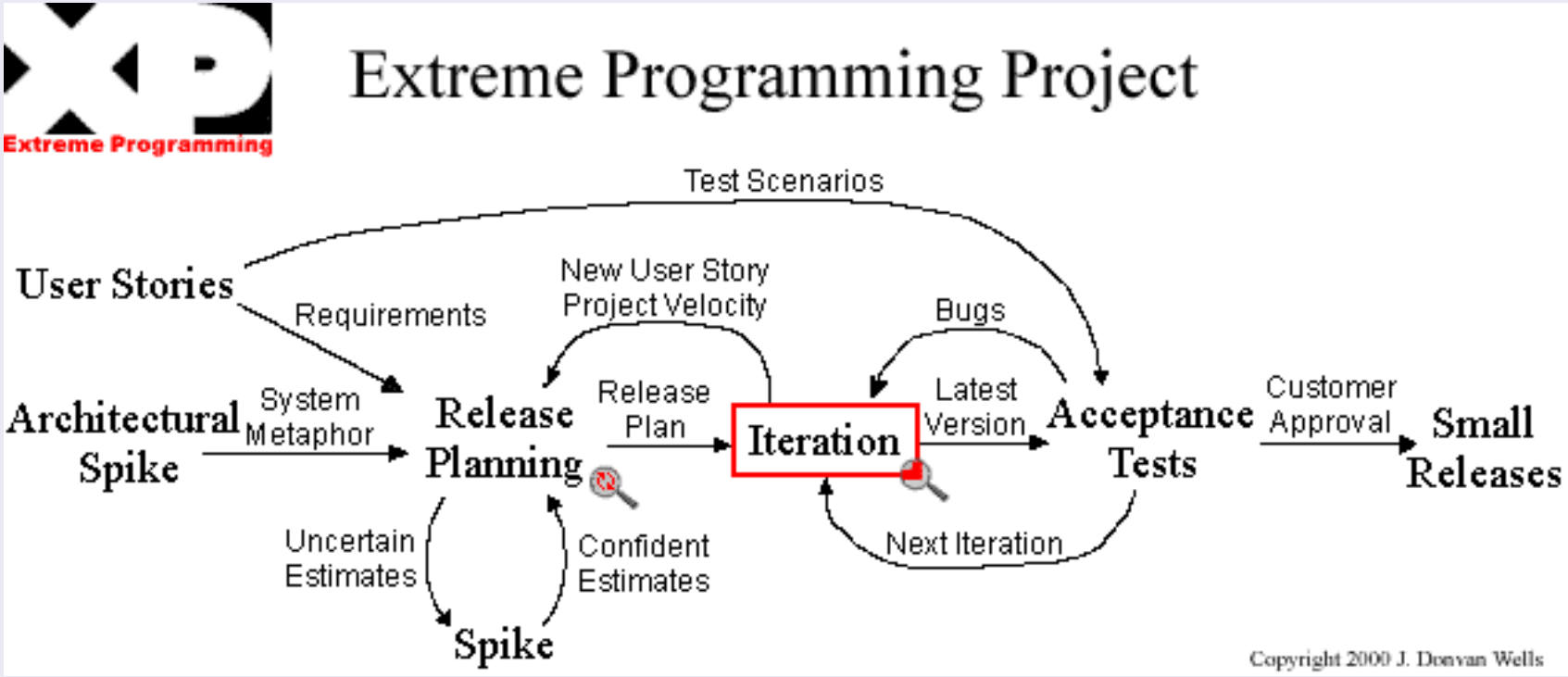


Figure 7: Estructura de un Proyecto

Referencias

- Derniame, J (1999) Software Process: Principles, Methodology, Technology. Lecture Notes in Computer Science 1500 Springer 1999, ISBN 3-540-65516-6 BibTeX A Discipline of Programming.
- Espinoza, A (2013). Manual para elegir una metodología de desarrollo de software dentro de un proyecto informático. Facultad de Ingeniería, Universidad de Piura. Piura Perú.
- Gacitúa, R (2003). MÉTODOS DE DESARROLLO DE SOFTWARE: EL DESAFÍO PENDIENTE DE LA ESTANDARIZACIÓN Depto. Sistemas de Información, Facultad de Ciencias Empresariales, Universidad del Bío-Bío, Avda. Collao 1202,Concepción, Chile.

- Amavizca, L & et.al (2014). Aplicación de la metodología semi-ágil ICONIX para el desarrollo de software: implementación y publicación de un sitio WEB para una empresa SPIN - OFF en el Sur de Sonora, México. Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI 2014)
- Rosenberg, D & Stephens, M. (2007) Use Case Driven Object Modeling with UML: Theory and Practice. Apress. (ISBN 1590597745)

- Mark Aked (2003). Risk reduction with the RUP phase plan developerWorks, IBM.
- Kruchten, P (2000). The Rational Unified Process An Introduction, Second Edition Addison Wesley Second Edition March 14, 2000 ISBN: 0-201-70710-1, 320 pages

- Beck, K. (1999). Extreme Programming Explained: Embrace Change. Addison-Wesley. ISBN 978-0-321-27865-4.