Principios de Arquitectura Empresarial

Fabio Castro Rozo

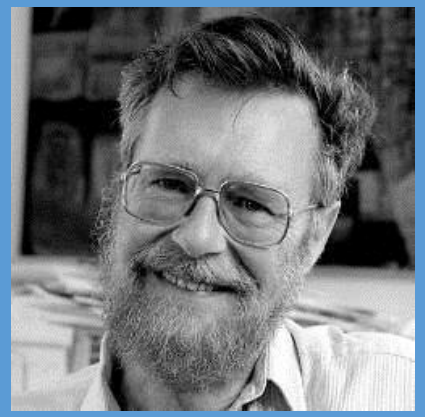
Contenido

1. Arquitectura de Software

Arquitectura de Software

Orígenes

- **1968**



Edsger Dijkstra

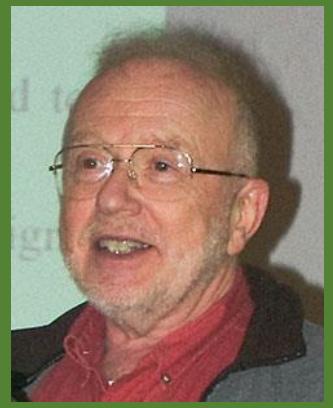
Vale la pena preocuparse por cómo se divide y estructura el software, en lugar de simplemente programarlo para producir un resultado correcto

Dijkstra estaba escribiendo sobre un sistema operativo, y primero planteó la noción de una estructura en capas, en la que los programas se agrupaban en capas, y los programas en una capa solo podían comunicarse con los programas en capas adyacentes. Dijkstra señaló la elegante integridad conceptual exhibida por dicha organización, con las ganancias resultantes en la facilidad de desarrollo y mantenimiento.

Arquitectura de Software

Orígenes

- **1972**



David Parmas

Contribuciones sobre módulos de ocultación de información [Parnas 72], estructuras de software [Parnas 74] y familias de programas [Parnas 76].

En **programación orientada a objetos** el principio de ocultación hace referencia a que los atributos privados de un objeto no pueden ser modificados ni obtenidos a no ser que se haga a través del paso de un mensaje (invocación a métodos, ya sean estos funciones o procedimientos).
En **informática**, se conoce como principio de ocultación de información a la ocultación de decisiones de diseño en un programa susceptible de cambios con la idea de proteger a otras partes del código si éstos se producen. Proteger una decisión de diseño supone proporcionar una interfaz estable que proteja el resto del programa de la implementación (susceptible de cambios). En los lenguajes de programación modernos el principio de ocultación de información se manifiesta de diferentes maneras, como por ejemplo la encapsulación.

Arquitectura de Software

Orígenes

- **1975**

Frederick Brooks

Arquitectura del Sistema: La especificación completa y detallada de la interfaz del usuario

No hay balas de plata: Los proyectos de software habituales pueden transformarse en un monstruo (hombre lobo) de plazos no alcanzados, presupuestos sobrepasados, y productos con errores. Entonces se escuchan gritos desesperados pidiendo una bala de plata–algo que haga disminuir los costos del software tan rápidamente como bajan los costos del hardware.

Ley de Brooks: "añadir más efectivos a un proyecto de software en retraso, lo retrasará más“. "Nueve mujeres no pueden tener un bebé en un mes".

Arquitectura de Software

Orígenes

- **1992**

Dewayne E. Perry & Alexander L. Wolf

Son los primeros que proponen un modelo para la arquitectura de software; este modelo contempla a la arquitectura formada por tres componentes: elementos, forma y razón

Los elementos pueden ser de procesamiento, datos o conexión; la forma se define de acuerdo a las propiedades de, y a las relaciones entre los elementos; la razón se contempla en términos de restricciones del sistema, que se derivan de los requerimientos del sistema.

