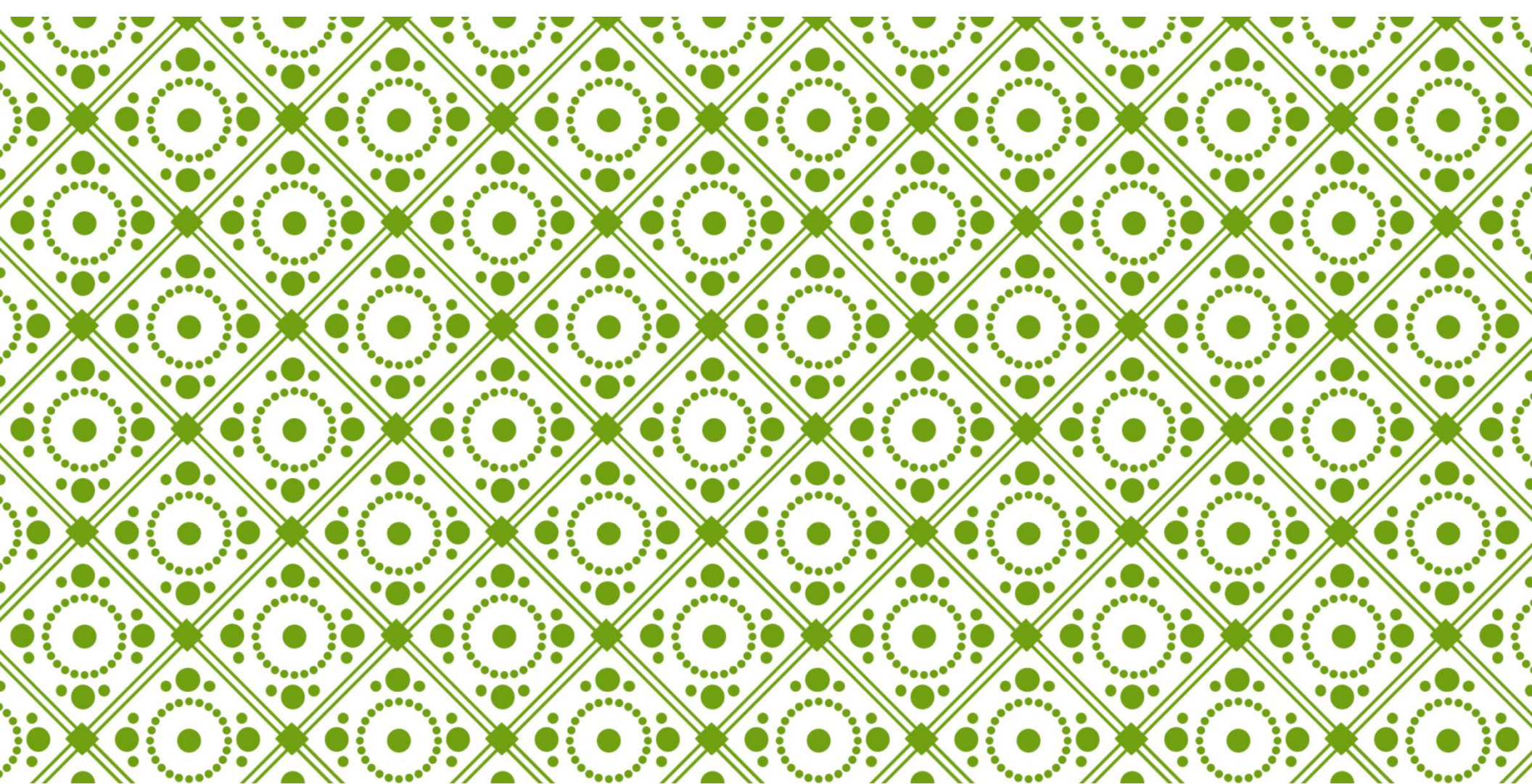


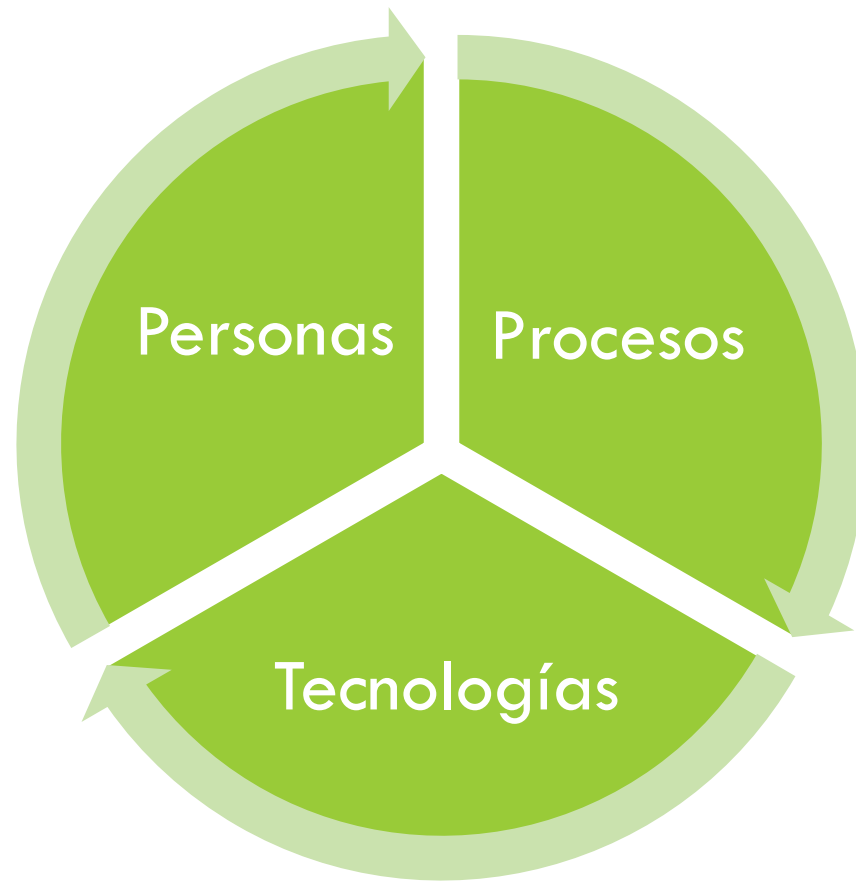
TECNOLOGÍA DE LA INFORMACIÓN EN LAS ORGANIZACIONES

Cursada 2024

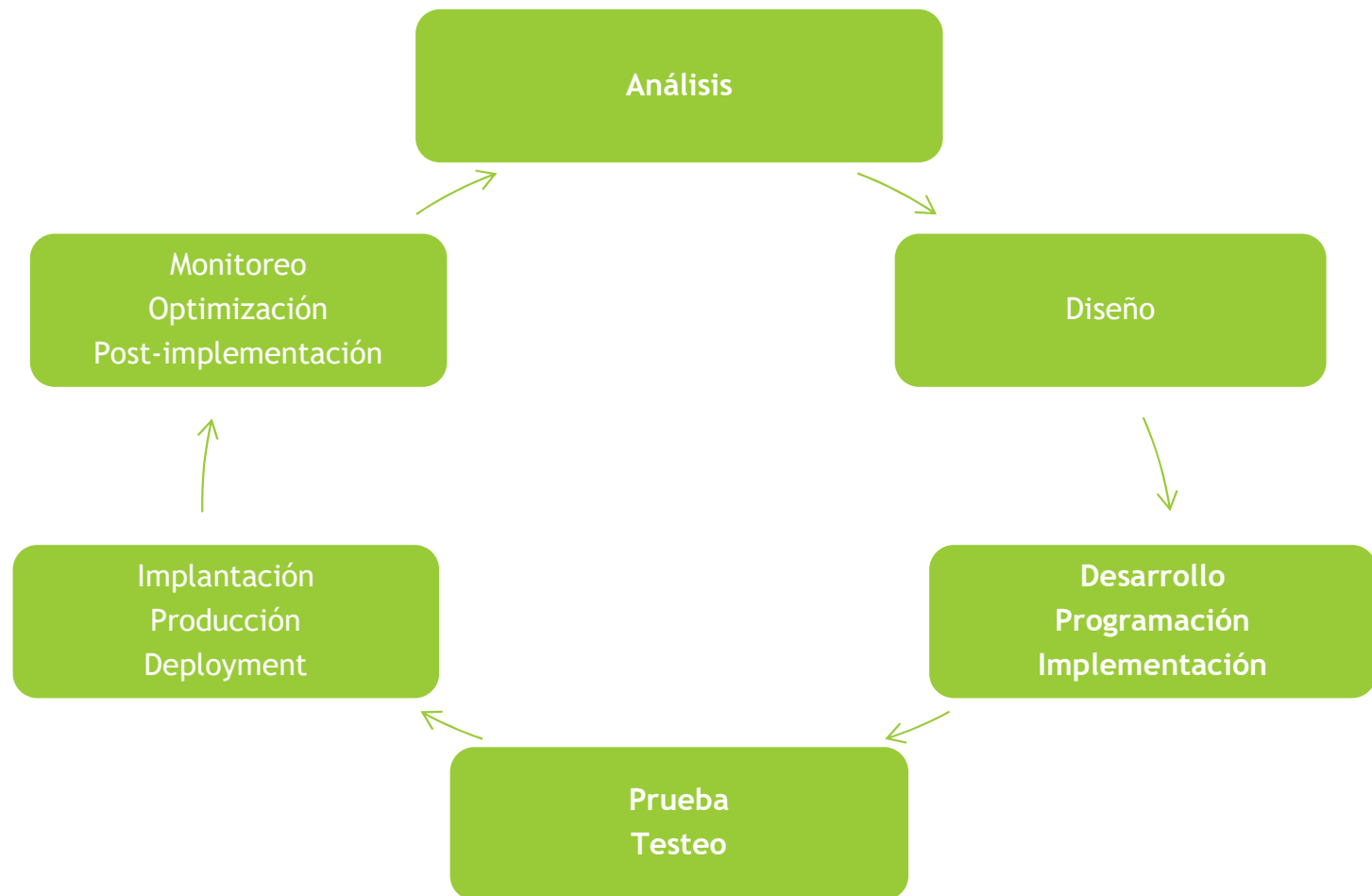


DESARROLLO DE SISTEMAS

ELEMENTOS CLAVE A TENER EN CUENTA PARA EL DESARROLLO EXITOSO



CICLO DE VIDA CLÁSICO PARA EL DESARROLLO DE SISTEMAS



Análisis	Diseño	Implementación	Prueba	Implantación	Optimización
Definición del problema, identificación de la solución, análisis de factibilidad, estimación de esfuerzo, recursos y duración, identificación de riesgos y especificación de requerimientos.	<ul style="list-style-type: none"> - Si se trata de realizar el desarrollo: Diseño lógico y Físico. - Si se trata de adquisición de sistema existente: identificación de las partes a configurar y adaptaciones a realizar 	<ul style="list-style-type: none"> - Si se trata de un nuevo desarrollo: Codificación del sistema. - Si se trata de adquisición de sistema existente: configuración y parametrización del sistema. 	<p>Comprobación del funcionamiento del sistema:</p> <ul style="list-style-type: none"> - Pruebas unitarias - Prueba de Sistemas - Pruebas de Aceptación de Usuario. <p>Otras clases de pruebas.</p> <p>Capacitaciones</p>	<p>Implantar el nuevo sistema.</p> <p>Estrategias posibles:</p> <ul style="list-style-type: none"> - Paralela - Cambio Directo - Estudio Piloto - Por Fases 	<p>Monitoreo del sistema para detectar:</p> <ul style="list-style-type: none"> - Errores - Modificaciones - Mejoras


CICLO DE VIDA CLÁSICO

METODOLOGÍAS Y HERRAMIENTAS PARA EL DESARROLLO DE SISTEMAS



METODOLOGÍAS Y HERRAMIENTAS PARA EL DESARROLLO DE SISTEMAS

Metodología: conjunto de métodos que se utilizan para cubrir las actividades de un proceso determinado.



Metodología de Desarrollo de Sistemas: conjunto de métodos que brindan soporte a cada una de las actividades dentro de las fases de un proyecto de sistemas.

