Gestión de Proyectos de Software

Ing. Geovanny Raura, PHD

Octubre 2025

1. Introducción a la Calidad y Desarrollo de Software

- Construir software es más difícil de lo que parece:
  - http://www.nbcnews.com/news/investigations/spy-plane-fries-air-traffic-control-computers-shuts-down-lax-n95886
  - http://www.nytimes.com/2014/05/01/business/faa-computer-problem-grounds-planes-in-los-angeles.html?_r=0
  - https://www.elcomercio.com/actualidad/falla-de-microsoft-operaciones-aereas-ecuador.html

1. Introducción a la Calidad y Desarrollo de Software

EL COMERCIO

ÚLTIMA HORA ACTUALIDAD TENDENCIAS TECNOLOGÍA DEPORTES OPINION VIDEOS PODCASTS

ECUADOR

Falla de Microsoft causa problemas en operaciones aéreas, ¿qué pasa en Ecuador?

Las operaciones aéreas en Ecuador se encuentran bajo vigilancia tras la falla mundial de Microsoft

https://www.elcomercio.com/actualidad/ecuador/falla-de-microsoft-operaciones-aereas-ecuador.html

1. Introducción a la Calidad y Desarrollo de Software

1. Introducción a la Calidad y Desarrollo de Software

BUSINESS DAY

F.A.A. Computer Problem Delays Flights in the West

By MATTHEW L. WALD APRIL 30, 2014

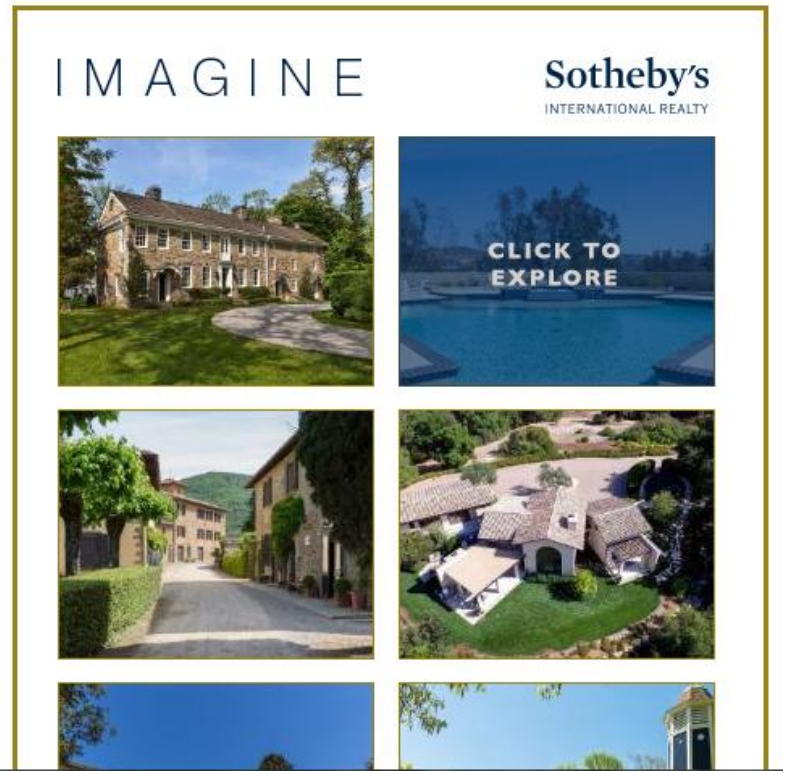
A computer failure at an air traffic control center forced the Federal Aviation Administration to hold planes on the ground at Los Angeles International Airport and other airports in the region on Wednesday afternoon.

The problem at the Los Angeles Air Route Traffic Control Center, which handles higher-altitude aircraft, meant planes bound for the region were also grounded.

Around 4:30 p.m., the Los Angeles International Airport reported that flights had resumed. But delays were expected to last several hours.

Government officials were not immediately sure why the system had failed. Areas affected by the shutdown included Southern California, western Arizona, southern Nevada and part of Utah.

The computer that failed is part of the En Route Automation Modernization system, known as ERAM. The F.A.A. has been rolling it out since 2008 in an effort to integrate data from more sources, increasing the number of planes that can be



:(

Se ha producido un problema en el equipo y es necesario reiniciarlo.

Solo estamos recopilando información de errores y, a continuación, te

you.

20% reiniciaremos. íntegro

Para obtener más información sobre este problema y las posibles correcciones, visite https•//www.windows.com/stopcode

Si •llama a una persona de soporte técnico, proporcióneles esta información:

Código de detención: CRÍTICO PROCESO_MURIÓ

1. Introducción a la Calidad y Desarrollo de Software

Construir software es más difícil de lo que parece:

El **16,3%** de los proyectos software tienen éxito

El proyecto es completado en tiempo y en presupuesto, con todas las características y funcionalidades especificadas al comienzo del proyecto

El **52,7%** de los proyectos software cuestan más, tardan más o hacen menos

El proyecto es completado y operacional, pero con mayor presupuesto que el presupuestado (189% mas), tardando más del tiempo estimado y ofreciendo menos características y funcionalidades de las especificadas inicialmente (42%)

El **31%** son cancelados

El proyecto es cancelado en cierto momento del desarrollo antes de poner el sistema en operación

1. Introducción a la Calidad y Desarrollo de Software

De hecho… ¿Quién no ha sufrido algún fallo misterioso en algún software?

¿El software falla más a menudo que otros productos ingenieriles?

¿Por qué?

¿Son los errores irremediables?
Complejidad del software

- La extrema dificultad de los sistemas software favorece a que existan errores remanentes en los sistemas finalizados
  - Un programa de unos centenares de líneas de código puede contener decenas de decisiones, lo que implica miles de rutas de ejecución alternativas. Es materialmente imposible el ensayo de todas las alternativas de ejecución posibles
  - Las alternativas posibles de la realidad a la que el software responde son cuasi-infinitas

¿Son los errores irremediables?

Falta de conocimiento

- Nuestra capacidad para garantizar la fiabilidad del software es muy inferior a lo necesario. **No podemos demostrar la corrección** de los programas al estilo de los otros ingenieros

  Los otros ingenieros usan análisis matemáticos para predecir el comportamiento de sus sistemas. Esa predicción permite descubrir defectos antes de que el producto esté operativo

- **Las matemáticas tradicionales**, aptas para la descripción de sistemas físicos, **no son aplicables** al universo sintético binario del software

  Es la matemática discreta, una especialidad mucho menos madura y casi no estudiada hasta la aparición de las computadoras, la que gobierna el campo de los sistemas software

¿Son los errores irremediables?

Estado Actual

- Dada la imposibilidad de aplicar métodos matemáticos rigurosos, el modo que tenemos para respaldar la confianza de los programas es la **ejercitación**

  Hacer funcionar los programas, observando directamente su comportamiento y depurándolos cada vez que aparece una deficiencia. La fiabilidad de los programas crecerá a lo largo de este proceso

- La calidad que se puede obtener mediante este **procedimiento artesanal es bastante baja**. De ahí que, a pesar de haber sido ensayados rigurosa y sistemáticamente, la mayoría de los sistemas software **contengan todavía defectos** cuando son entregados

Resumiendo

- Es imposible garantizar un producto desarrollado 100% libre de defectos debido a la inmadurez de la IS
  - Falta de conocimientos científicos que predigan los resultados de las técnicas
  - Falta de normas sobre quien o cómo se puede desarrollar software
  - …

Pero…

¡¡Esta situación no debe ser una licencia para el "todo vale"!!

1. Introducción a la Calidad y Desarrollo de Software

Profesionalidad = Calidad

El objetivo de un
Ingeniero Software
debe ser
entregar un producto con el nivel de
Calidad
que las técnicas de hoy permitan

¿Qué es la Calidad del Software?

- ¿Qué entienden ustedes por calidad de un software?

- Un concepto demasiado abstracto que necesita descomponerse en atributos más palpables

¿Qué es la Calidad del Software?

- “La calidad es la suma de todos aquellos aspectos o características de un producto o servicio que influyen en su capacidad para satisfacer las necesidades, expresadas o implícitas” (ISO 8402)
- “Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas” (IEEE 729-83)
- “Capacidad del producto software para satisfacer los requisitos establecidos”(DoD 2168)

GrimpSoft
Grupo de Investigación en Modelos de Producción de Software

Criterios de Calidad del Software

- Fiabilidad
- Funcionalidad
- Eficiencia
- Usabilidad
- Mantenibilidad
- Portabilidad
- Seguridad

1. Introducción a la Calidad y Desarrollo de Software

Descomposición de la Calidad de Software por la ISO 9126-1998

CHARACTERISTICS AND SUBCHARACTERISTICS	DESCRIPTION
Functionality	**Characteristics relating to achievement of the basic purpose for which the software is being engineered**
Suitability	The presence and appropriateness of a set of functions for specified tasks
Accuracy	The provision of right or agreed results or effects
Interoperability	Software's ability to interact with specified systems
Security	Ability to prevent unauthorized access, whether accidental or deliberate, to programs and data
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols.
Reliability	**Characteristics relating to capability of software to maintain its level of performance under stated conditions for a stated period of time**
Maturity	Attributes of software that bear on the frequency of failure by faults in software
Fault tolerance	Ability to maintain a specified level of performance in cases of software faults or unexpected inputs
Recoverabiltiy	Capability and effort needed to re-establish level of performance and recover affected data after possible failure
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols
Usability	**Characteristics relating to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users**
Understandability	The effort required for a user to recognize the logical concept and its applicability
Learnability	The effort required for a user to learn its application, operation, input and output
Operability	The ease of operation and control by users
Attractiveness	The capability of the software to be attractive to the user
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols
Efficiency	**Characteristics related to the relationship between the level of performance of the software and the amount of resources used, under stated conditions**
Time behavior	The speed of response and processing times and throughput rates in performing its function
Resource utilization	The amount of resources used and the duration of such use in performing its function
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols
Maintainability	**Characteristics related to the effort needed to make modifications, including corrections, improvements or adaptation of software to changes in environment, requirements and functional specifications**
Analyzability	The effort needed for diagnosis of deficiencies or causes of failures, or for identification parts to be modified
Changeability	The effort needed for modification fault removal or for environmental change
Stability	The risk of unexpected effect of modifications
Testability	The effort needed for validating the modified software
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols
Portability	**Characteristics related to the ability to transfer the software from one organization or hardware or software environment to naother**
Adaptability	The opportunity for its adaptation to different specified environments
Installability	The effort needed to install the software in a specified environment
Co-existence	The capability of a software product to co-exist with other independent software in common environment
Replaceability	The opportunity and effort of using it in the place of other software in a particular environment
Compliance	Adherence to application-related standards, conventions, regulations in laws and protocols

1. Introducción a la Calidad y Desarrollo de Software

Criterios de Calidad del Software

CALIDAD FUNCIONAL

- **Fiabilidad** Se mantiene operativo
- **Funcionalidad** Realiza el trabajo deseado
- **Eficiencia** Responde con velocidad apropiada
- **Usabilidad** El usuario está cómodo con él

CALIDAD NO FUNCIONAL

- **Mantenibilidad** Modificable con adecuado costo
- **Portabilidad** Opera en diferentes entornos

Funcionalidad y Fiabilidad

- **Fiabilidad**

  El sistema software se mantiene operativo

- **Funcionalidad**

  El sistema software realiza el trabajo deseado por el usuario

¿Cómo se comprueban?