“la década de los noventas, creemos, será la década de la arquitectura de software”

http://users.ece.utexas.edu/~perry/work/papers/swa-sen.pdf

Arquitectura de Software

Orígenes

- **1996**

Mary Shaw

"Software Architecture: Perspectives on an Emerging Discipline“

Arquitecturas para sistemas de software, así como mejores formas de apoyar el desarrollo de software.

reunir las abstracciones útiles del diseño de sistemas y las anotaciones y herramientas del desarrollador de software, y observar los patrones utilizados para la organización del sistema ... para ilustrar la disciplina y examinar las formas en que el diseño arquitectónico puede afectar el diseño de software. Nuestra selección enfatiza las descripciones informales, tocando ligeramente las anotaciones y especificaciones formales y las herramientas para apoyarlas.

Arquitectura de Software

Orígenes

- **90s**

Mary Shaw

"Software Architecture: Perspectives on an Emerging Discipline“

Arquitecturas para sistemas de software, así como mejores formas de apoyar el desarrollo de software.

reunir las abstracciones útiles del diseño de sistemas y las anotaciones y herramientas del desarrollador de software, y observar los patrones utilizados para la organización del sistema ... para ilustrar la disciplina y examinar las formas en que el diseño arquitectónico puede afectar el diseño de software. Nuestra selección enfatiza las descripciones informales, tocando ligeramente las anotaciones y especificaciones formales y las herramientas para apoyarlas.

