
‹Mision
TIC 2022›

# Agenda

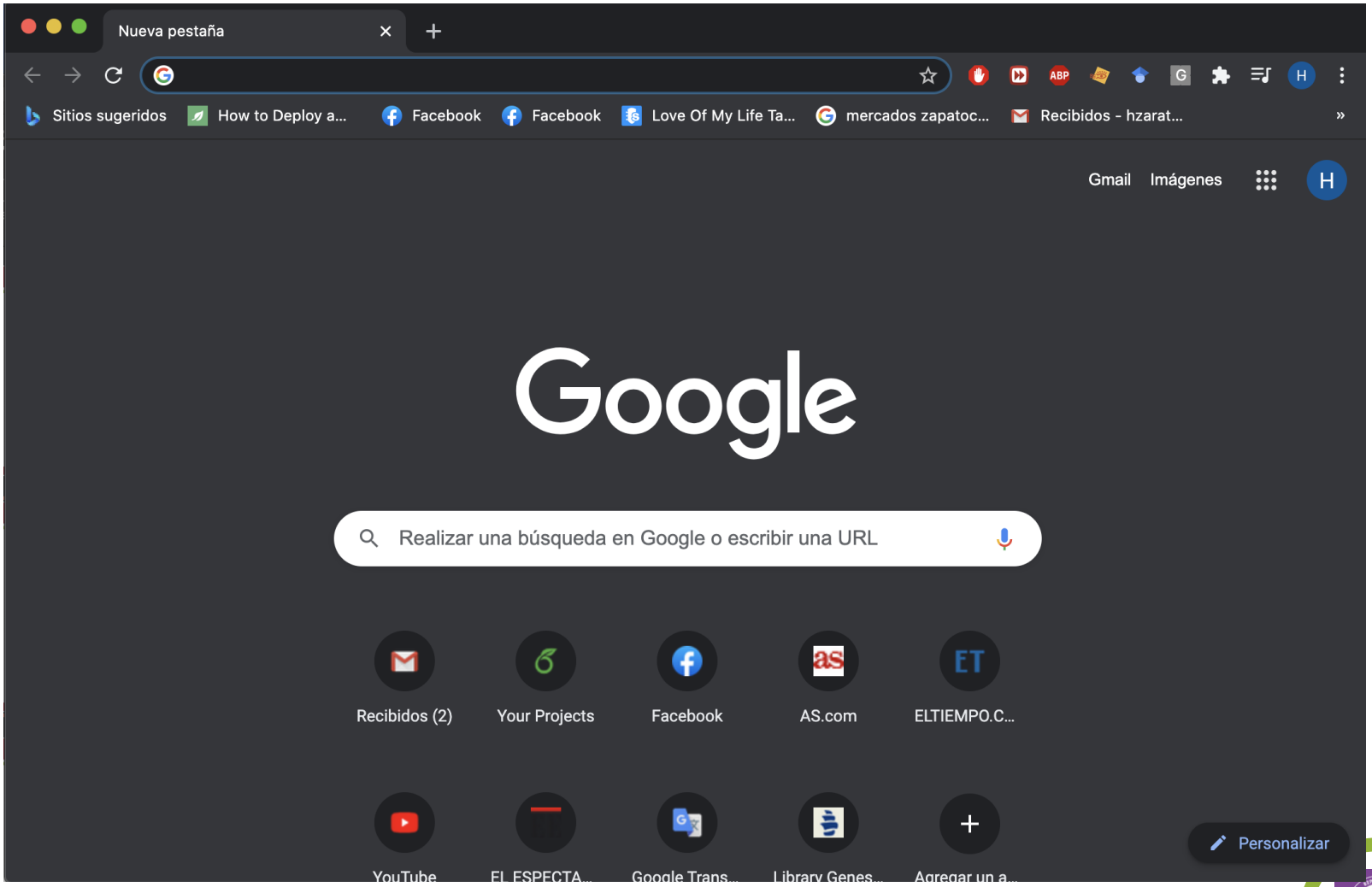
1. ¿Qué es Desarrollar software?
2. ¿Cuál es el ciclo de Vida del Software?
3. Metodologías de Desarrollo.

# 1. ¿Qué es Desarrollar Software?

# Desarrollo de Software

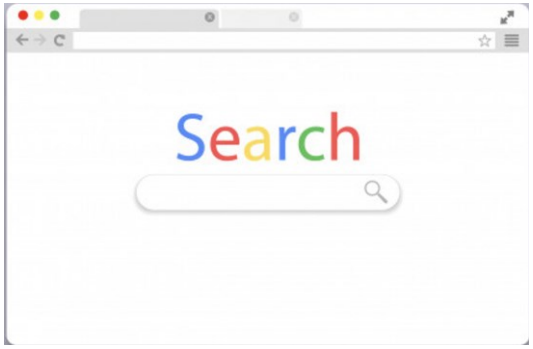
# Herramientas

# Ejemplo


Nueva pestaña
Sitios sugeridos
How to Deploy a...
Facebook
Facebook
Love Of My Life Ta...
mercados zapatoc...
Recibidos - hzarat...
Gmail
Imágenes
H
Google
Realizar una búsqueda en Google o escribir una URL
Recibidos (2)
Your Projects
Facebook
AS.com
ELTIEMPO.C...
YouTube
Google Trans...
Library Genes...
Agregar un a...
Personalizar