Evaluando la Fiabilidad

- **Si la Fiabilidad es**
  El sistema software se mantiene operativo

- **La Evaluación se realizará**
  Buscando defectos que provoquen la mala operación del sistema (en cualquier contexto)

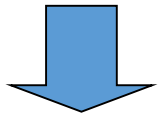
Evaluando la Funcionalidad

- **Si la Funcionalidad es**

  El software realiza la tarea deseado por el usuario

- **La Evaluación se realizará**

  Comparando la tarea que realiza el sistema con la esperada por el usuario (Especificación de Requisitos)



  - **Realiza las tareas encomendadas**
  - **Tiene los efectos o las respuestas correctas o acordadas**

1. Introducción a la Calidad y Desarrollo de Software

¿Entonces?

- Deben evaluarse los productos del desarrollo según se generan

  ¡En lugar de esperar a tener código!

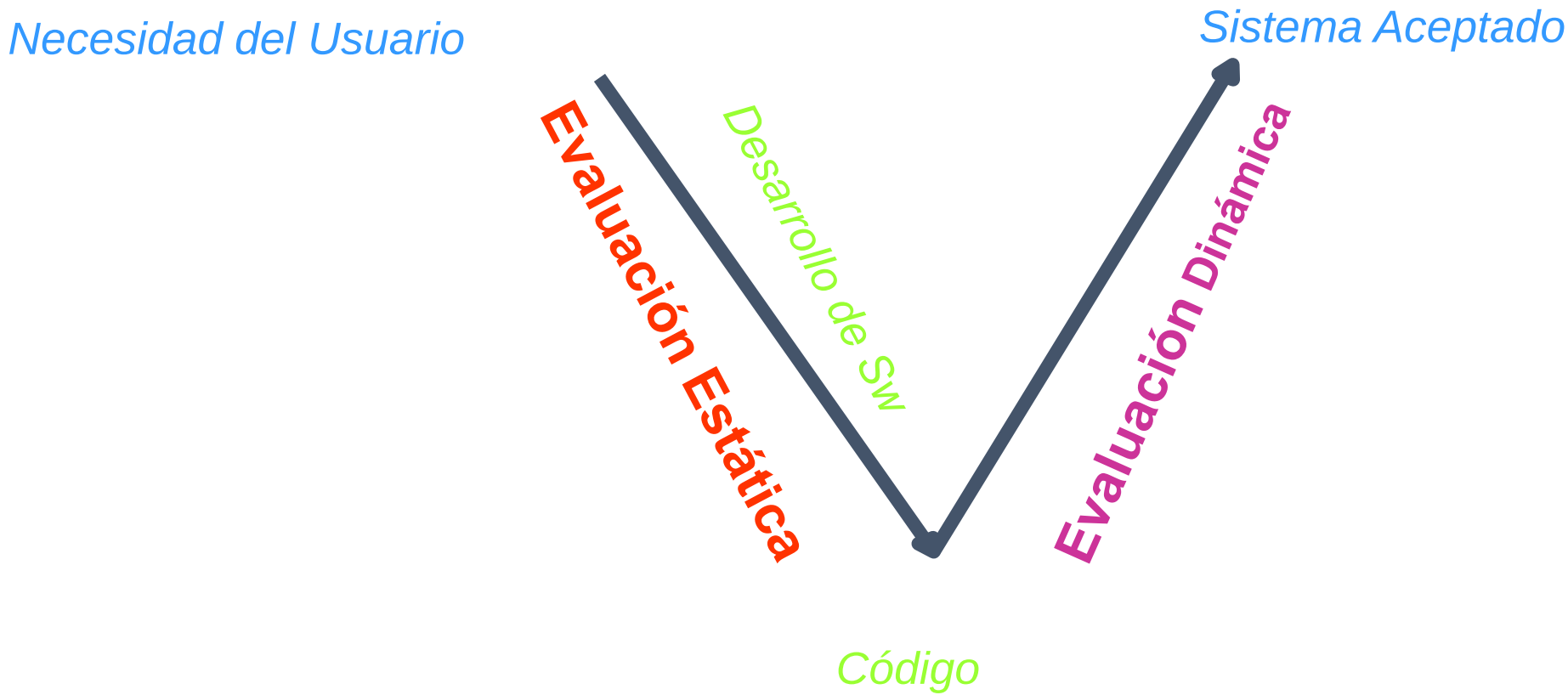
- Debe ejercitarse el código adecuadamente

TÉCNICAS ESTÁTICAS

TÉCNICAS DINÁMICAS

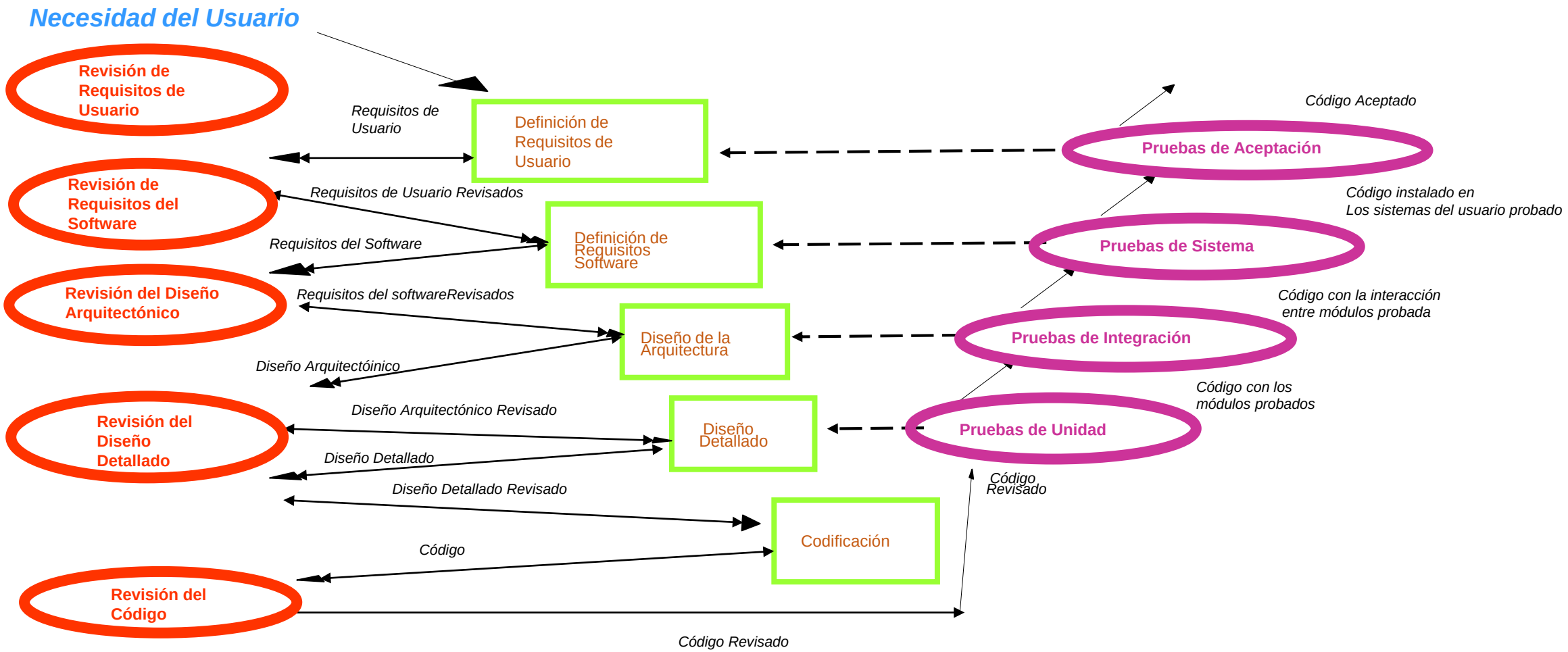
1. Introducción a la Calidad y Desarrollo de Software

Evaluación DURANTE la construcción



1. Introducción a la Calidad y Desarrollo de Software

Actividades de desarrollo y actividades de evaluación



Evaluación y Defectos

- La evaluación busca defectos en los productos (Req. Dsñ. Cdg) del desarrollo
  - Provocan mala operación
  - No se reflejan las tareas deseadas
  - Se producen respuestas/efectos incorrectos
- No confundir defecto con fallo

Defecto: Error/Falta/Fallo

- **Error**

  Acción humana que produce una falta

- **Falta**

  Algo que está mal en un producto

- **Fallo**

  Manifestación de una falta

Humano

Req, Disñ, Codg, …

Sistema

TÉCNICAS ESTÁTICAS

TÉCNICAS DINÁMICAS

DEFECTO

Resumiendo lo aprendido hasta aquí

- Es imposible garantizar un software 100% libre de defectos debido a la inmadurez de la IS
- Pero existen técnicas para reducir al mínimo los defectos remanentes
  - Técnicas Estáticas (o Revisiones)
    - Buscan faltas
    - Aplicables a cualquier producto
  - Técnicas Dinámicas (o T. de Pruebas)
    - Generan casos de prueba
    - Buscan fallos
    - Aplicables sólo a código
  - Ambas se centran en defectos de fiabilidad y funcionalidad

¿Por qué fracasan los proyectos de software?