Arquitectura de Software

Orígenes

SG **Arquitectura de Software.**

https://sg.com.mx/content/view/409

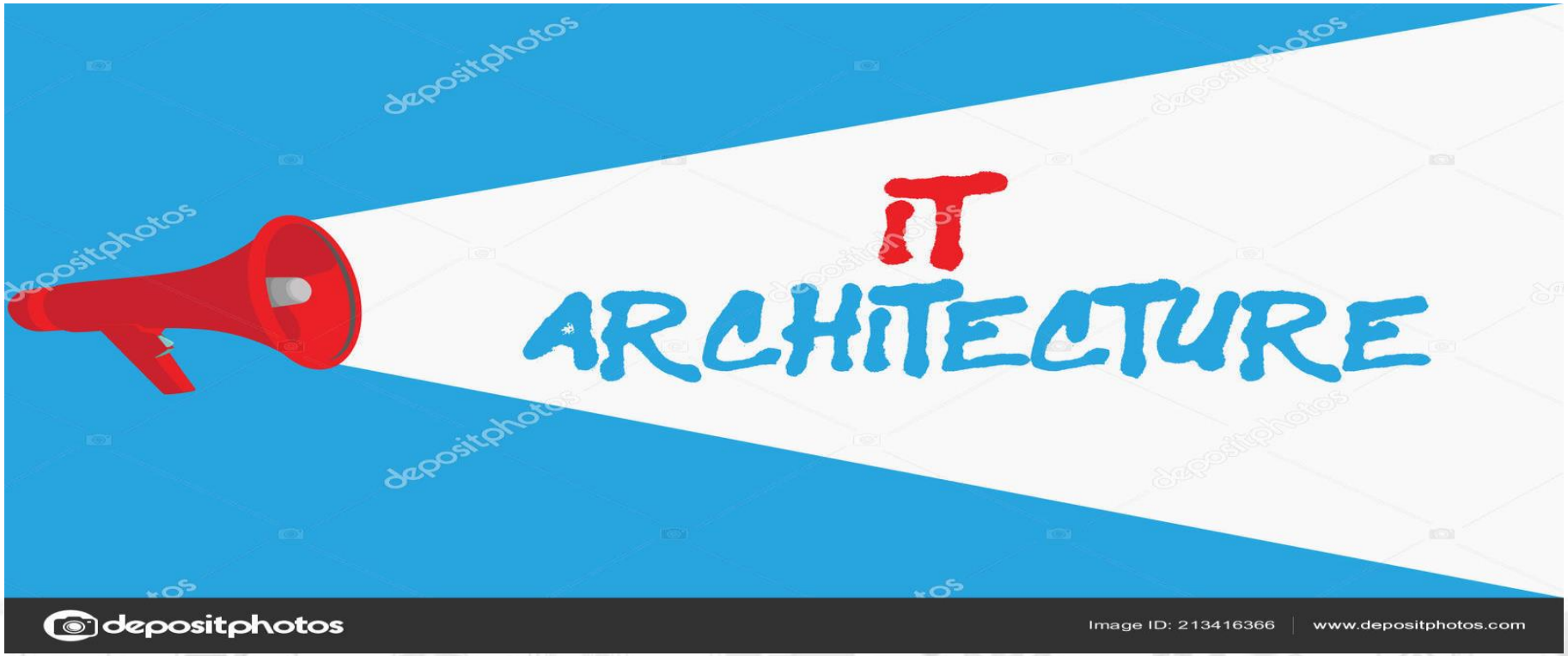
Arquitectura de Software

Sean grandes o pequeños, simples o complejos, todos los sistemas informáticos están formados por las mismas tres partes fundamentales:

- Hardware (por ejemplo, procesadores, memoria, discos, tarjetas de red);
- Software (por ejemplo, programas o bibliotecas); y
- Datos, que pueden ser transitorios (en memoria) o persistentes (en disco o ROM).

Arquitectura de Software

Cuando intenta entender un sistema informático, le interesa entender qué hacen realmente sus partes individuales, cómo trabajan juntas y cómo interactúan con el mundo que las rodea…….



Arquitectura de Software

Definición:

La arquitectura de un sistema intensivo en software es la estructura o estructuras del sistema, la cual comprende los elementos de software, las propiedades externas visibles de esos elementos y las relaciones entre ellos.

Software Architecture in Practice (2nd edition), Bass, Clements, Kazman; AddisonWesley 2003

https://jegadeesansite.files.wordpress.com/2018/01/sei-series-in-software-engineering-len-bass-paul-clements-rick-kazman-software-architecture-in-practice-addison-wesley-professional-2012.pdf

What is your definition of software architecture?

https://resources.sei.cmu.edu/asset_files/FactSheet/2010_010_001_513810.pdf

Arquitectura de Software

¿Por qué es importante?

La manera en que se estructura un sistema

Ciclo de vida del Software

¿Qué es Software?

Definición Clásica

1) Instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, función y desempeño buscados;
2) Estructuras de datos que permiten que los programas manipulen en forma adecuada la información, y
3) Información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

Roger S. Pressman. Ingeniería del Software. Un enfoque práctico

Ciclo de vida del Software

SDLC Software Development Lifecycle

Es un proceso sistemático para crear software que garantiza la calidad y la corrección del software creado. El proceso SDLC tiene como objetivo producir software de alta calidad que cumpla con las expectativas del cliente. El desarrollo del software debe completarse en el marco de tiempo y costo predefinidos.

SDLC consiste en un plan detallado que explica cómo planificar, construir y mantener software específico. Cada fase del ciclo de vida de SDLC tiene su propio proceso y entregables que alimentan la siguiente fase

Ciclo de vida del Software

¿Por qué SDLC?

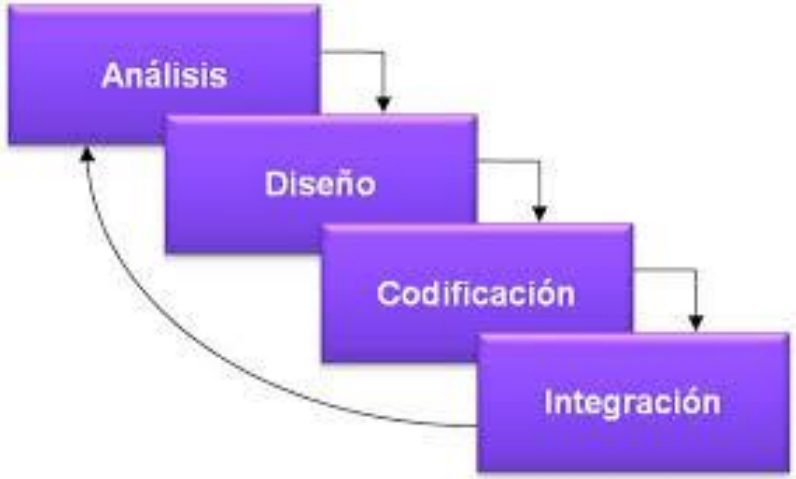
1. Ofrece una base para la planificación, programación y estimación de proyectos.
2. Proporciona un marco para un conjunto estándar de actividades y entregables.
3. Es un mecanismo para el seguimiento y control de proyectos.
4. Aumenta la visibilidad de la planificación del proyecto para todas las partes interesadas involucradas en el proceso de desarrollo.
5. Aumento y mejora de la velocidad de desarrollo.
6. Relaciones mejoradas con el cliente
7. Le ayuda a disminuir el riesgo del proyecto y la sobrecarga del plan de gestión del proyecto