# ¿Cómo funciona un Navegador?

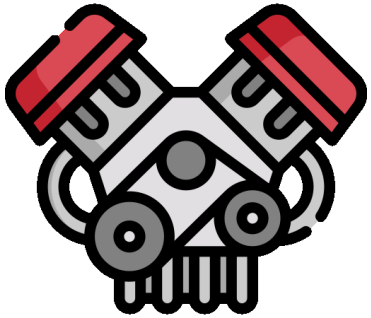
1



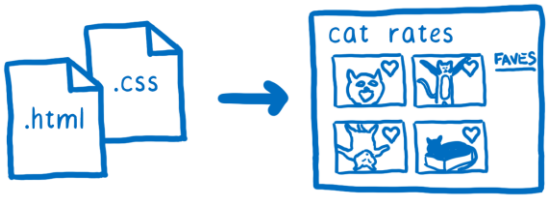

**Interfaz de Usuario**

**Usuario**

2



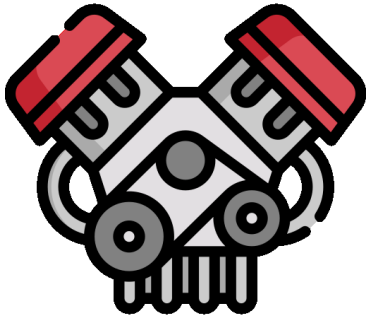
**Motor de Renderizado**




**“Dibuja” sobre el navegador con base en los archivos de entrada**

# ¿Cómo funciona un Navegador?

3

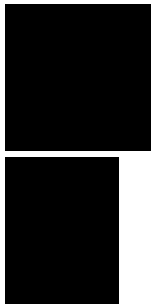


**Motor de búsqueda**

**Localiza los servidores en la Red**

4

**Red**

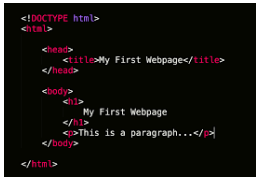



**Cargar los archivos y contenidos en el navegador**

# ¿Cómo funciona un Navegador?

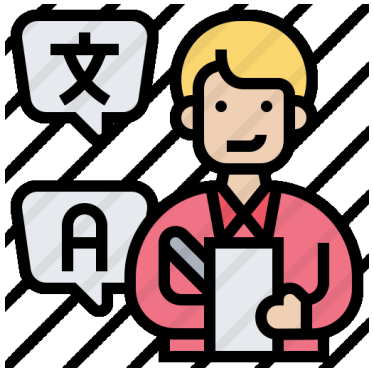
7

Lenguaje de Programación



```
<!DOCTYPE html>
<html>

    <head>
        <title>My First Webpage</title>
    </head>

    <body>
        <h1>
            My First Webpage
        </h1>
        <p>This is a paragraph...</p>
    </body>

</html>
```

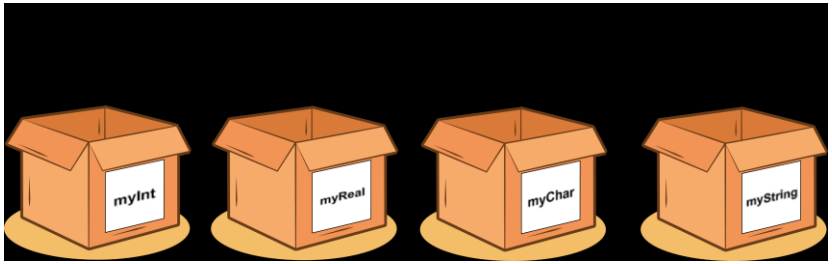
```
.hero-image {
  padding-top: 250px;
  padding-bottom: 225px;
  background-image: url("public/img_hero.jpg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100%;
  position: relative;
}

.hero-text {
  text-align: center;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  color: #4F4D53;
  font: 'Roboto';
  font-size: 50px;
}
```