- Retrasos y desviaciones en la planificación.
- Costo de mantenimiento elevados.
- Alta tasa de defectos.
- Requisitos mal comprendidos.
- Cambios de negocio.
- Falsa riqueza de características.
- Cambios de personal.

¿Qué busca la Ingeniería de Software?

- Es necesario construir en un **tiempo corto**, **sin un costo excesivo**, aplicaciones de software complejas, de **calidad** y que soporten las **necesidades del usuario.** Estas aplicaciones deberían ser **fáciles y rápidas de modificar**.

1. Naturaleza de los proyectos Software

- La naturaleza de un proyecto de software se refiere a las características y aspectos esenciales que definen la planificación, desarrollo, implementación y mantenimiento del software. Contiene los siguientes elementos clave:
- **Objetivos y Alcance**.
- **Requerimientos.**
- **Ciclo de Vida del Desarrollo de Software (SDLC)**
- **Equipo de Trabajo**
- **Metodologías de Desarrollo**
- **Herramientas y Tecnologías**
- **Gestión de Proyecto**
- **Riesgos y Desafíos**
- **Entrega y Valor al Cliente**

1. Naturaleza de los proyectos Software

- **Objetivos y Alcance:**
  - **Objetivos**: Los proyectos de software tienen objetivos específicos que buscan cumplir, como resolver un problema particular, mejorar un proceso existente o crear una nueva funcionalidad.
  - **Alcance**: Define los límites del proyecto, incluyendo qué se hará y qué no se hará, estableciendo claramente las funcionalidades y características que debe tener el software.
- **Requerimientos:**
  - **Funcionales**: Son las especificaciones sobre lo que el software debe hacer, incluyendo características y funciones.
  - **No funcionales**: Incluyen requisitos de rendimiento, seguridad, usabilidad, escalabilidad, etc.

2. Características de los proyectos de software

Los proyectos de software tienen varias características, entre ellas:

Ciclo de vida

El software debe cumplir con todas las etapas del ciclo de vida del desarrollo de software, como la planificación, el diseño, la integración, la prueba, la implementación y el mantenimiento.

Gestión

- La gestión de proyectos de software se encarga de planificar el desarrollo del producto, hacer un seguimiento del trabajo y garantizar que se cumplan los estándares y el presupuesto. Dentro de la gestión se contempla: Equipo de trabajo, herramientas y tecnologías, riesgos y desafíos

2. Características de los proyectos de software

Equipo de Trabajo:

Incluye desarrolladores, analistas de sistemas, diseñadores, testers, gerentes de proyecto, y otros roles especializados que colaboran para llevar a cabo el proyecto.

- **Herramientas y Tecnologías:**
  - Uso de lenguajes de programación, frameworks, sistemas de gestión de bases de datos, herramientas de integración continua, sistemas de control de versiones, etc.

- **Riesgos y Desafíos:**
  - Identificación y mitigación de riesgos, como cambios en los requisitos, problemas técnicos, limitaciones de recursos, etc.

2. Características de los proyectos de software

Definición

En la definición del proyecto se debe incluir la idea general y los objetivos del desarrollo.

Proceso de desarrollo

De forma general el proceso de desarrollo de software incluye el análisis de requisitos, el diseño, el desarrollo, las pruebas, la implementación y el mantenimiento. El proceso de desarrollo contempla el uso de **Metodologías de Desarrollo:**

- **Tradicionales**: Como el modelo en cascada, donde cada fase del SDLC se realiza de forma secuencial.
- **Ágiles**: Como Scrum y Kanban, que se enfocan en iteraciones rápidas, colaboración constante con el cliente y flexibilidad para cambios.

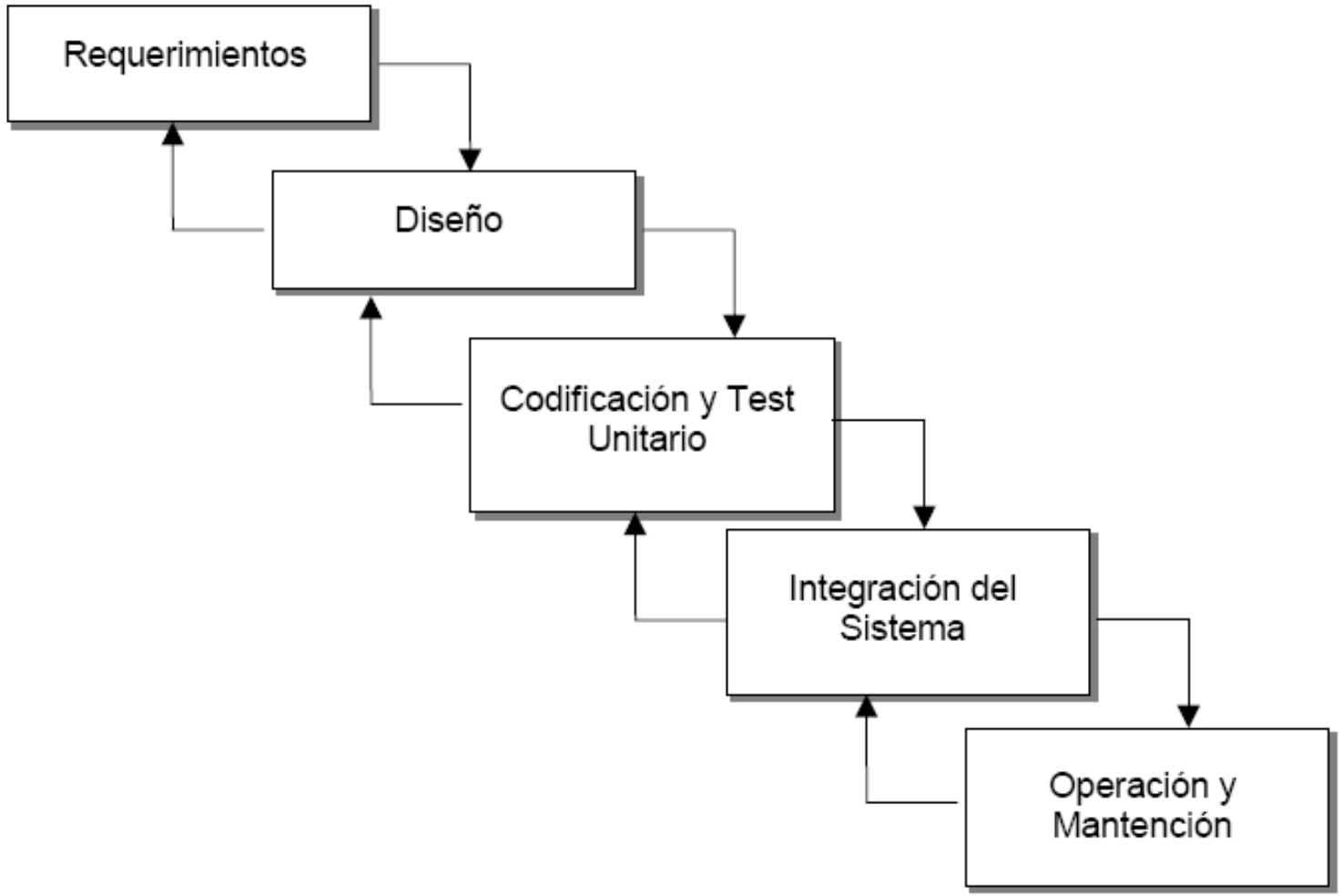
3. Ciclo de vida de los proyectos de software

- **Ciclo de Vida del Desarrollo de Software:**

  - **Planificación**: Definición del proyecto, estimación de recursos, cronograma y presupuesto.
  - **Análisis de Requerimientos**: Recopilación y documentación de los requisitos del cliente.
  - **Diseño**: Creación de la arquitectura del software, diseño de interfaces y base de datos.
  - **Desarrollo**: Codificación y construcción del software.
  - **Pruebas**: Verificación y validación del software para asegurar su calidad y funcionamiento correcto.
  - **Implementación**: Despliegue del software en el entorno de producción.
  - **Mantenimiento**: Actualización y corrección de errores del software después de su lanzamiento.