Ciclo de vida del Software

Fases del SDLC

Arquitectura de Software en el SDLC

Fases del SDLC



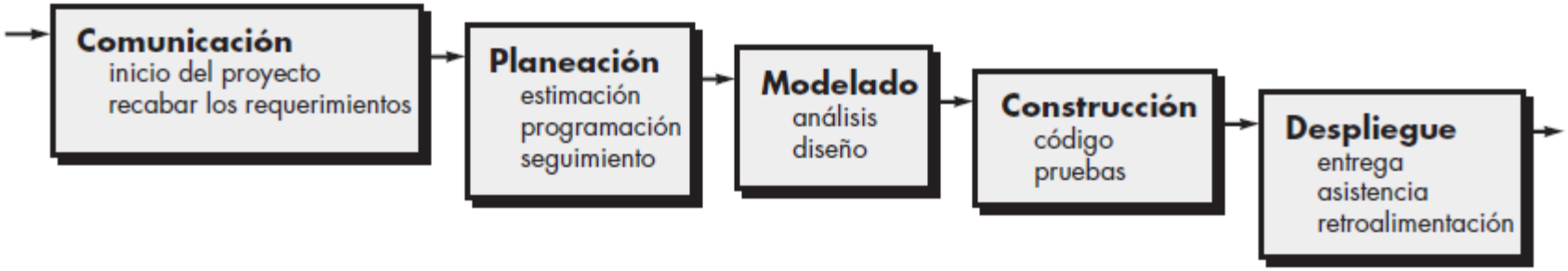
Arquitectura de Software en el SDLC

Las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las **metodologías** de desarrollo