Interprete

6

Almacenamiento



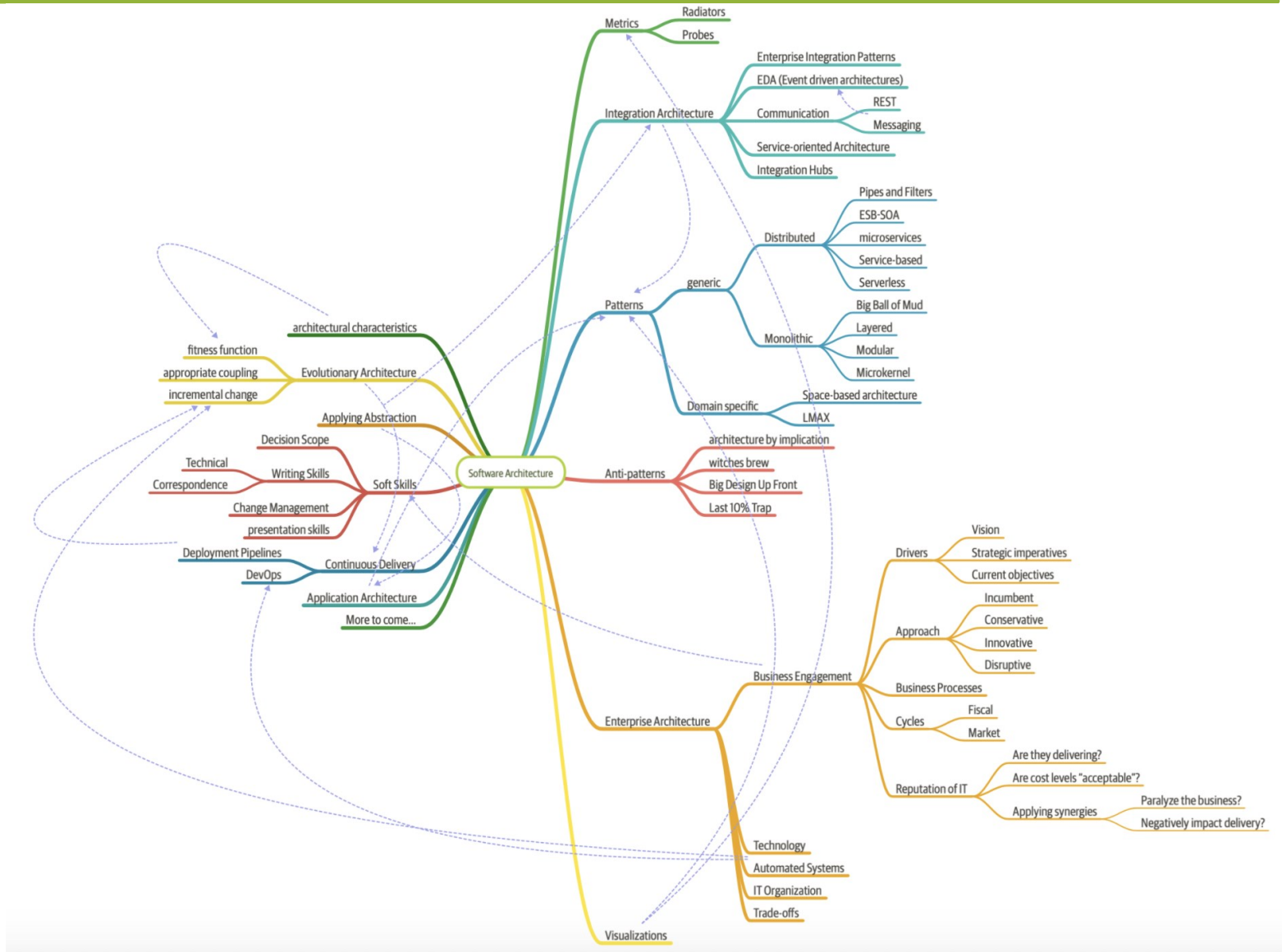

Variables

Cookies

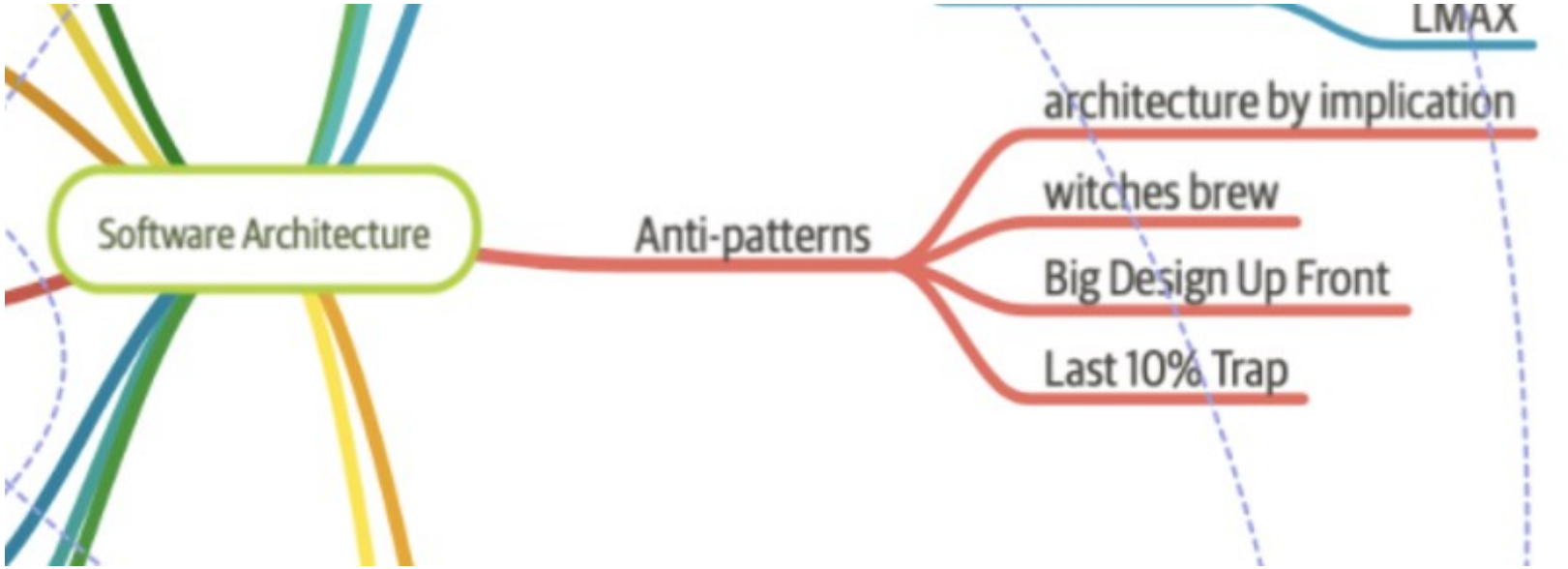
# Arquitectura de Software

"La arquitectura de software consiste en tomar decisiones estructurales fundamentales que son costosas de cambiar una vez implementadas"