**Metodologías
Estructuradas**

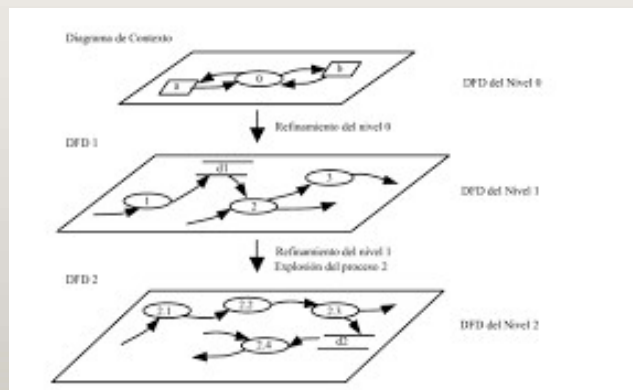
**RUP (Rational
Unified
Process)**

**Desarrollo
orientado a
Objetos**

**Métodos
Agiles**

**Reingeniería
del Software**

METODOLOGÍAS ESTRUCTURADAS



Técnicas orientadas a los procesos más que a los datos.

Son descendentes: desde el nivel más alto y abstracto hasta el nivel más bajo y detallado.

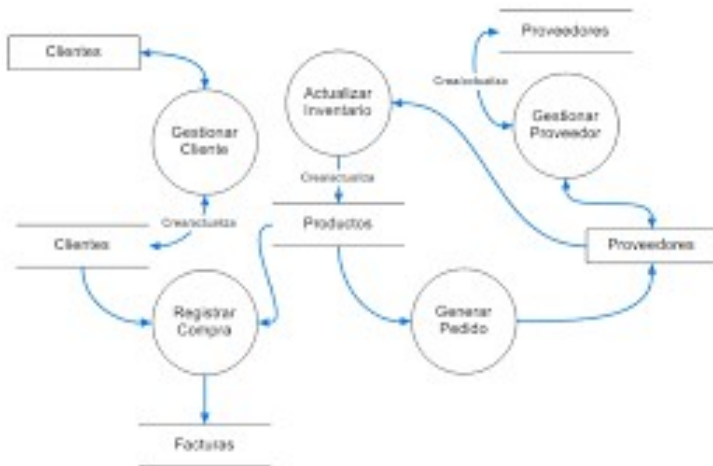
Utilizadas para el enfoque de ciclo de vida tradicional.

Incluyen:

Análisis estructurado → DFD, Especificaciones de Procesos.

Diseño estructurado → Diagrama de estructura

Programación estructurada → Diagrama de Flujo de Sistemas, Estructuras de control (secuencia, selección, iteración), Modularización

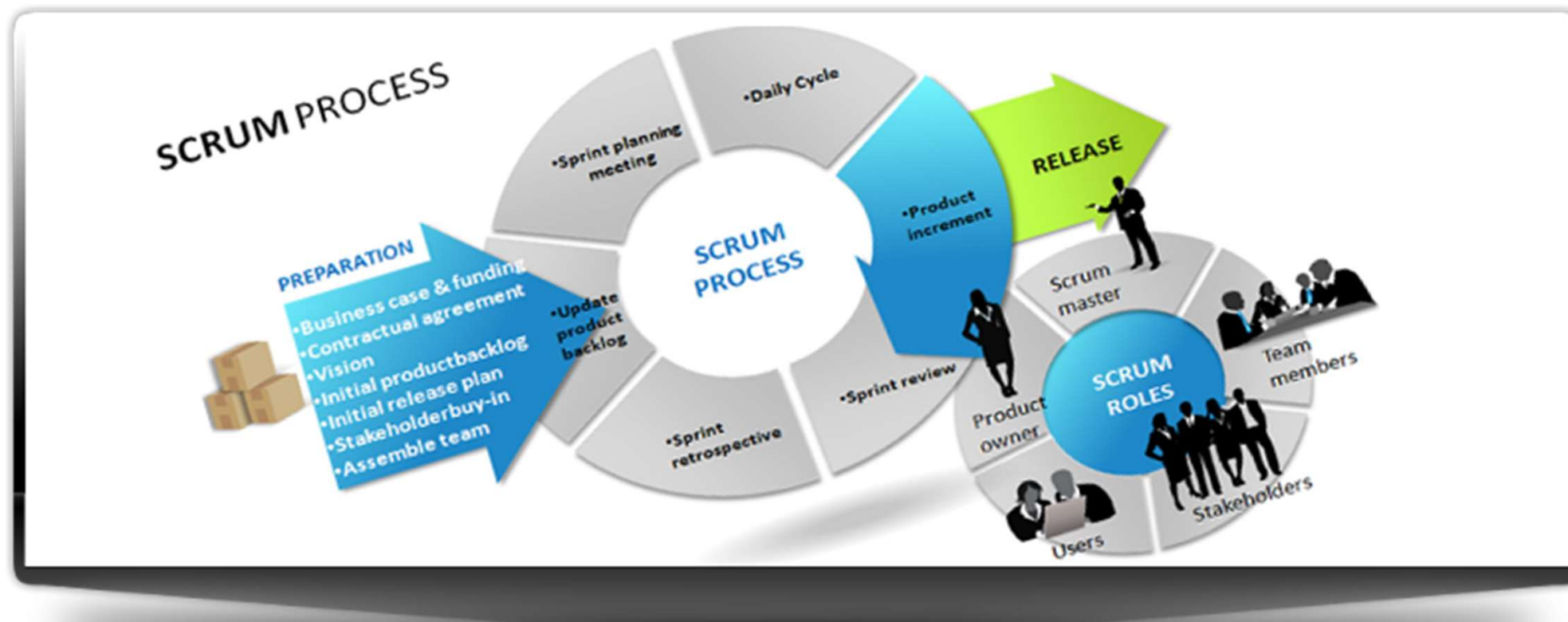




MÉTODOS AGILES

- ✓ Proceso de Creación de Sistemas funcionales en tiempo muy corto.
- ✓ Proceso no secuencial. Partes clave del desarrollo se realizan en paralelo.
- ✓ Utilizadas para el enfoque de prototipos y con herramientas de cuarta generación.
- ✓ Colaboración estrecha entre usuarios y especialistas de sistemas.
- ✓ No se genera casi documentación.
- ✓ Técnicas utilizables: diseño conjunto de aplicaciones (Joint Application Design –JAD–, Scrum, XP).