3. Ciclo de vida de los proyectos de software

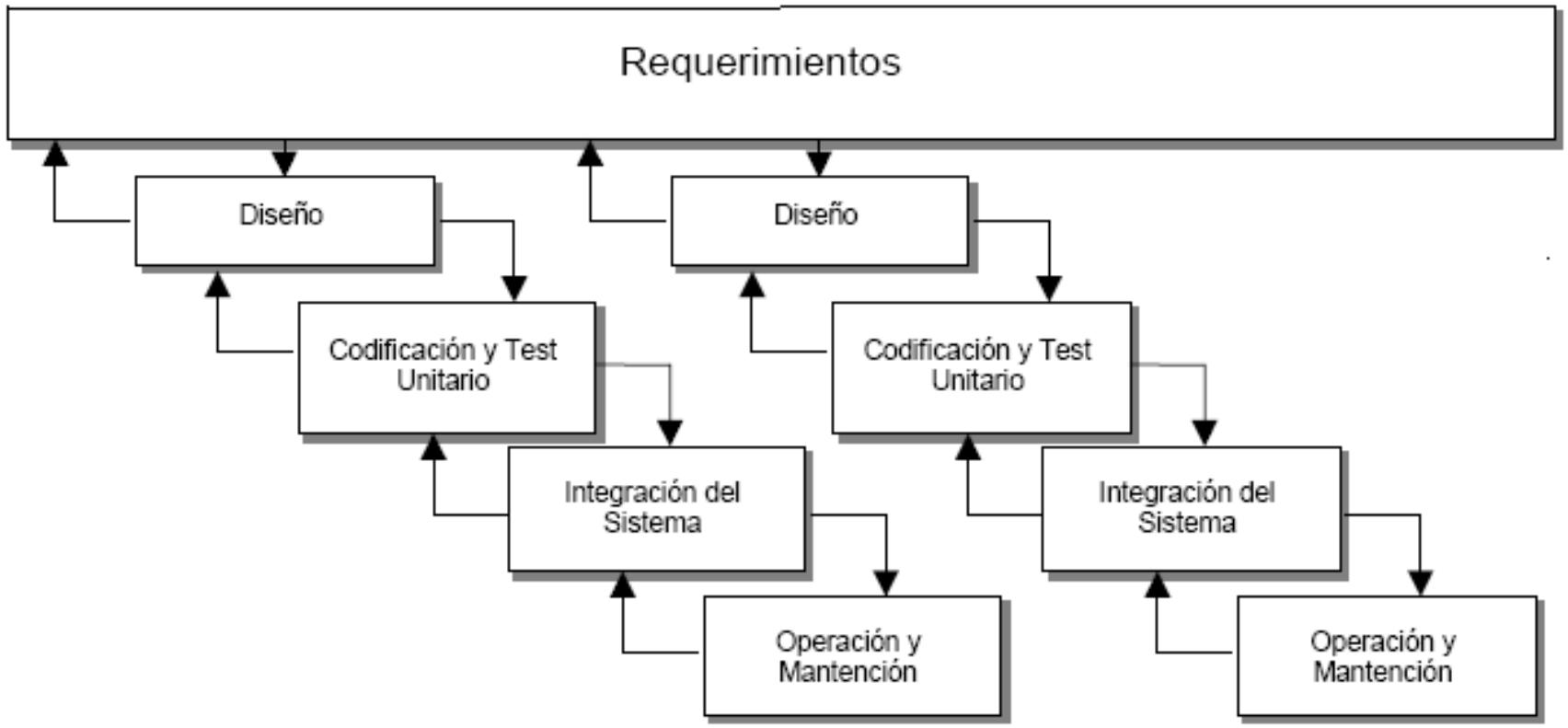
- **Ciclo de Vida del Desarrollo de Software CASCADA:**



Modelo de Ciclo de Vida Cascada.

3. Ciclo de vida de los proyectos de software

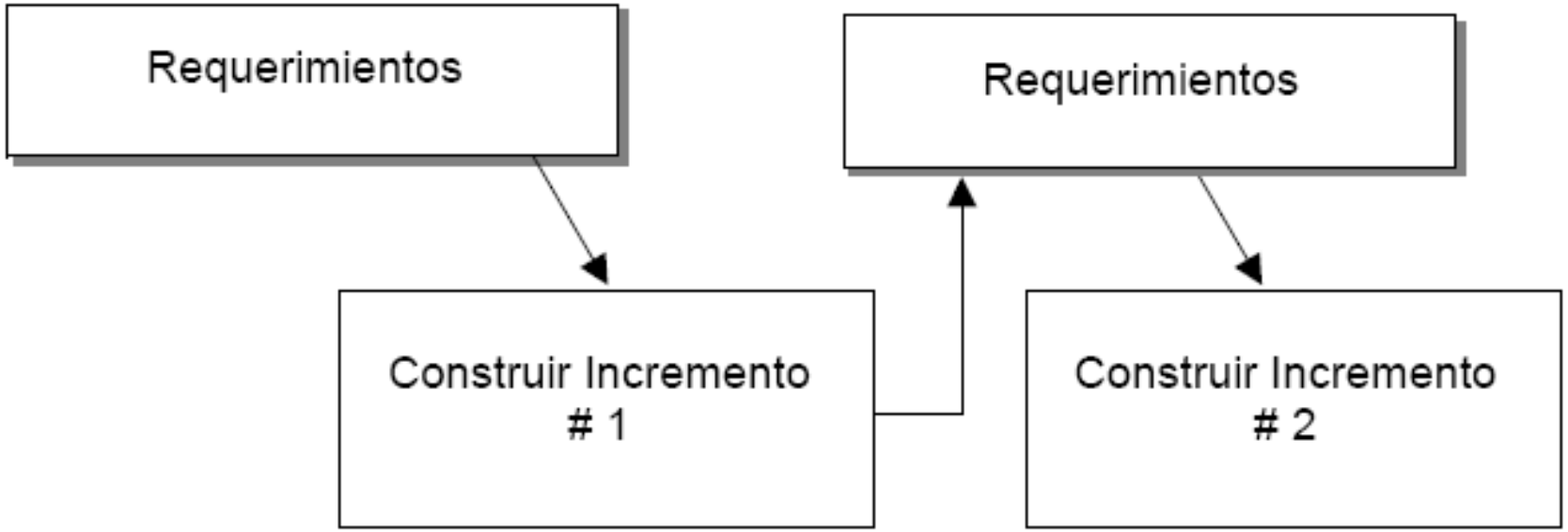
- **Ciclo de Vida del Desarrollo de Software INCREMENTAL:**



Modelo de Desarrollo Incremental con desarrollo en cascada de los incrementos.

3. Ciclo de vida de los proyectos de software

- **Ciclo de Vida del Desarrollo de Software EVOLUTIVO:**



Modelo Evolutivo.

3. Ciclo de vida de los proyectos de software

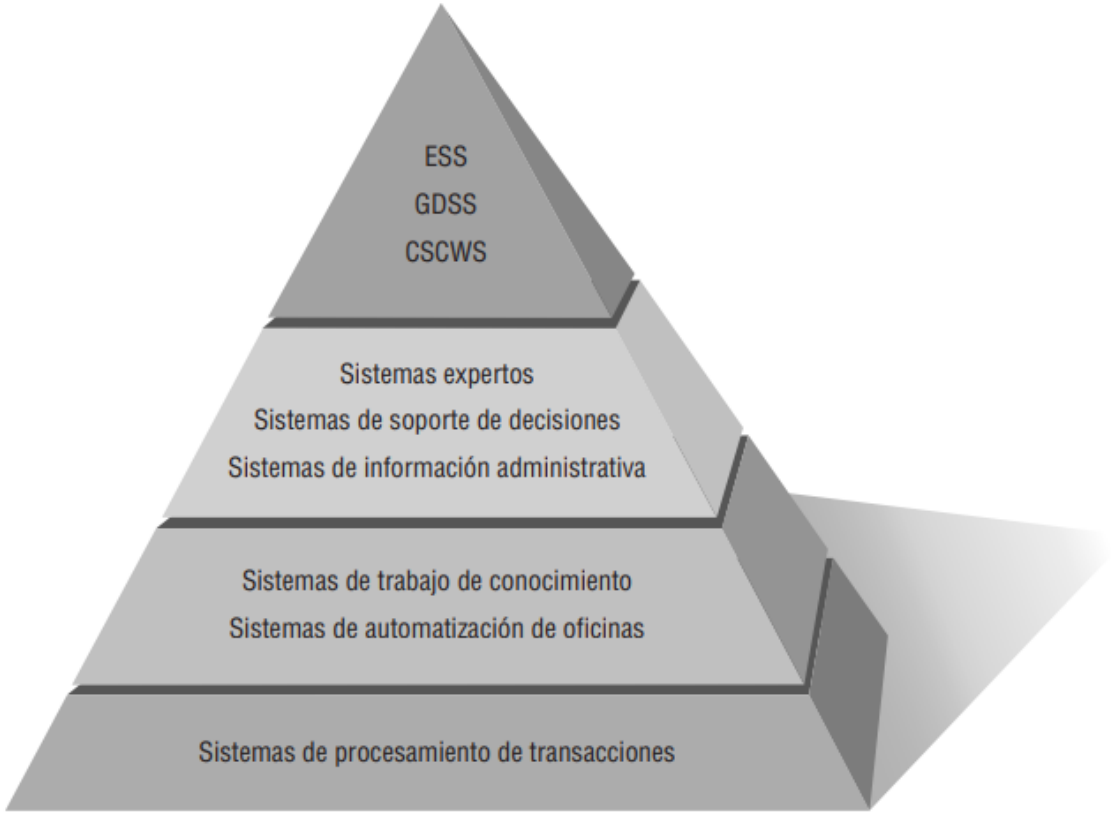
- **Seleccione el ciclo de vida para el siguiente caso práctico**