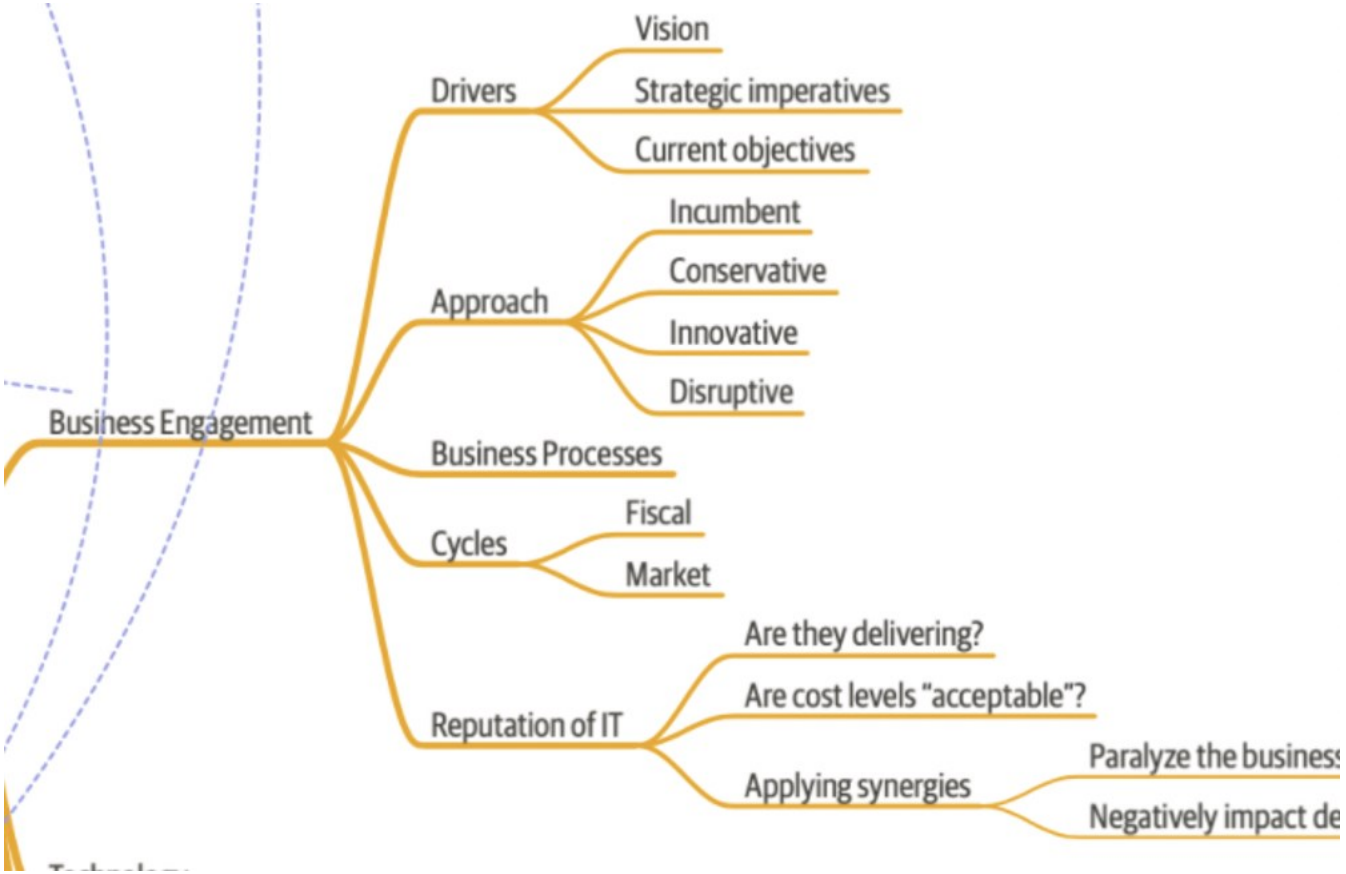
# Arquitectura de Software



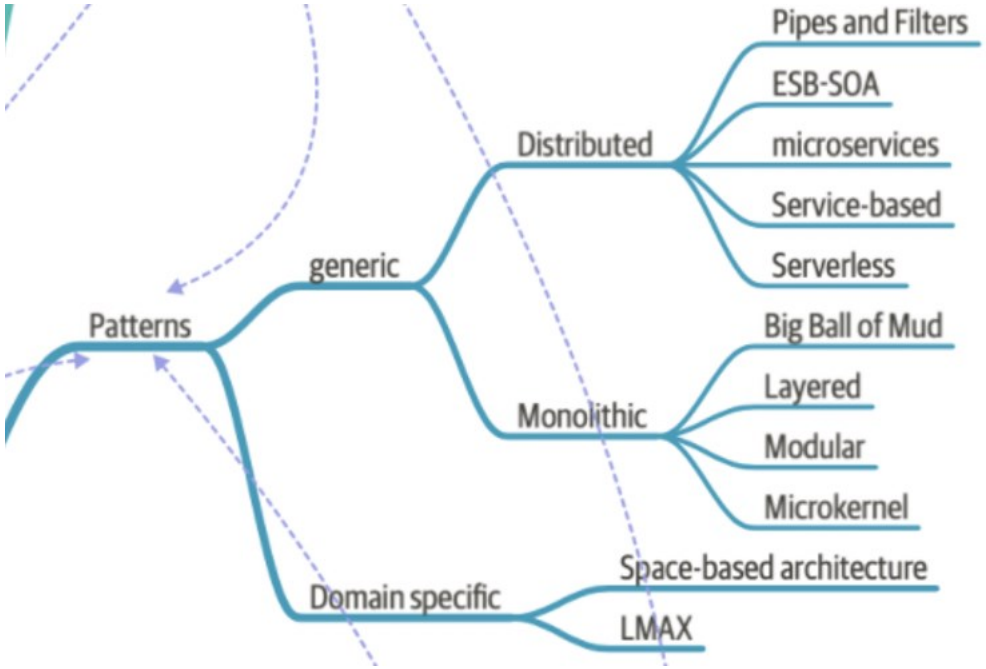
# Arquitectura de Software


Software Architecture
Anti-patterns
LMAX
architecture by implication
witches brew
Big Design Up Front
Last 10% Trap