MÉTODOS AGILES



RUP(RATIONAL UNIFIED PROCESS)

✓ Metodología que divide el proceso en 4 fases:

1. Inicio,
2. Elaboración,
3. Construcción
4. Transición.

✓ Utilizable para cualquier tipo de proyecto. Cada fase tiene sus actividades asociadas

✓ Metodología iterativa con desarrollo incremental o en cascada

✓ La documentación se basa en ciertos diagramas y para esto, utiliza el UML. Ejemplos de diagramas:

- Para el análisis: Casos de Uso, Diagramas de estados –
- Para el diseño: Diagramas de clase, Diagramas de componentes, Diagrama de comunicaciones, etc.)



DESARROLLO ORIENTADO A OBJETOS



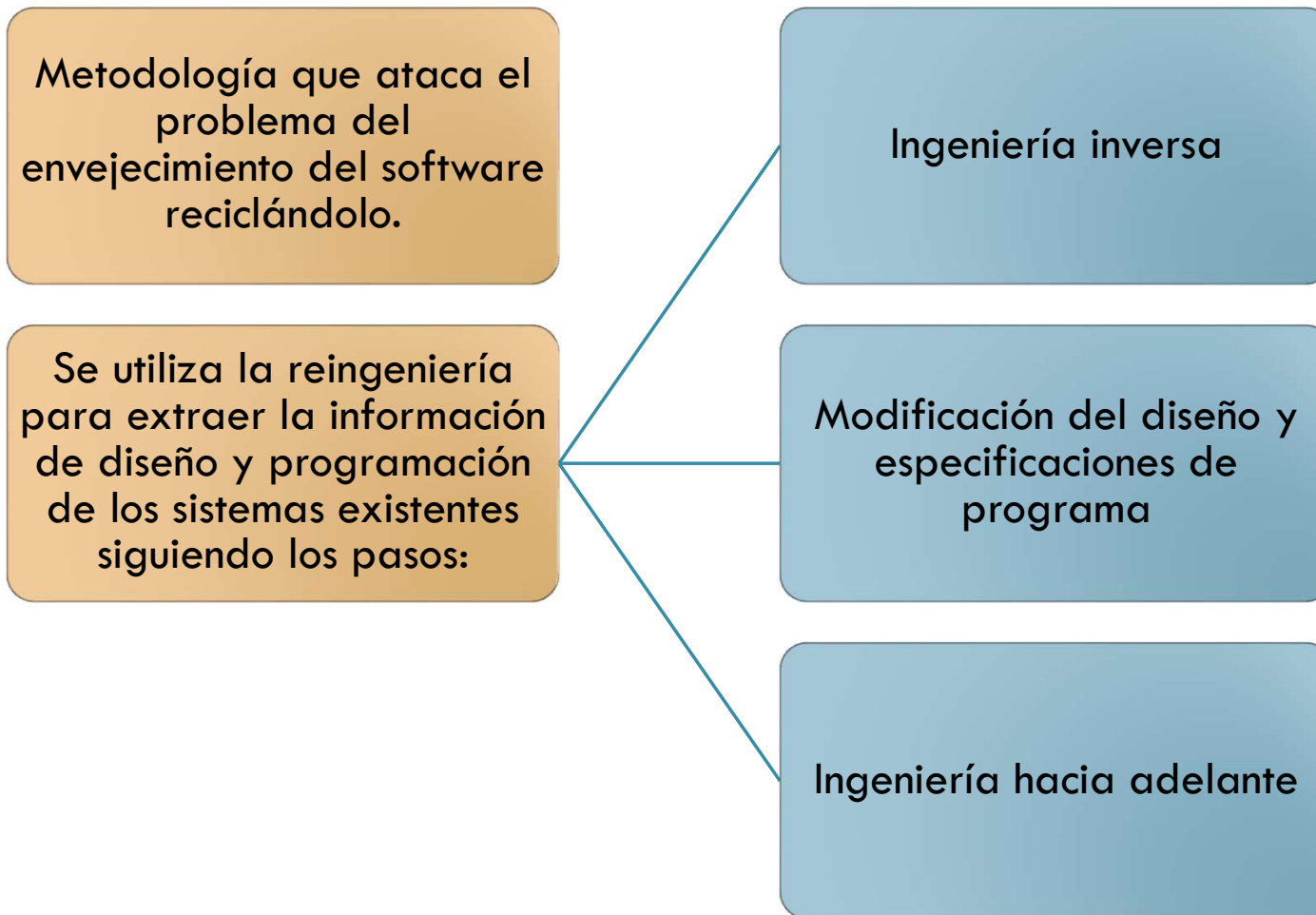
Objeto como unidad básica que encapsula sus datos y acciones que se pueden realizar sobre los mismos. Los objetos se relacionan entre si.

Los objetos tienen la característica que son reutilizables, se agrupan en clases y utilizan herencia.

Utilizado para enfoque híbrido entre ciclo de vida tradicional y prototipos

Desarrollo de frameworks
→ Reusabilidad

REINGENIERÍA DE SOFTWARE



TENDENCIAS ACTUALES

- Las condiciones del entorno actual (globalización, negocios en línea, mercado cambiante, transformación digital) demandan:

Desarrollos incrementales con integración continua

Componentes de software fáciles de agregar, modificar, reemplazar o reconfigurar (sistemas flexibles).

Sistemas escalables.

Green computing

Mobile computing

Cloud computing

Las organizaciones adoptan procesos de desarrollo más cortos para aplicaciones a compartir con proveedores, clientes y/o socios de negocios que proporcionen soluciones rápidas y no desestabilicen sus sistemas de procesamiento de transacciones y bases de datos organizacionales esenciales.