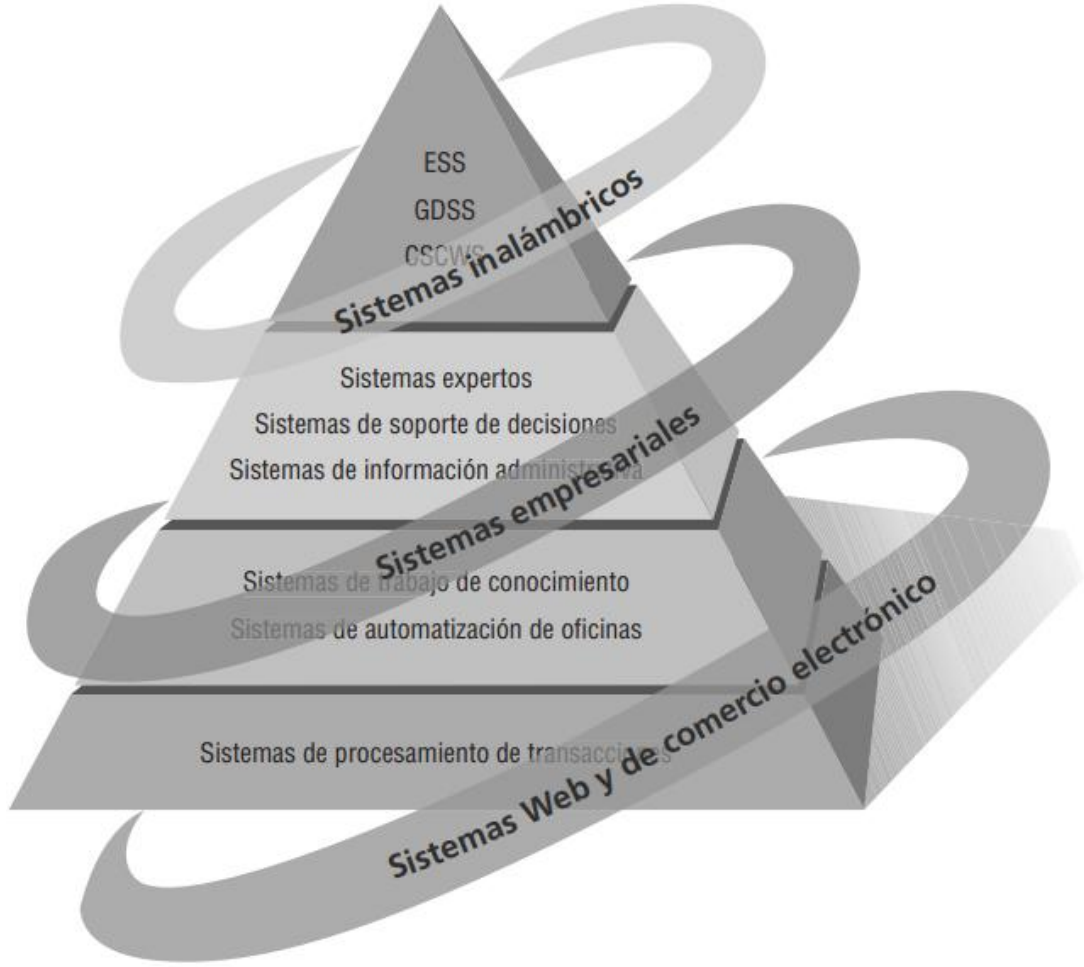
Un Disk Jockey desea tener un pequeño sistema de información y se identifican los siguientes requisitos:

R1: El sistema permitirá la administración (Ingreso, Modificación, eliminación) de Discos Compactos CD (Álbum musical).
R2: Un Disco contiene la siguiente Información:
Título del Disco.
Grupo Musical
Productor
Fecha de Producción
Título de la Canción
Nombre del Cantante
Autor de la Canción
Tiempo de duración
R3: El sistema permitirá consultar los datos del CD por su Título.
R4: El sistema generará un reporte del total de CD por grupo musical.
R5: Se generará un reporte sobre el tiempo de reverberación tomando como coeficiente de absorción el promedio de los materiales del local y asumiendo un volumen cúbico.
Existen otros requisitos que el cliente aún no los tiene claros en esta etapa.

4. Tipos de proyectos de software

- Evolución de los sistemas de Software:





4. Tipos de proyectos de software

Se pueden clasificar de muchas maneras diferentes:

Por su función:

- **Desarrollo de software a medida:** Creación de software específico para satisfacer las necesidades únicas de un cliente o empresa.
- **Desarrollo de software comercial:** Creación de software para ser vendido a un amplio mercado.
- **Desarrollo de software de código abierto:** Creación de software con código fuente abierto, disponible para que cualquiera lo utilice, modifique y distribuya.
- **Mantenimiento de software:** Actualización y mejora de software existente para corregir errores, mejorar el rendimiento y añadir nuevas funcionalidades.
- **Migración de software:** Transferencia de software de un entorno a otro, como de un sistema operativo a otro o de un servidor a otro.

4. Tipos de proyectos de software

Por su metodología de desarrollo:

- **Modelo en cascada:** Enfoque lineal y secuencial, donde cada fase debe completarse antes de pasar a la siguiente.
- **Modelo ágil:** Enfoque iterativo e incremental, donde el software se desarrolla en ciclos cortos y se entrega al cliente en etapas tempranas para obtener retroalimentación.
- **Modelo DevOps:** Enfoque que integra el desarrollo, las pruebas y las operaciones para acelerar la entrega de software y mejorar la calidad.

4. Tipos de proyectos de software

Por su tamaño y complejidad:

- **Proyectos pequeños:** Requieren pocos recursos y se pueden completar en un corto período de tiempo.
- **Proyectos medianos:** Requieren más recursos y tiempo que los proyectos pequeños.
- **Proyectos grandes:** Requieren una gran cantidad de recursos y tiempo, y a menudo involucran a múltiples equipos y tecnologías.

4. Tipos de proyectos de software

Por su industria o sector:

- **Software empresarial:** Software utilizado en empresas para gestionar procesos como finanzas, recursos humanos, ventas y marketing.
- **Software de salud:** Software utilizado en hospitales, clínicas y otras organizaciones de salud para gestionar registros médicos, programar citas y realizar diagnósticos.
- **Software educativo:** Software utilizado en escuelas y universidades para facilitar el aprendizaje y la enseñanza.
- **Software de entretenimiento:** Software utilizado para jugar, ver películas, escuchar música y realizar otras actividades de ocio.

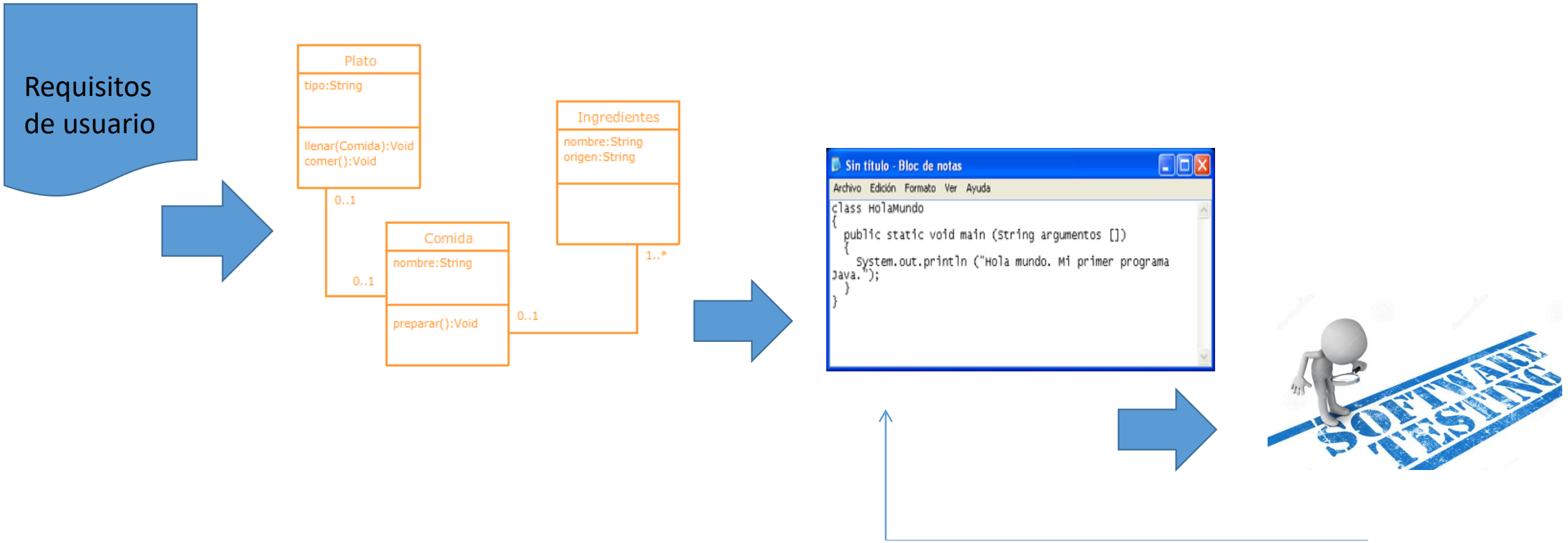
4. Tipos de proyectos de software

Por tipo de aplicación:

- **Aplicaciones web:** Estas aplicaciones se ejecutan en un servidor web y se acceden a través de un navegador web.
- **Aplicaciones móviles:** Estas aplicaciones se ejecutan en dispositivos móviles, como teléfonos inteligentes y tabletas.
- **Aplicaciones de escritorio:** Estas aplicaciones se ejecutan en computadoras personales.
- **Aplicaciones embebidas:** Estas aplicaciones se ejecutan en dispositivos electrónicos, como electrodomésticos y automóviles.
- **Videojuegos:** Estos proyectos se centran en el desarrollo de juegos para diferentes plataformas.
- **Aplicaciones de inteligencia artificial (IA):** Estos proyectos se centran en desarrollar sistemas que puedan realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje y la resolución de problemas.