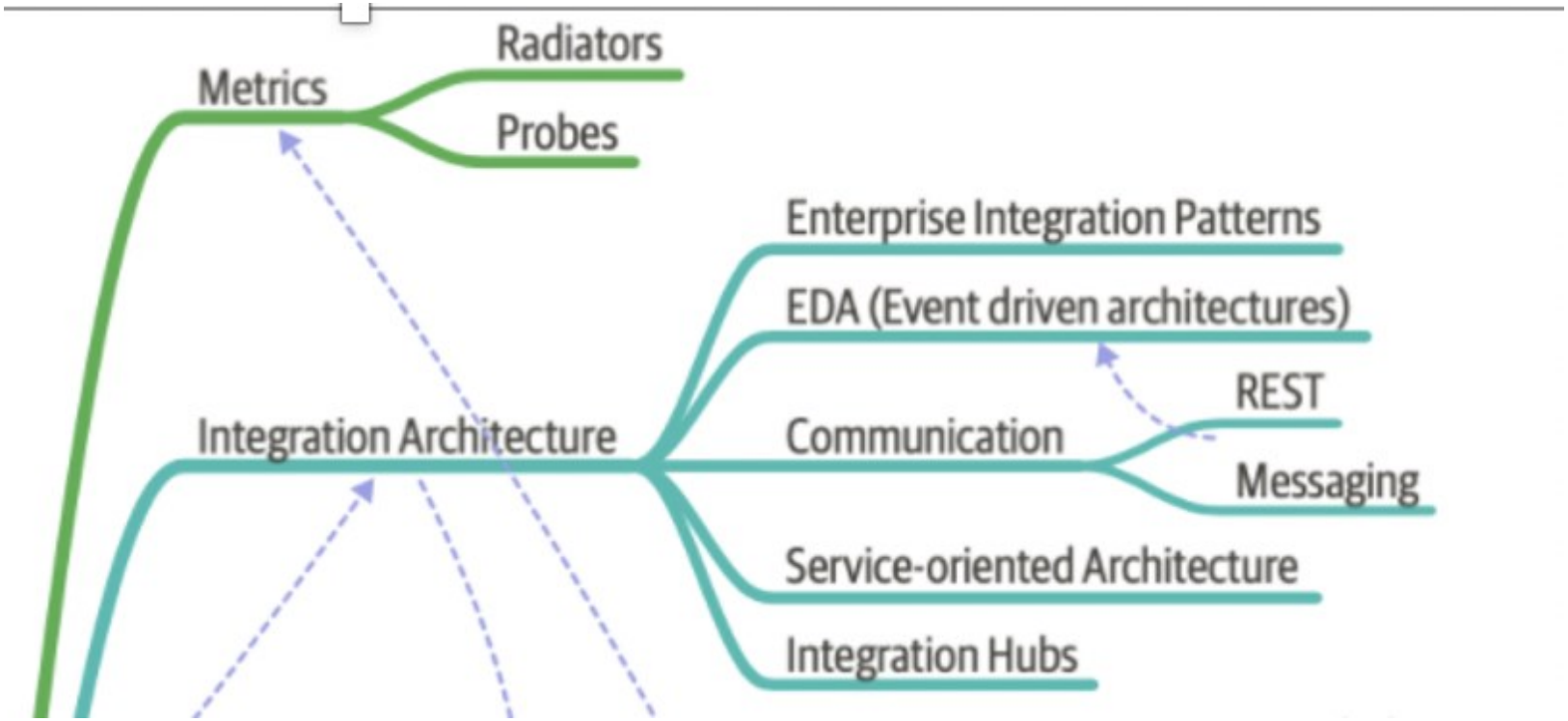
# Arquitectura de Software



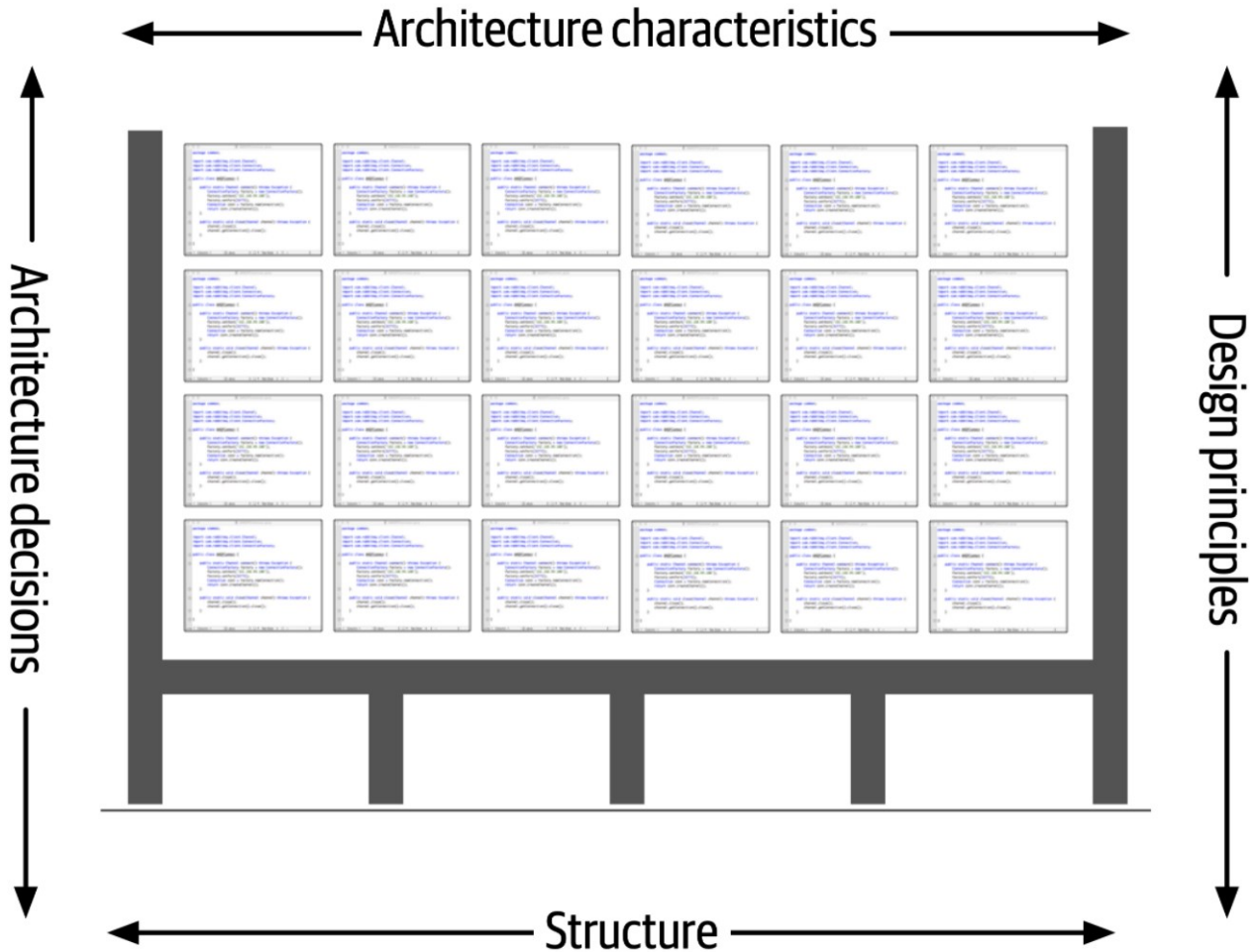
# Arquitectura de Software



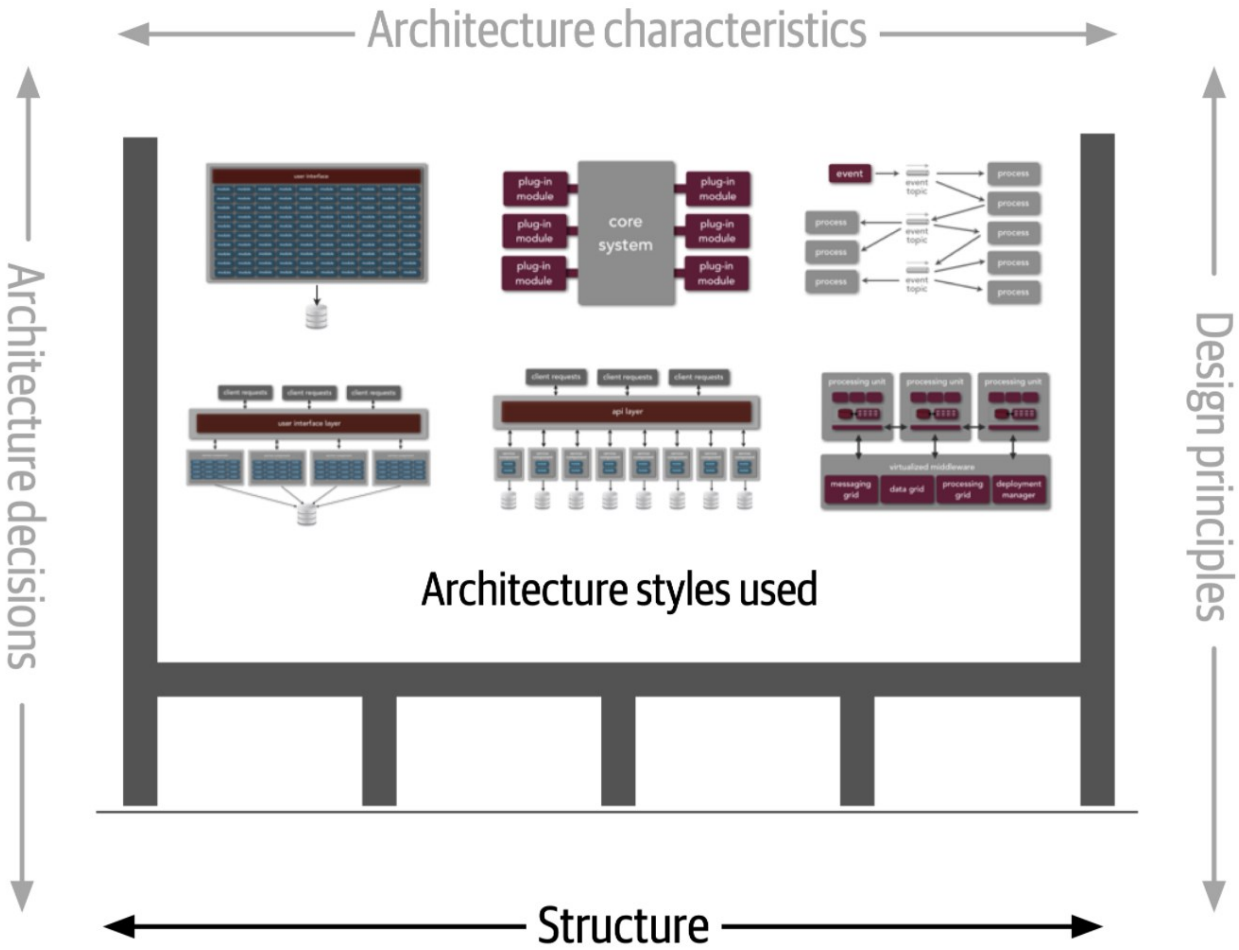
# Arquitectura de Software



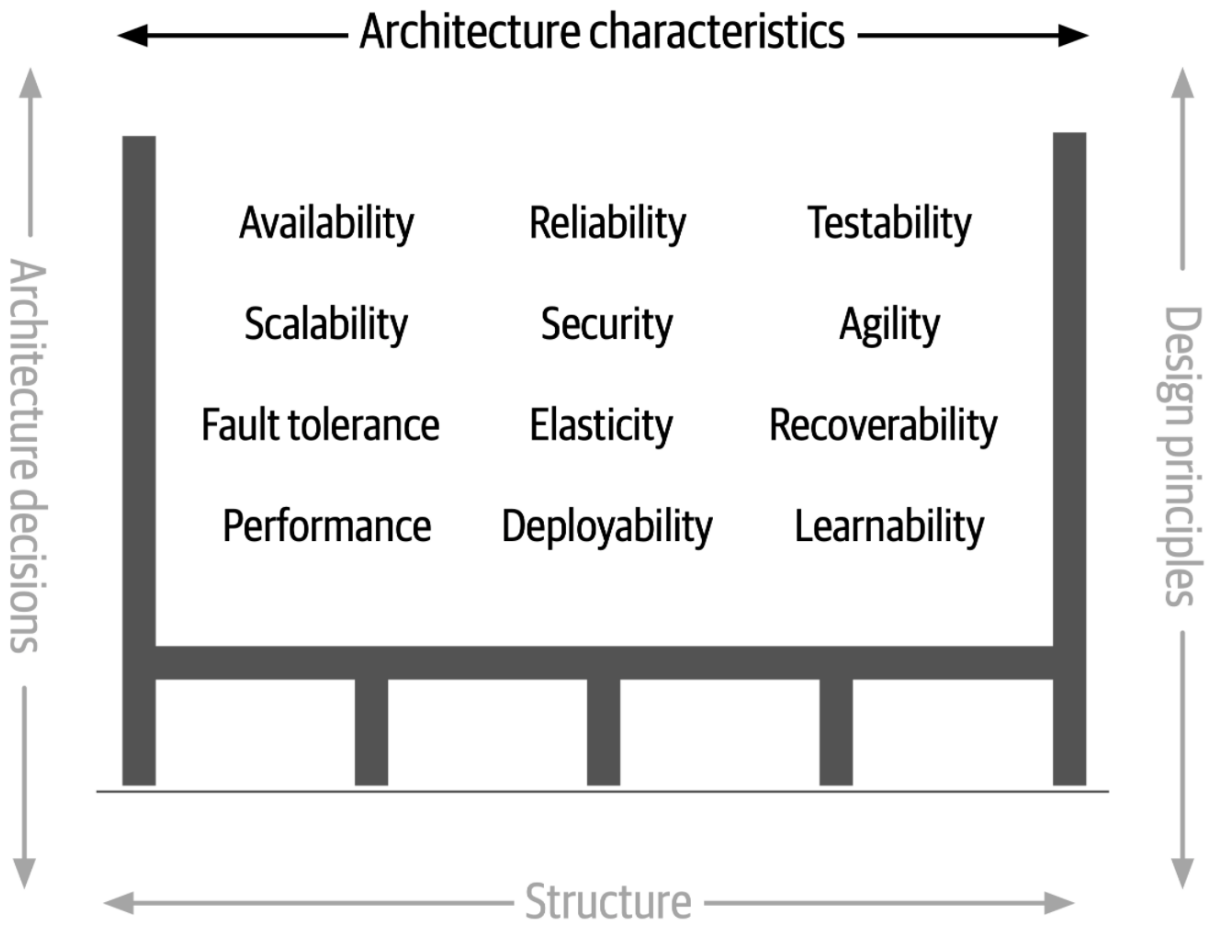
# Conceptos Básicos



# Conceptos Básicos



# Conceptos Básicos



# Características de Operación Tradicionales

**Disponibilidad**

- Cuato tiempo se requiere que este disponible el sistema(si es 24/7, se debe definir la forma en la cual en caso de falla este disponible rapidamente el sistema).

**Continuidad**

- Capacidad de Recuperación de Desastre

**Rendimiento**

- Incluye pruebas de estrés, análisis de picos, análisis de la frecuencia de funciones utilizadas, capacidad requerida y tiempos de respuesta. La aceptación del desempeño a veces requiere un ejercicio propio, que demora meses en completarse.

**Recuperación**

- Requerimientos de continuidad de NegocioPor ejemplo, en caso de desastre, ¿con qué rapidez se requiere que el sistema vuelva a estar en línea?). Esto afectará la estrategia de respaldo y los requisitos para hardware duplicado.

**Confiabilidad/ Seguridad**

- Evalúe si el sistema debe ser a prueba de fallas o si es de misión crítica de una manera que afecte vidas. Si fracasa, ¿le costará a la empresa grandes sumas de dinero?

**Robuztes**

- Capacidad para manejar errores y condiciones de límite mientras se ejecuta si la conexión a Internet se cae o si hay un corte de energía o falla de hardware.

**Escalabilidad**

- Capacidad del sistema para funcionar y funcionar a medida que aumenta el número de usuarios o solicitudes .

# Características Estructurales

Configurabilidad

- Capacidad de los usuarios finales para cambiar fácilmente aspectos de la configuración del software (a través de interfaces utilizables).

Extensibilidad

- Qué importante es incorporar nuevas funciones.

Instalabilidad

- Facilidad de instalación del sistema en todas las plataformas necesarias

Reutilización

- Capacidad para aprovechar componentes comunes en múltiples productos.

# Características Estructurales

## Localización

- Support for multiple languages on entry/query screens in data fields; on reports, multibyte character requirements and units of measure or currencies.

## Mantenibilidad

- ¿Qué tan fácil es aplicar cambios y mejorar el sistema?

## Portabilidad

- ¿El sistema necesita ejecutarse en más de una plataforma? (Por ejemplo, ¿el frontend debe ejecutarse tanto en Oracle como en SAP DB?