Arquitectura de Software en el SDLC

Metodologías

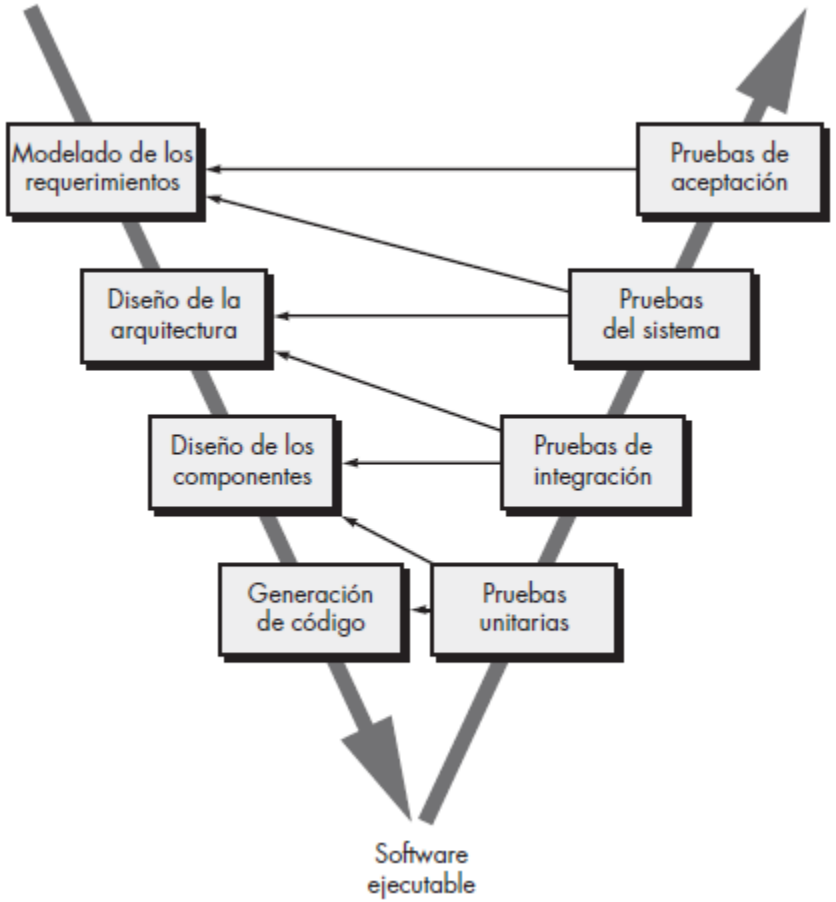
Cascada



Arquitectura de Software en el SDLC

Metodologías

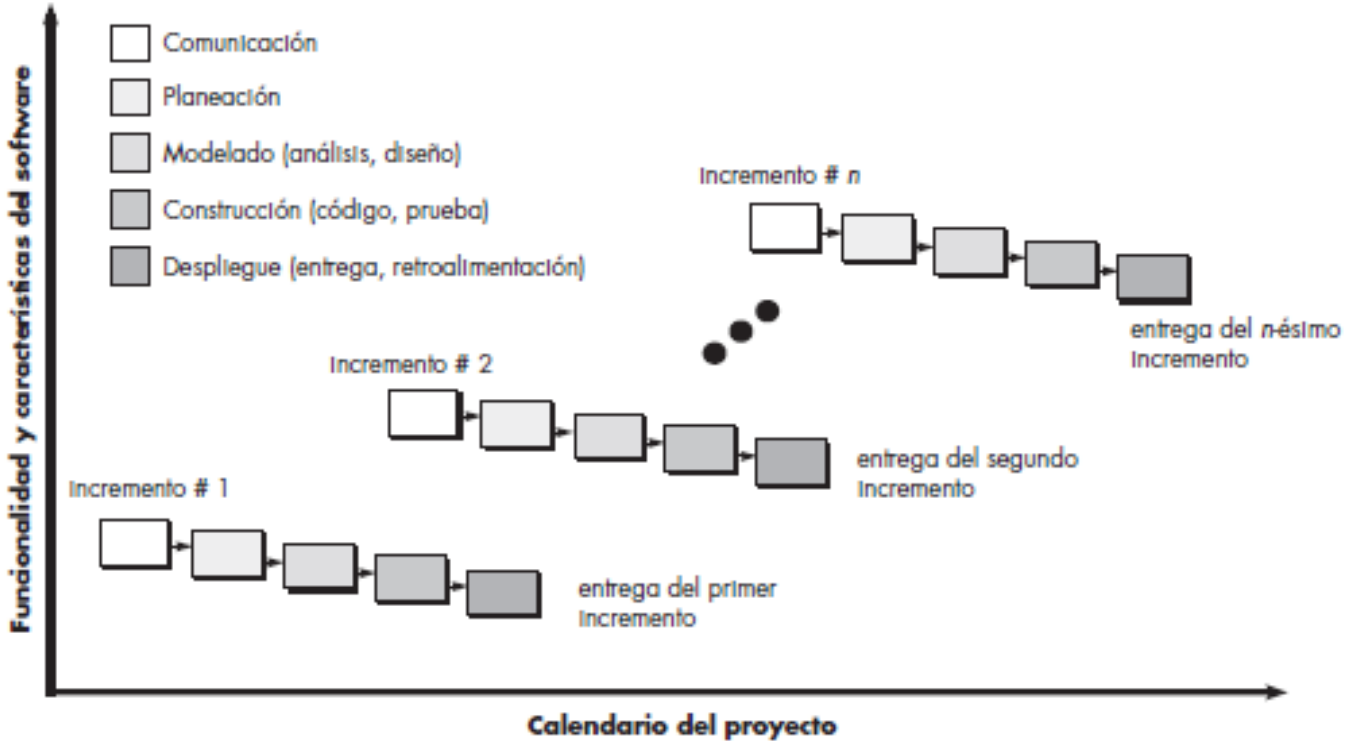
- V



Arquitectura de Software en el SDLC

Metodologías

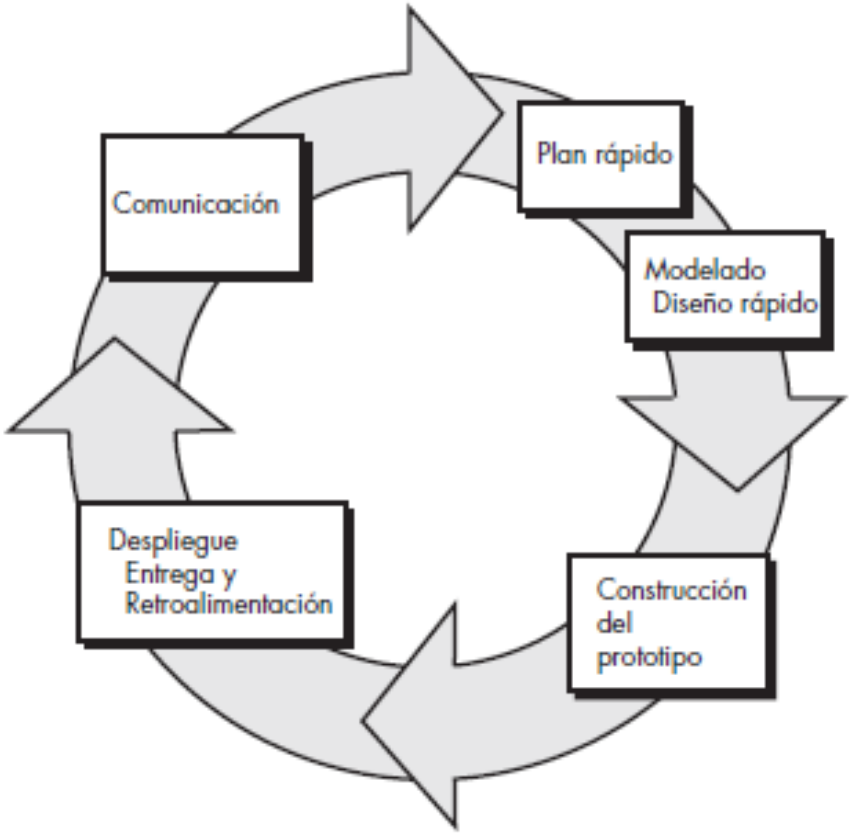
Incremental



Arquitectura de Software en el SDLC

Metodologías