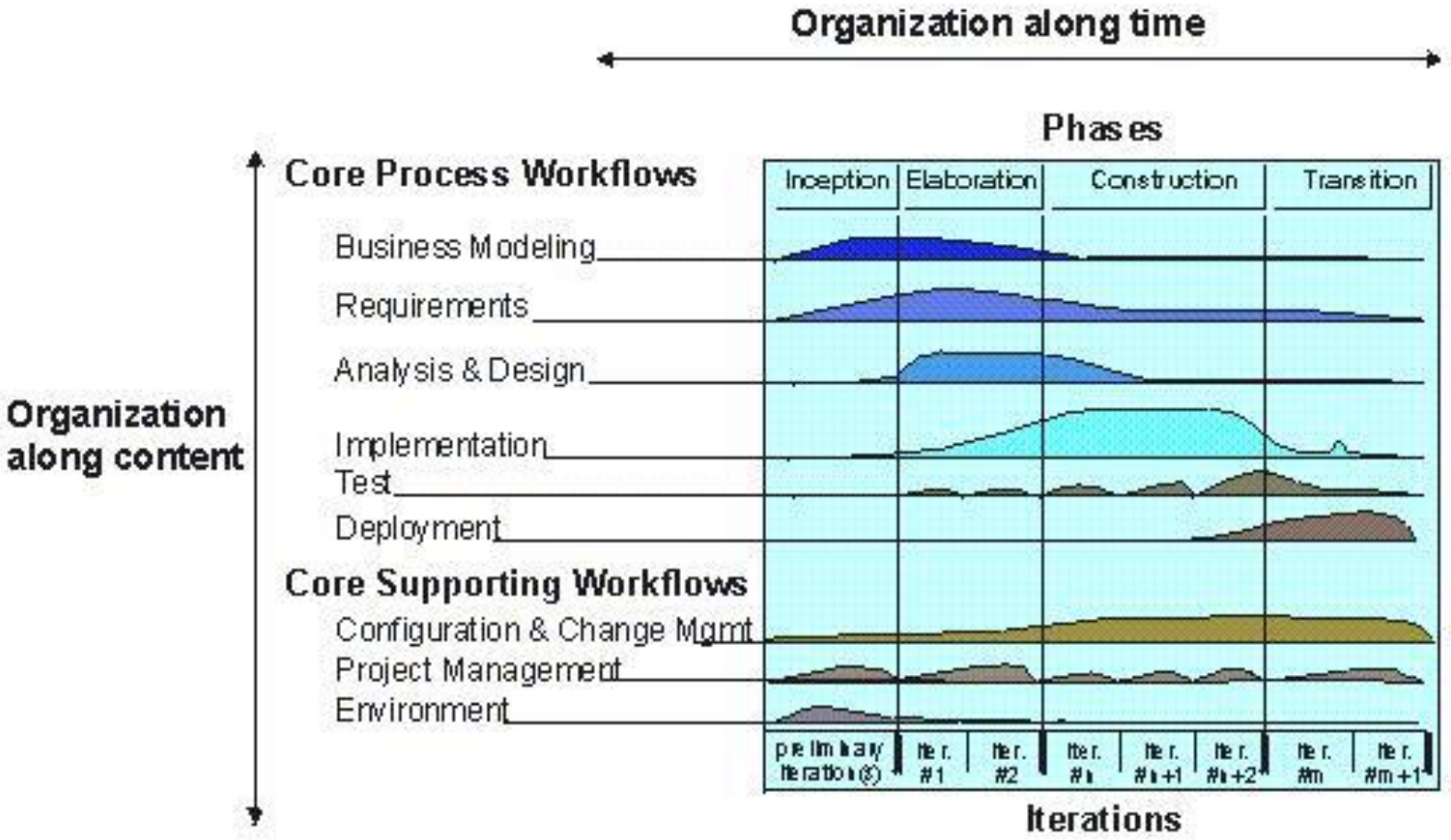
3. Procesos de dirección de proyectos de software

- Proceso de desarrollo de software basado en modelos



3. Procesos de dirección de proyectos de software

Fases de proyectos software
Proceso RUP (Inicio, Elaboración, Construcción, Transición)



Rational Unified Process - RUP

- RUP es una metodología de desarrollo de Software tradicional que pretende implementar las siguientes prácticas en ingeniería de software:
  - Desarrollo iterativo del software
  - Administración de requisitos
  - Uso de arquitecturas basadas en componentes
  - Modelamiento visual del software (basado en UML)
  - Verificación de la calidad del software
  - Control de cambios

3. Procesos de dirección de proyectos de software

Productos: FASE DE INICIO

- Un documento de visión general:
  - Requisitos generales del proyecto
  - Características principales
  - Restricciones
- Modelo inicial de casos de uso (10% a 20 % listos).
- Glosario.
- Caso de negocio:
  - Contexto
  - Criterios de éxito
  - Pronóstico financiero
- Identificación inicial de riesgos.
- Plan de proyecto.
- Uno o más prototipos.

3. Procesos de dirección de proyectos de software

Productos: FASE DE ELABORACIÓN

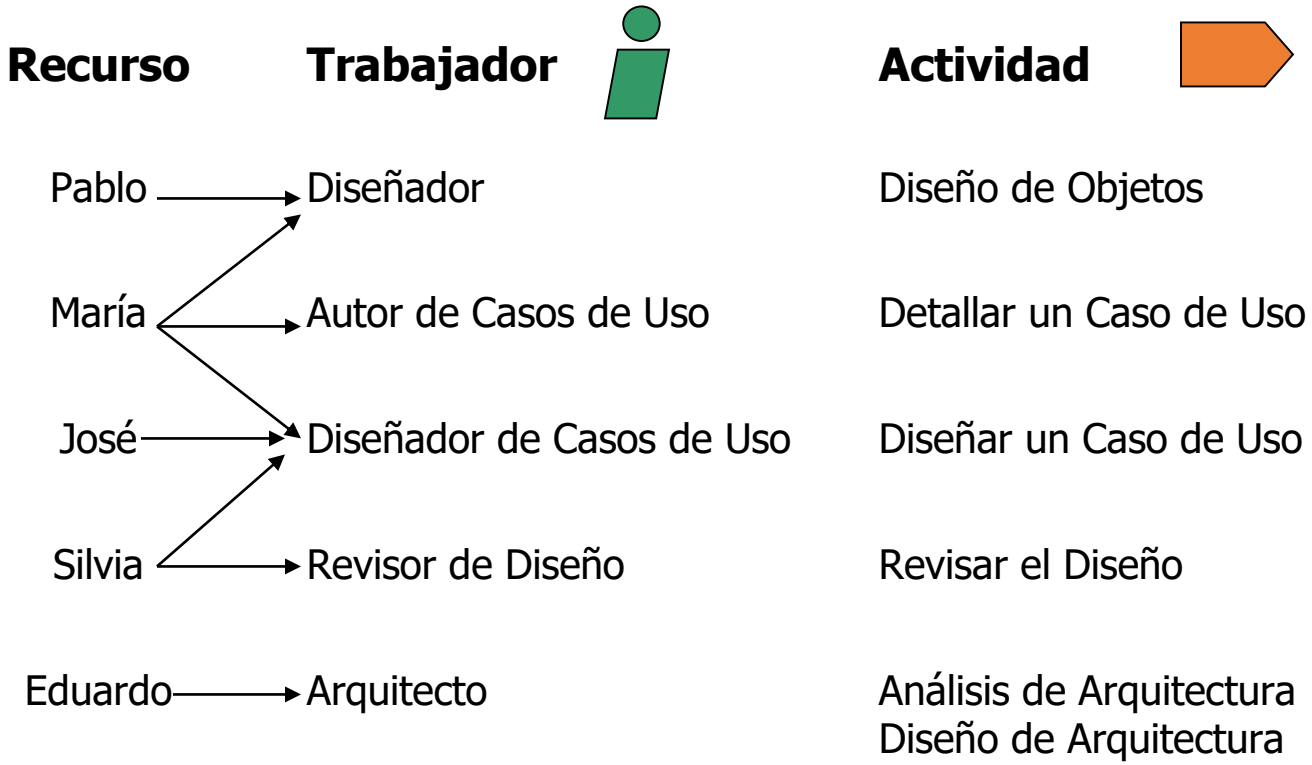
- Modelo de casos de uso (80% completo) con descripciones detalladas.
- Otros requisitos no funcionales o no asociados a casos de uso.
- Descripción de la Arquitectura del Software.
- Un prototipo ejecutable de la arquitectura.
- Lista revisada de riesgos y del caso de negocio.
- Plan de desarrollo para el resto del proyecto.
- Un manual de usuario preliminar.

3. Procesos de dirección de proyectos de software

Productos: FASE DE CONSTRUCCIÓN

- El producto de software integrado y corriendo en la plataforma adecuada.
- Manuales de usuario.
- Una descripción del “release” actual.