## Compatibilidad

- ¿Qué nivel de soporte técnico necesita la aplicación? ¿Qué nivel de registro y otras facilidades se requieren para depurar errores en el sistema?

## Actualización

- Capacidad para actualizar fácil / rápidamente desde una versión anterior de esta aplicación / solución a una versión más nueva en servidores y clientes.

# Características Transversales de la arquitectura

**Accessibilidad**

- Acceso a todos sus usuarios, incluidos aquellos con discapacidades como daltonismo o pérdida auditiva.

**Archivabilidad**

- ¿Será necesario archivar o eliminar los datos después de un período de tiempo? (Por ejemplo, las cuentas de los clientes deben eliminarse después de tres meses o marcarse como obsoletas y archivarse en una base de datos secundaria para acceso futuro).

**Autenticación**

- Requisitos de seguridad para garantizar que los usuarios sean quienes dicen ser.

**Autorización**

- Requisitos de seguridad para garantizar que los usuarios solo puedan acceder a determinadas funciones dentro de la aplicación (por caso de uso, subsistema, página web, regla comercial, nivel de campo, etc.).

**Leyes**

- ¿En qué restricciones legislativas funciona el sistema (protección de datos, Sarbanes Oxley, GDPR, etc.)? ¿Qué derechos de reserva requiere la empresa? ¿Alguna normativa sobre la forma en que se creará o desplegará la aplicación?

**Privacidad**

- Capacidad para ocultar transacciones a los empleados internos de la empresa (transacciones cifradas para que ni siquiera los administradores de bases de datos y los arquitectos de redes puedan verlas).

**Seguridad**

- Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?

**Compatibilidad**

- What level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system?

**Usabilidad**

- Level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue."

# 2. ¿Cuál es el ciclo de Vida del Software?

# Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) es la sigla en inglés de Tiempo de Vida del Desarrollo de Software , como proceso usado para diseñar, desarrollar , probar y validar la calidad de un software. También es llamado Proceso de Desarrollo de Software.

Esta definido por el estándar ISO/IEC 12207 y existen diversas metodologías para validar el ciclo de vida del proceso de Desarrollo de Software.

# Ciclo de Vida del Sofware

## 1 Requisitos

**Ingeniería de Software**

Casos de Uso

Historias de Usuario

Elementos No Funcionales

**Cliente**

Necesidades

Objetivo Funcional

## 2 Diseño

**Arquitectonico**

Estructuras
Propiedades

**Detallado**

Detallado

# Ciclo de Vida del Sofware

3 **Desarrollo**

Metodologías
Herramientas (Lenguajes , Framework)
Conocimientos de programación

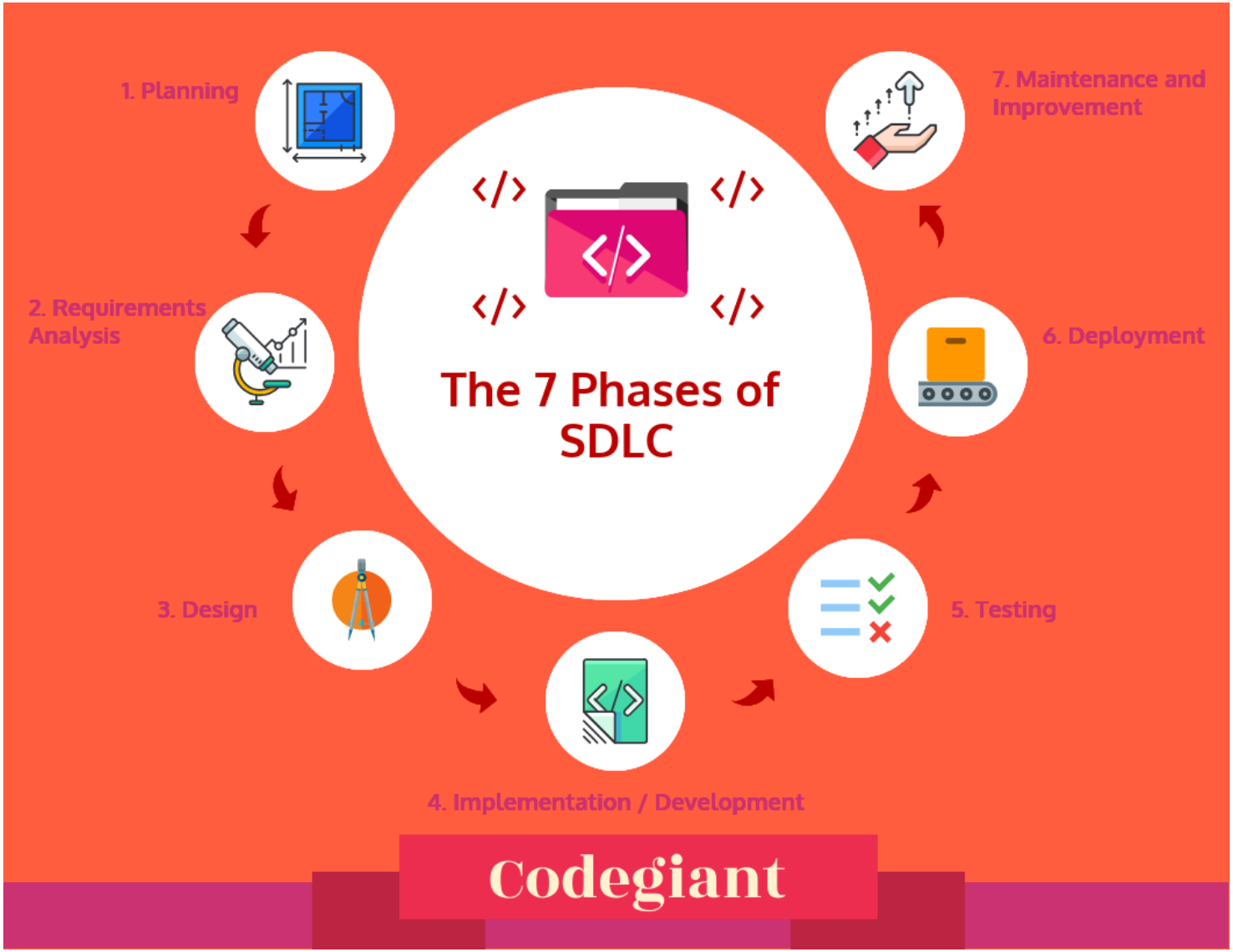
4 **Pruebas**

Validar
Verificar

5 **Mantenimiento y Evolución**

Salida a Producción

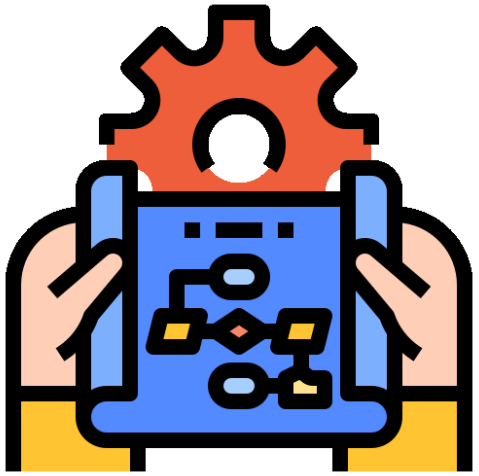
# Etapas en el SDLC



# 1. Planeación

En esta fase, se está realizando una investigación exhaustiva sobre el producto que planea desarrollar. Luego, está discutiendo sus planes con los clientes y las partes interesadas.