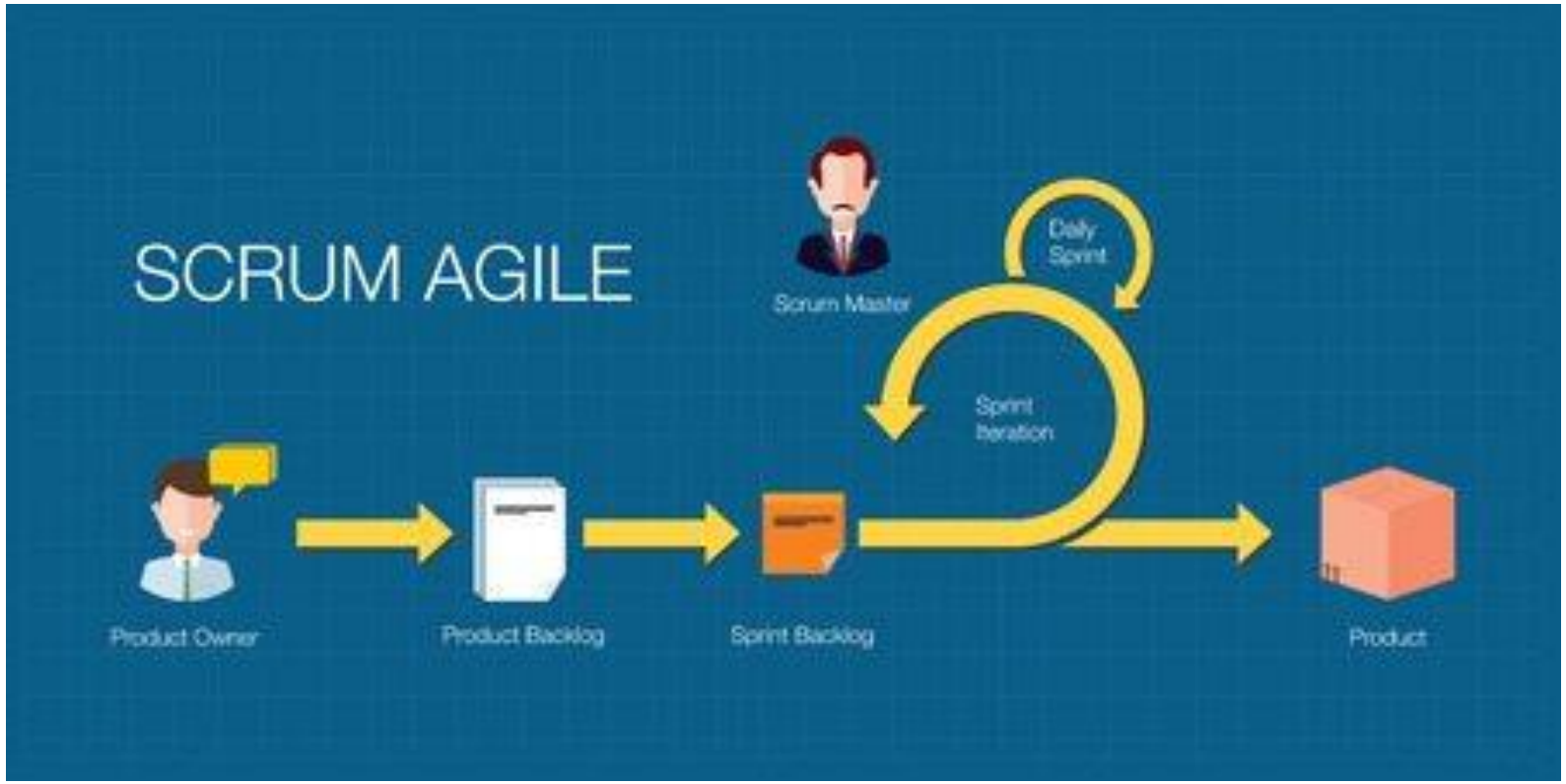
Prototipo



Arquitectura de Software en el SDLC

Metodologías

Agile



Arquitectura de Software en el SDLC

¿Cómo se involucra en las Fases?

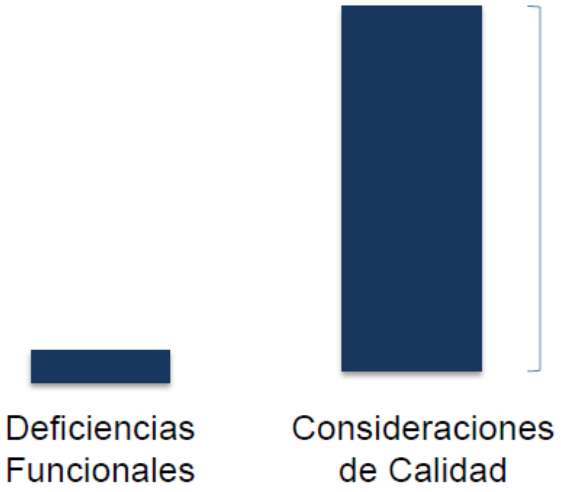
Requerimientos: La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura → **Atributos de Calidad**

Diseño: Etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, tácticas de diseño y elecciones tecnológicas

Evaluación: Es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos.

Arquitectura de Software y Calidad

¿Por qué se rediseñan los sistemas?



- Dificultades de mantenimiento
- Falta de portabilidad
- Problemas de escalabilidad
- Problemas de performance
- Problemas de seguridad
- Interfaces gráficas obsoletas
- Etc.