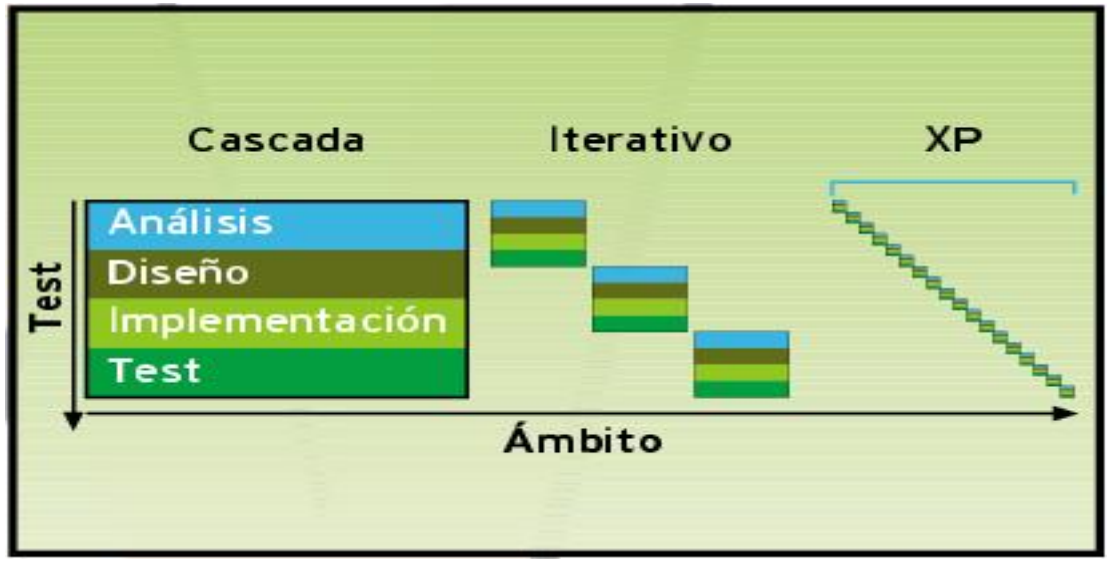
Asignación de Actividades



4. Introducción al Desarrollo de Software Ágil

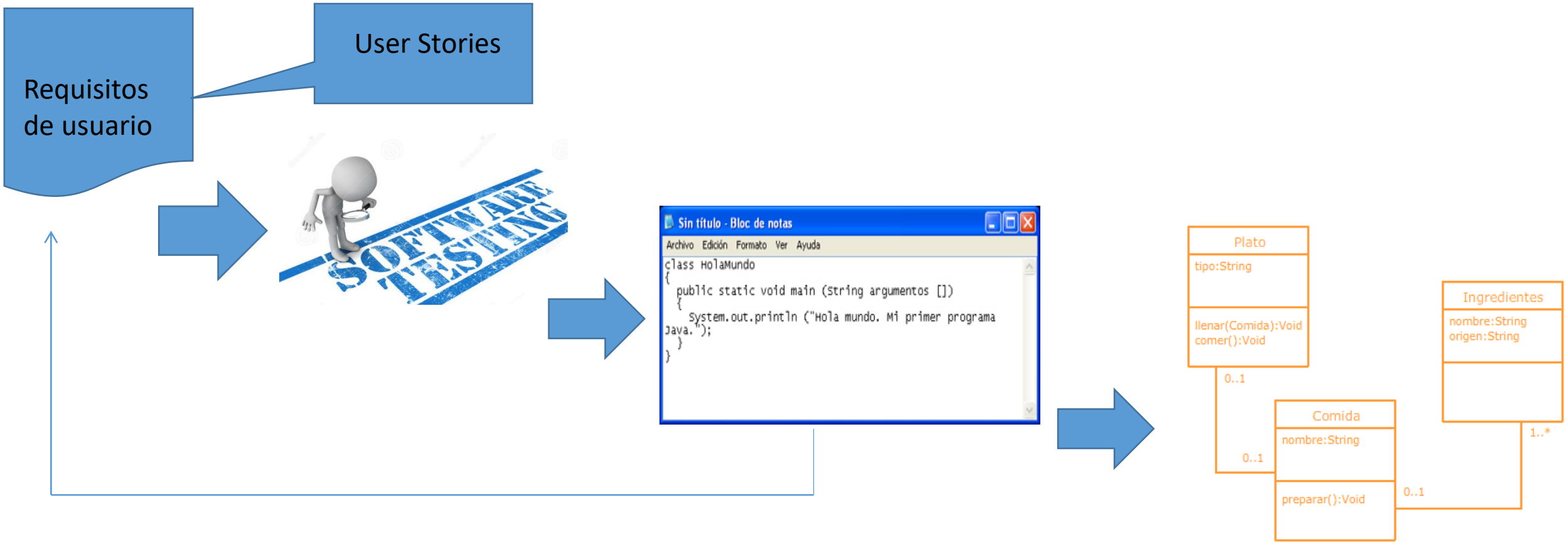
¿Como abordan las metodologías ágiles los problemas del desarrollo de software?

- Retrasos y desviaciones: versiones cortas.
- Cancelan el proyecto: entregas periódicas.
- Sistemas deteriorados y defectos: pruebas continuas.
- Requisitos mal comprendidos: cliente dentro del equipo.
- Cambios de negocio: versiones cortas.
- Falsa riqueza de características: realizar tareas prioritarias.
- Cambios de personal: anima el contacto y la integración.



4. Introducción al Desarrollo de Software Ágil

- Proceso de desarrollo de software basado en pruebas



4. Introducción al Desarrollo de Software Ágil: Historias Usuario

- ✓ Relato acerca de qué problema debe resolver el sistema. Representa una parte de la funcionalidad del sistema que es coherente para el cliente.
- ✓ La historia de usuario debe responder a tres preguntas: ¿Quién se beneficia?, ¿qué se quiere? y ¿cuál es el beneficio?..

Historia: Responder a comentarios

Como: Lector del Blog

Quiero: adicionar comentarios a las entradas y responder a comentarios de otros lectores

Para: mantenerme en contacto con los demás usuarios del blog

3

Como (rol) quiero (algo) para poder (beneficio).

Mike Cohn

4. Introducción al Desarrollo de Software Ágil

- Desde su introducción en la década de los 90's, las metodologías ágiles (Scrum, XP, AUP,..) han venido ganando adeptos y actualmente se configuran como una de las aproximaciones más utilizadas para desarrollar software.
- Las metodologías ágiles se basan en una serie de prácticas, tales como la programación por pares o el desarrollo dirigido por pruebas (TDD, Test-Driven Development).
- Las técnicas de desarrollo ágil prometen mejorar la **calidad** del producto software y la **productividad** de los desarrolladores.
- Conocidos anteriormente como Metodologías Livianas, los procesos ágiles de desarrollo de software se enfocan en la gente y los resultados.

4. Introducción al Desarrollo de Software Ágil

- Minimizar la cantidad de esfuerzo y tiempo gastados en construir modelos que en general servirán como documentación.
- Asegurar que el software entregado funciona para los usuarios.
- Permitir que el proyecto se adapte de manera flexible e inmediata a los cambios originados por tecnologías y/o requisitos.

4. Introducción al Desarrollo de Software Ágil

- Programación extrema (XP)
- Metodologías Crystal
- SCRUM
- Desarrollo de software adaptativo
- Desarrollo guiado por características (FDD)
- Metodología de desarrollo de sistemas dinámicos (DSDM)

¿Qué es XP?

- “Un proceso ligero, de bajo riesgo, flexible, predecible, científico y divertido de desarrollar software”.

Kent Beck

(Extreme Programming Explained)

Características de XP

- Metodología creada a base de prueba y error.
- Surge considerando 5 valores que pueden mejorar cualquier proyecto de software: Simplicidad, Comunicación, Realimentación, Coraje.
- Expresada en forma de prácticas (algunas ya existentes desde hace años), que se soportan las unas a las otras y conforman un conjunto completo. (Manifiesto Ágil http://www.agilemanifesto.org/iso/es/)

Los 4 valores

- Simplicidad: XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación. Es mejor hacer algo simple hoy, que hacerlo más complicado hoy y probablemente nunca usarlo.
- Comunicación: Algunos problemas en los proyectos tienen su origen en que alguien no dijo algo a alguien más sobre algo importante en algún momento. XP hace casi imposible la falta de comunicación.

Los 4 valores

- Realimentación: retroalimentación concreta y frecuente del cliente, del equipo y de los usuarios finales da una mayor oportunidad de dirigir el esfuerzo.
- Coraje: se requiere coraje para confiar en que la retroalimentación durante el camino es mejor que tratar de adivinar todo con anticipación. Se requiere valor para comunicarse con los demás cuando eso podría exponer la propia ignorancia. Se requiere valor para mantener el sistema simple, dejando para mañana las decisiones de mañana. Y, sin un sistema simple, comunicación constante y retroalimentación, es difícil ser valeroso.
- Respeto: Todo el mundo da y siente el respeto que merecen como miembro del equipo. Todo el mundo aporta valor incluso si es simplemente entusiasmo. Los desarrolladores respetan la experiencia del cliente y viceversa. El gestor del proyecto respeta nuestro derecho a aceptar la responsabilidad y recibir autoridad sobre nuestro propio trabajo.

XP en la práctica

- Retroalimentación a escala fina:
  - ✓Desarrollo guiado por pruebas
  - ✓Planificación iterativa
  - ✓Cliente como parte del equipo
  - ✓Programación en pares

- Proceso continuo:
  - ✓Integración continua
  - ✓Refactorización
  - ✓Liberación pequeña, entregas frecuentes

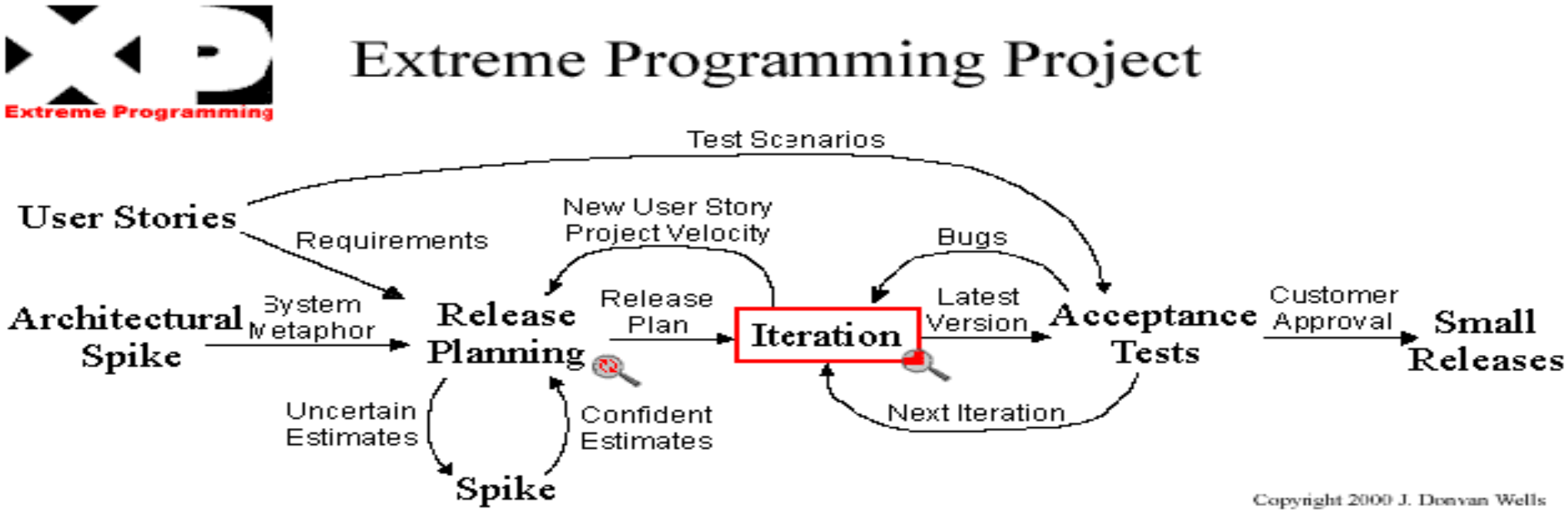
XP en la práctica

- Entendimiento compartido:
  - ✓ Diseño simple
  - ✓ Metáforas del sistema
  - ✓ Propiedad colectiva del código
  - ✓ Estándares de codificación