CADENA DE
VALOR EN EL
DESARROLLO
DE SISTEMAS



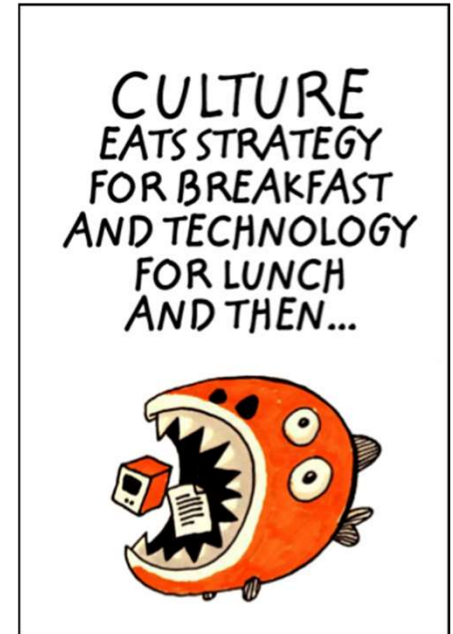
ÉXITO Y FRACASO

Los proyectos de Desarrollo de sistemas fracasan por:

- ✓ Mayores costos de lo previsto → Costos y Recursos
- ✓ No terminan en el tiempo estimado → Tiempos
- ✓ No son operables → Alcance.

Principales áreas de problemas (origen en factores de organización):

- ✓ Diseño no compatible con la estructura, cultura y metas de la organización.
- ✓ Datos: Información ambigua, errónea, desglosada indebidamente o incompleta.
- ✓ Costos: Los gastos excesivos no pueden justificarse con el valor que el sistema proporciona al negocio.
- ✓ Operaciones: Información que no se proporciona en forma oportuna o eficiente por fallas en el procesamiento de la misma.





ROLES EN LOS EQUIPOS DE DESARROLLO DE SOFTWARE

ROL

Es la tarea (**responsabilidad**) asignada a cada persona en un equipo de trabajo.

Depende de:

- Metodología de desarrollo
- Envergadura del proyecto
- Madurez tecnológica de la empresa y de sus procesos de negocios
- Versatilidad, aptitudes y actitudes de las personas involucradas

Cada rol tiene sus demandas actitudinales y aptitudinales



EJEMPLOS DE ROLES EN TI

- ✓ Analista / Analista funcional
- ✓ Analista del Negocio
- ✓ Desarrollador de Front-end
- ✓ Desarrollador de back-end
- ✓ Desarrollador Full stack
- ✓ Tester/QA
- ✓ Arquitecto de Sistema
- ✓ Arquitecto de Aplicaciones
- ✓ Ingeniero de Sistemas
- ✓ Líder de proyecto
- ✓ Ingeniero de redes
- ✓ Gerente de Producto
- ✓ Administrador de Base de Datos
- ✓ Administrador del sistema
- ✓ Administrador de red
- ✓ Gerente de Proyecto
- ✓ Gerente de Desarrollo
- ✓ Líder técnico
- ✓ Diseñadores UX y UI
- ✓ ...

A Day in the Life of a...

Computer Programmer

Median Salary: \$84,280



Know computer languages



Write computer programs



Collaborate with other programmers



Test software programs

the balance



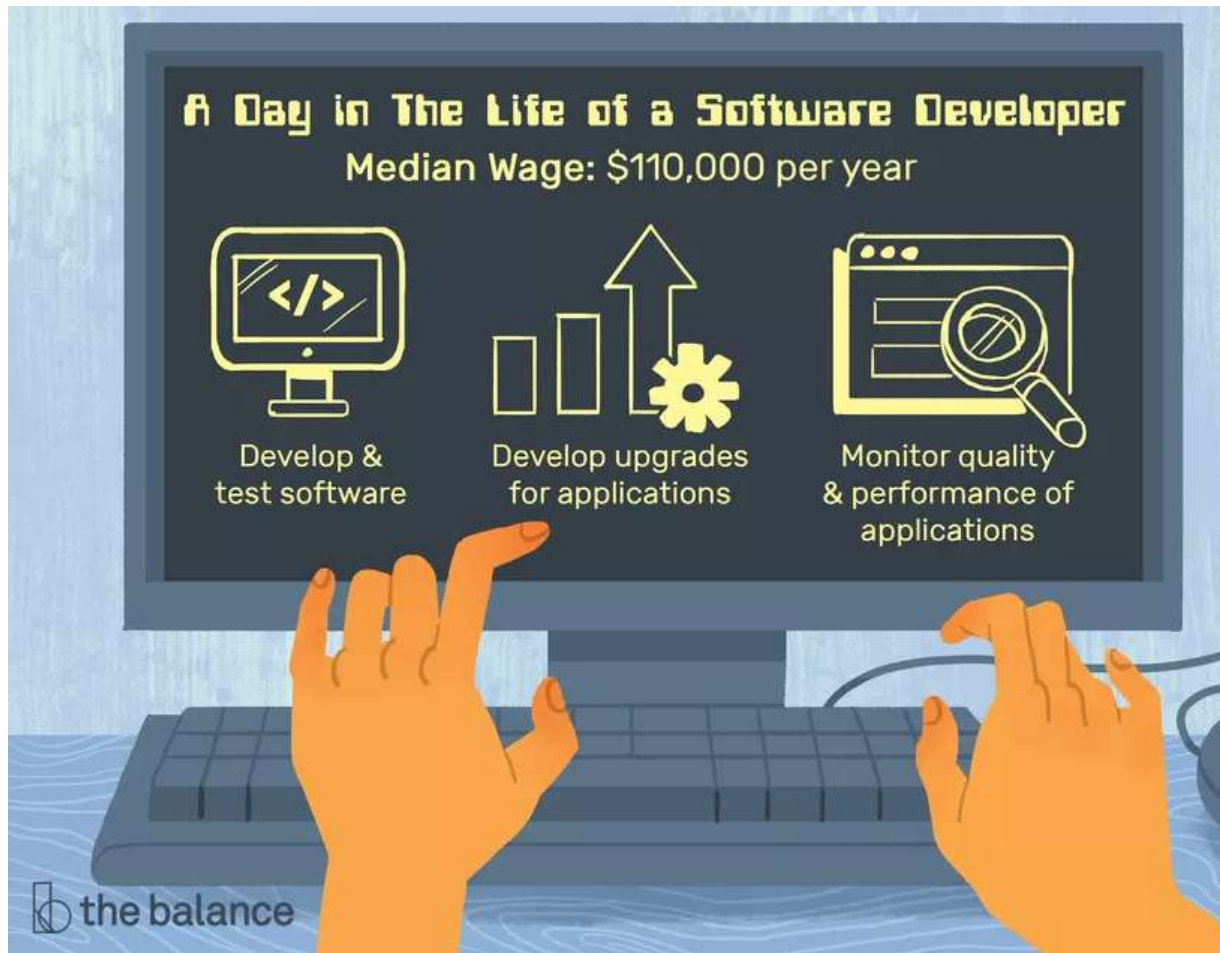
- ✓ Saber lenguajes de programación
- ✓ Escribir programas de computadora
- ✓ Actualizar programas de computadora
- ✓ Solucionar problemas de programas
- ✓ Probar programas de software
- ✓ Colaborar con otros programadores.