Las **consideraciones de calidad** se desprenden de las necesidades del negocio y deben jugar un **rol fundamental** durante el ciclo de vida del desarrollo de software. **No todo es funcionalidad en un sistema.**

Arquitectura de Software y Calidad

Funcionalidad y Atributos de Calidad

La funcionalidad es la capacidad de un sistema para realizar la tarea para el cual fue implementado

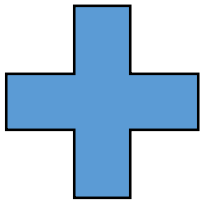
Un diseño puede cumplir con la funcionalidad deseada y fallar a la hora de satisfacer sus requerimientos de calidad.

Los sistemas se descomponen en componentes (módulos) a fin de satisfacer objetivos adicionales a la funcionalidad.

Sin embargo, existe cierto trade-offs(compromiso) entre funcionalidad y atributos de calidad.

Arquitectura de Software y Calidad

¿Qué es la Calidad en el Software?



producto que funciona

calidad

cumplimiento de condiciones de entrega (plazo y presupuesto).

Satisfacción del usuario

Arquitectura de Software y Calidad

Pero …. ¿Qué es la Calidad en el Software?

"Los factores de un producto de software que contribuyen a la satisfacción completa y total de las necesidades de un usuario u organización".

Arquitectura de Software y Calidad

Pero entonces ¿Cuándo un software tiene calidad?

Se puede decir que el software tiene calidad si cumple o excede las expectativas del usuario en cuanto a:

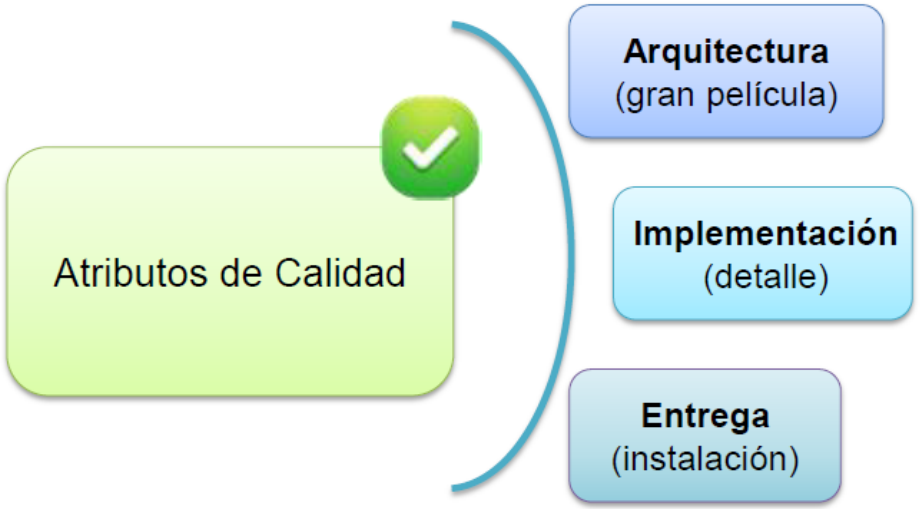
1. Funcionalidad (que sirva un propósito),
2. Ejecución (que sea práctico),
3. Confiabilidad (que haga lo que debe),
4. Disponibilidad (que funcione bajo cualquier circunstancia) y
5. Apoyo, a un costo menor o igual al que el usuario está dispuesto a pagar.

Arquitectura de Software y Calidad

Atributos de Calidad

Los atributos de calidad deben ser considerados a lo largo de todo el ciclo de vida del software para tener éxito:

- No dependen solamente de la etapa de diseño (y/o arquitectura)
- No dependen solamente de la implementación o entrega



Arquitectura de Software y Calidad

Atributos de Calidad

- Disponibilidad
- Performance
- Flexibilidad
- Interoperabilidad
- Mantenibilidad
- Portabilidad
- Reusabilidad
- Robustez
- Testeabilidad
- Usabilidad
- Integridad
- Confiabilidad