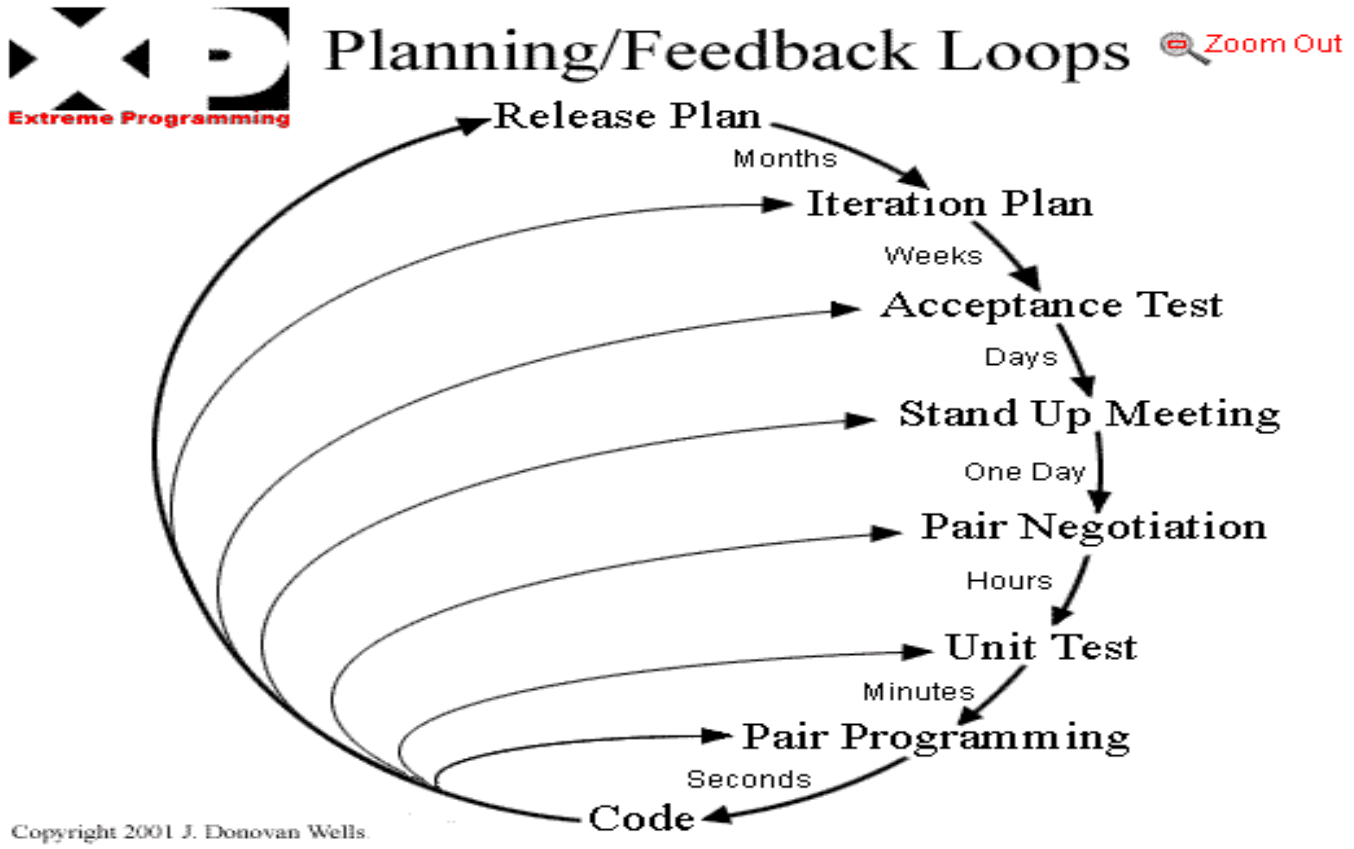
- Bienestar del programador:
  - ✓ Ritmo sostenible (Semanas de 40 horas)

4. Introducción al Desarrollo de Software Ágil

Proceso de desarrollo de software con XP



Proceso de desarrollo de software con XP



Desarrollo basado en Scrum

- ✓ Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80.

- ✓ En 1996 se definió por primera vez un patrón para aplicar esos principios de desarrollo en “campos de scrum” al software.

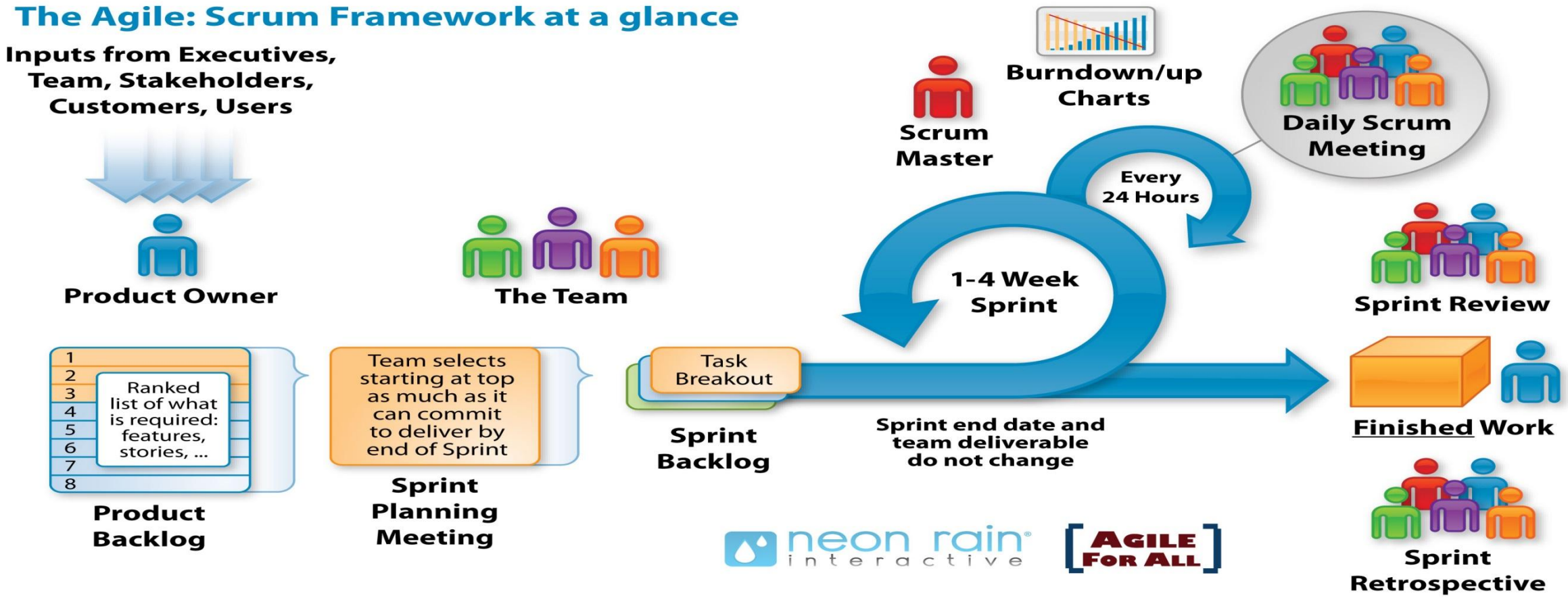
- ✓ Esta fue la primera definición de un patrón Scrum aplicado al software, diseñada por Jeff Sutherland y Ken Schwaber y presentada en OOPSLA 96

Desarrollo basado en Scrum:

Scrum es un método adaptativo de gestión de proyectos que se basa en los principios ágiles:

- Colaboración estrecha con el cliente.
- Predisposición y respuesta al cambio
- Prefiere el conocimiento tácito de las personas al explícito de los procesos
- Desarrollo incremental con entregas funcionales frecuentes
- Comunicación verbal directa entre los implicados en el proyecto
- Motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso.
- Simplicidad. Supresión de artefactos innecesarios en la gestión del proyecto.

Proceso de desarrollo de software con SCRUM



Ejemplo TDD Calculadora

Realizar una calculadora para enseñanza de las operaciones básicas a niños en edad escolar con las siguientes características:

US1. Como profesor quiero que la calculadora me permita realizar sumas de números enteros del 0 al 10 sin negativos, para que los niños aprendan el concepto de adición.

US2. Como profesor necesito que la calculadora también pueda realizar restas de números, donde el minuendo debe ser mayor que el sustraendo y solo permita valores del 0 al 10, sin negativos para que los alumnos aprendan el concepto de la substracción

US3. Como profesor quiero que la calculadora permita realizar multiplicaciones de un solo dígito entero no negativo del 0 al 9 para poder enseñar el concepto de multiplicación de números

US4. Como docente espero que la calculadora permita realizarla división de números enteros del 1 al 10 , donde el numerador debe ser mayor que el denominador, para así poder enseñar el concepto de la división de números.

DEFINICIÓN DE CASOS DE PRUEBA

- REALIZAR LAS PARTICIONES DE EQUIVALENCIA Y EL ANÁLISIS DE VALORES LÍMITE PARA CADA HISTORIA DE USUARIO Y DETERMINAR UNA LISTA DE CASOS DE PRUEBA DE CAJA NEGRA.

PARTICIONES DE EQUIVALENCIA Y ANÁLISIS DE VALORES LÍMITE			
HISTORIA DE USUARIO	**TIPO DE DATO**	**CLASES DE EQIVALENCIA VÁLIDAS**	**CLASES DE EQUIVALENCIA INVÁLIDAS**
US1	OPERADOR A (RANGO [0..10]) OPERADOR B (RANGO [0..10])	0, 1, 9,10 0, 1, 9,10	-1,11,1.1 -1,11,1.1
US2	MINUENDO (RANGO [0..10]) SUSTRAENDO (RANGO [0..10])	1,9,10 0,8,10	0,-1,11,1.1 1,-1,11,1.1

CASOS DE PRUEBA			
HISTORIA DE USUARIO	**NO. CASO**	**ENTRADA**	**SALIDA ESPERADA**
US1	US11	0+0	0
	US12	1+1	2
	US13	9+9	18
	US14	10+10	20
	US15	-1-1	Error No negativos
	US16	11+11	Error No mayores de 10
	US17	1.1+1.1	Error No decimales
US2	US21	1-0	1
	US22	9-8	1
	US23	10-10	0
	US24	0-1	Error minuendo > sustraendo
	US25	-1-1	Error No negativos
	US26	11-11	Error No mayores de 10
	US27	1.1-1.1	Error No decimales