PROGRAMADOR: DEBERES Y RESPONSABILIDADES

PROGRAMADOR: HABILIDADES Y COMPETENCIAS

- ✓ **Pensamiento analítico:** los programadores informáticos necesitan comprender, manipular y reparar códigos informáticos complejos. Esto a veces implica tratar de aislar un problema que podría estar enterrado en algún lugar en miles de líneas de código, por lo que deben poder pensar en el problema y reducir dónde buscar.
- ✓ **Atención al detalle:** los programadores informáticos deben prestar atención a cada línea de código escrita. Un comando incorrecto y todo el programa podría funcionar mal.
- ✓ **Colaboración:** los programadores informáticos pueden necesitar la ayuda de otro departamento o colega para solucionar un problema de software. Es importante que tengan una mentalidad colaborativa. Los trabajos de programadores a menudo implican escribir software para agilizar el trabajo o resolver un problema de flujo de trabajo, y deben colaborar con quienes utilizarán el software.
- ✓ **Enfoque:** escribir programas de computadora implica largas horas escribiendo código o resolviendo problemas. Para tener éxito, los programadores deben poder mantener su atención centrada en el trabajo que están haciendo.



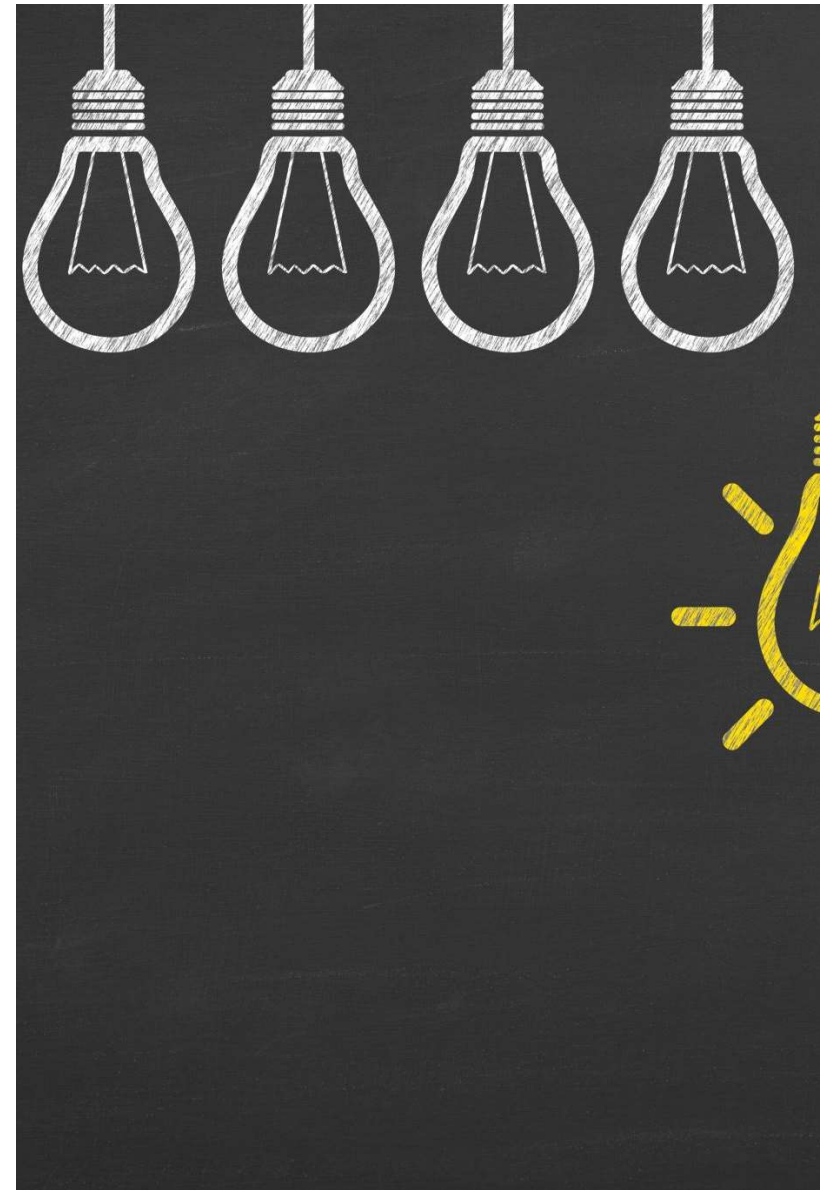


DESARROLLADOR DE SOFTWARE: DEBERES Y RESPONSABILIDADES

- ✓ Desarrollo y prueba de software para satisfacer las necesidades de los clientes.
- ✓ Desarrollar actualizaciones para aplicaciones existentes.
- ✓ Monitorea la calidad y el rendimiento de las aplicaciones mediante pruebas y mantenimiento.
- ✓ Documenta todo el trabajo para referencia futura.

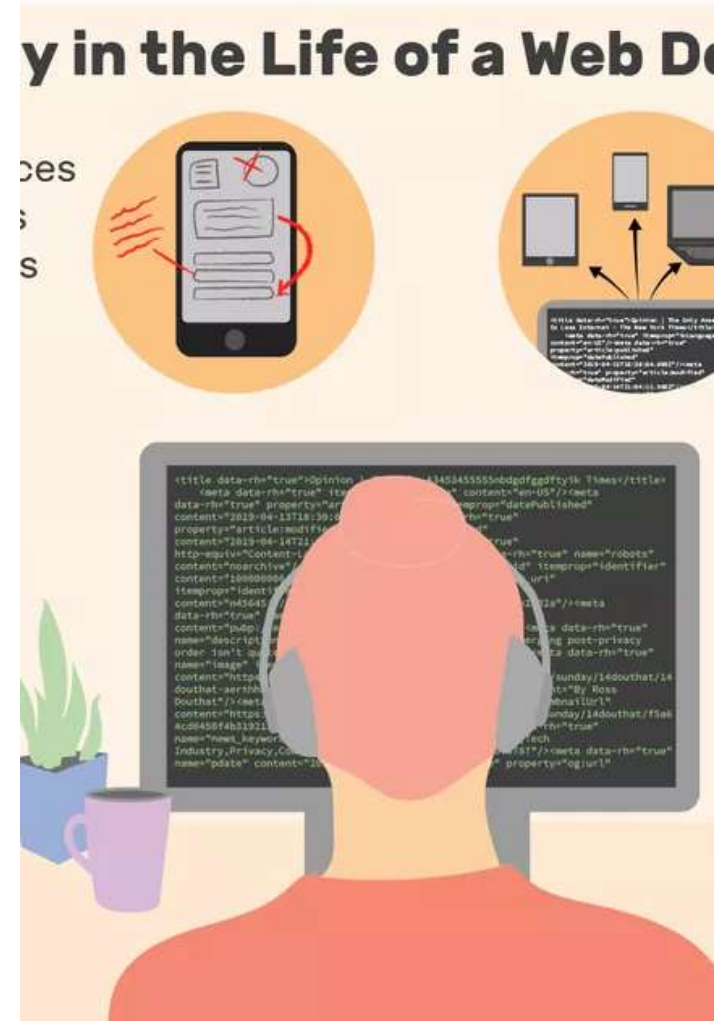
DESARROLLADOR DE SOFTWARE: HABILIDADES Y COMPETENCIAS

- ✓ **Creatividad e innovación:** las tuyas deben ser las mentes detrás de las nuevas capacidades de software, incluso cuando otros dicen que no se puede hacer.
- ✓ **Habilidades interpersonales y de comunicación:** no trabajará en un vacío. Este trabajo a menudo implica la colaboración con un equipo, y tendrá que poder comunicar de manera concisa y clara las instrucciones a los demás.
- ✓ **Habilidades analíticas:** tendrá que combinar las mejoras y las creaciones con las necesidades de los usuarios y clientes.
- ✓ **Concentración y enfoque:** el desarrollo de software implica numerosas y diminutas piezas entrelazadas. No puedes ser propenso a la distracción si quieres tener éxito.



DESARROLLADOR WEB: DEBERES Y RESPONSABILIDADES

- ✓ Conocer HTML, CSS, JavaScript, PHP y otros lenguajes de codificación de diseño web relevantes
- ✓ Crear y probar aplicaciones para sitios web.
- ✓ Colaborar
- ✓ Presente especificaciones de diseño
- ✓ Trabaja con gráficos y otros diseñadores.
- ✓ Solucionar problemas del sitio web
- ✓ Mantener y actualizar sitios web
- ✓ Monitorear el tráfico del sitio web
- ✓ Manténgase actualizado sobre tecnología



DESARROLLADOR WEB: HABILIDADES Y COMPETENCIAS

Orientado a detalles: una línea de código puede tener un impacto significativo en la funcionalidad o apariencia de un sitio web, y los desarrolladores web deben asegurarse de que no faltan detalles clave. Al resolver problemas, deben poder saber dónde buscar el problema.

Multitarea: los proyectos no siempre se manejan uno a la vez, y la emergencia de un cliente a veces puede llevar a otro proyecto a un segundo plano. Los desarrolladores web deben poder hacer malabarismos con múltiples proyectos sin perder plazos.

Auto motivado: el trabajo puede ser solitario a veces. Los desarrolladores web deben ser capaces de mantenerse en la tarea sin que nadie los distraiga.

Resolución de problemas: los sitios web deben ser funcionales y atractivos, y las necesidades de los clientes a este respecto pueden no ser siempre fáciles de satisfacer. Los desarrolladores web necesitan descubrir cómo traducir la visión de un cliente a un sitio web real y funcional.

Bueno bajo presión: los plazos ajustados son comunes al diseñar o actualizar sitios web. Los desarrolladores deben ser capaces de manejar la presión de hacer el trabajo cuando sea necesario.



GESTIÓN DE CONFIGURACIÓN DE SOFTWARE

GESTIÓN DE CONFIGURACIÓN DE SOFTWARE

Se refiere a las prácticas y herramientas transversales al desarrollo de software (a los requerimientos o el diseño en sí del programa), para atacar incumbencias como la:

Trazabilidad

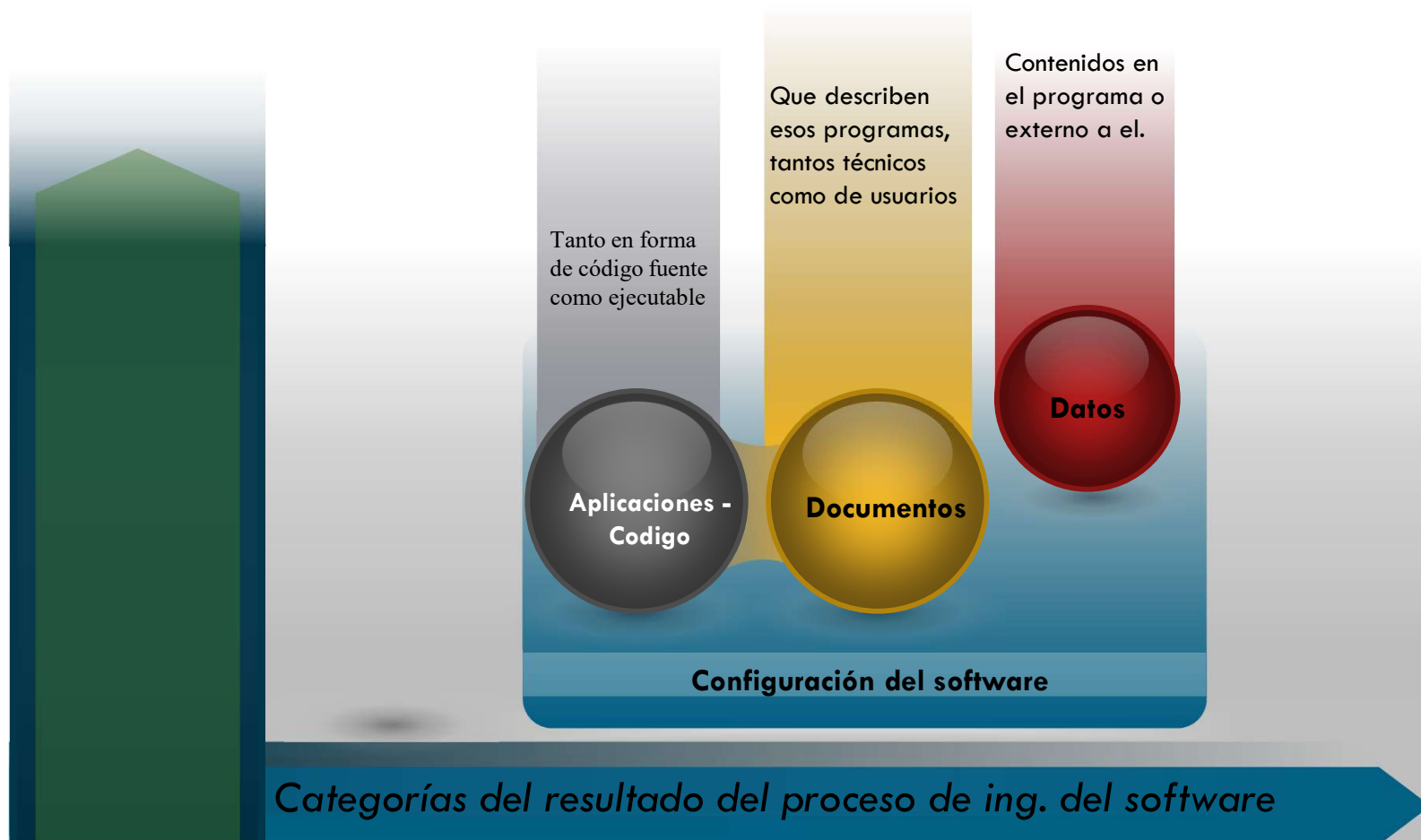
- ✓ Gestión de Versiones
- ✓ Gestión de Cambios
- ✓ Gestión de Requerimientos
- ✓ Gestión de Incidencias.

Reproducibilidad de releases

Comunicación

Interacción, coordinación e integración del trabajo de los diferentes miembros del equipo

Proceso de Gestión de Configuración de Software



SISTEMA DE VERSIONADO DE CÓDIGO

Un Sistema de Versionado de Código (SVC) nos permite **compartir el código** fuente de nuestros desarrollos y a la vez **mantener un registro de los cambios** por los que va pasando.

En general va a ser la herramienta más importante y fundamental dentro del desarrollo. Veremos que también vamos a tener otras herramientas para publicar releases, documentación, para integración y control de calidad, pero todo va a depender de la herramienta SVC.

Si se pierden los releases, más allá de ser un inconveniente, se puede salvar en base al SVC.

SISTEMA DE VERSIONADO DE CÓDIGO

Se basan en versionar archivos y carpetas (repositorio)

Los desarrolladores utilizan un programa cliente que permite una serie de operaciones básicas:

- **obtener una copia** local de repositorio
- **publicar cambios** a un repositorio
- **crear ramas**

Otras funciones:

- revisar los cambios
- obtener una copia de una **revisión** en particular
- deshacer cambios.
- marcar (taggear) una revisión
- ver quién, por qué y cuándo modifico cierto archivo/carpeta

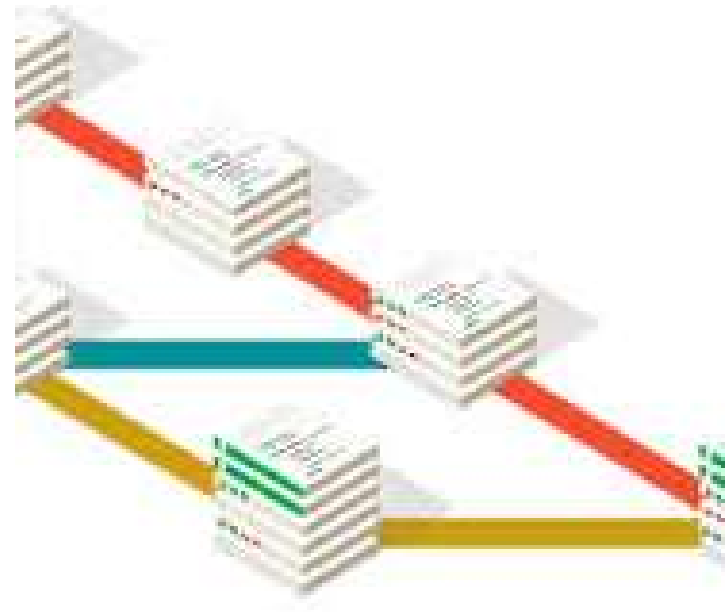
Estas operaciones dependen un poco del tipo de SCV que estemos utilizando.

TIPOS DE VERSIONADORES DE CÓDIGO

Sistemas Centralizados:

Son los más "tradicionales", por ejemplo SVN, CVS, etc.

Sistemas Distribuidos (o descentralizados): son los que están en auge actualmente como: Git, Mercurial, Bazaar, etc.



PARA RECORDAR!!!

- ✓ *Elementos claves en el desarrollo:
Personas, procesos, tecnologías*
- ✓ *Metodologías de desarrollo:*
 - *Metodologías Estructuradas*
 - *RUP (Rational Unified Process)*
 - *Desarrollo orientado a Objetos*
 - *Métodos Agiles*
 - *Reingeniería del Software*
- ✓ *Roles en el equipo=RESPONSABILIDAD*
- ✓ *Gestión de la configuración*
 - *Trazabilidad*
 - *Gestión de Versiones*
 - *Gestión de Cambios*
 - *Gestión de Requerimientos*
 - *Gestión de Incidencias*