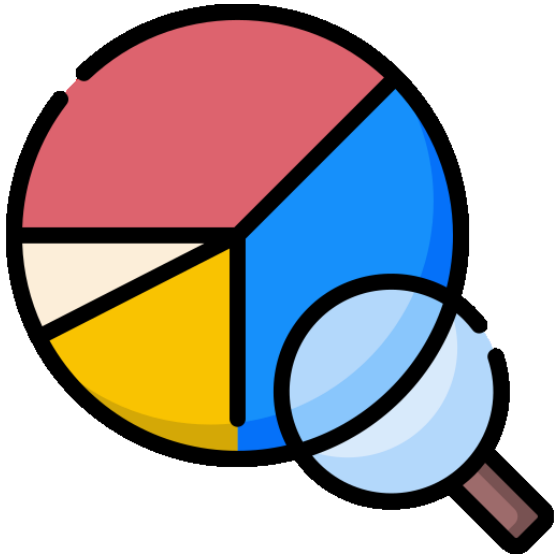
También está identificando los pros y los contras de los métodos de software actuales que está utilizando. Por lo tanto, puede duplicar los pros y reducir los contras al mínimo



# 2. Análisis de Requerimientos

- Una vez finalizada la investigación, puede continuar con la creación de undocumento SRS (Especificación de requisitos de software). En este documento, describirá todas las características del producto.

- Luego, presentará el documento SRS a las partes interesadas para que puedan aceptarlo o rechazarlo. Depende del tiempo y las limitaciones financieras.



# 3. Diseño.

Una vez que se completa el documento de SRS, su equipo, específicamente los arquitectos del producto, creará otro documento: la DDS (Especificación de documento de diseño).

En el DDS, tendrá sus funciones descritas detalladamente. Dentro del documento, también tendrá el presupuesto y las estimaciones de tiempo necesarias para que el producto se complete con éxito. Básicamente, tendrá todo lo que sus desarrolladores necesitan para comenzar a trabajar en el producto real.

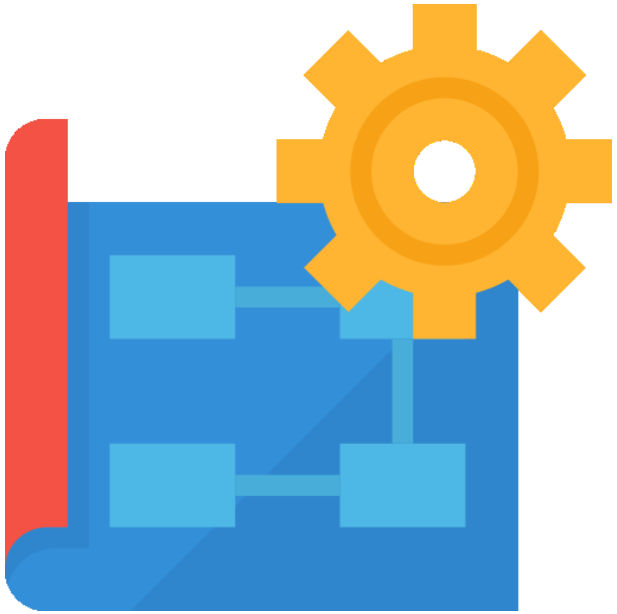
Pero antes de eso, el DDS debe ser aprobado por el cliente y las partes interesadas. A veces, se requieren cambios debido a varias razones que van desde estimaciones de tiempo y presupuesto hasta la solidez del software.

# 4. Implementación y Desarrollo

La fase de implementación en SDLC generalmente toma el período más largo, ya que implica el desarrollo real del producto. Sus desarrolladores trabajarán en la creación de un producto basado en el DDS. Además, dependiendo de la solidez de DDS, los desarrolladores codificarán sin mucha molestia o tendrá problemas en el camino.

También deben seleccionar el lenguaje de codificación más apropiado para el tipo de software que está creando.

Es vital tener en cuenta que la comunicación entre su equipo en esta fase debe ser eficaz y precisa. Esto se debe a que sus desarrolladores deberán comunicarse con los evaluadores de QA (garantía de calidad), los gerentes de productos y proyectos. Esto les ayudará a desarrollar un producto que sus clientes realmente disfruten.



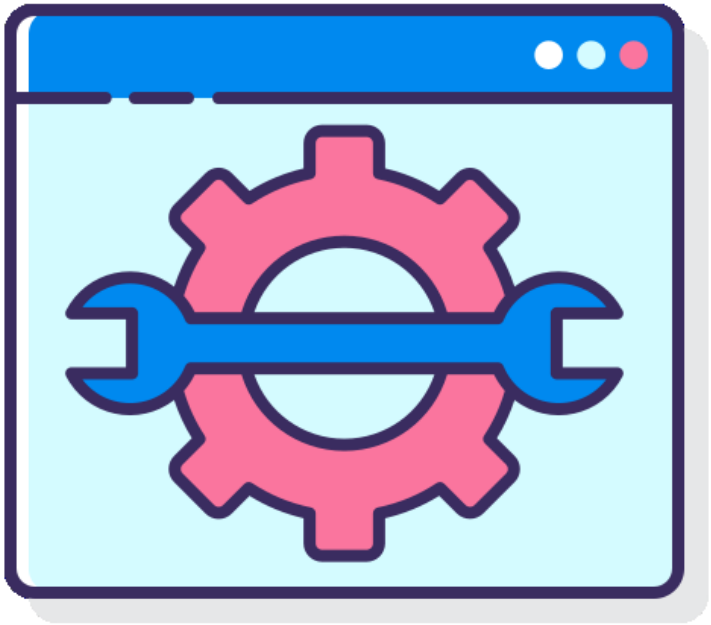
# 5. Pruebas

Una vez que se desarrolla el producto, sigue la fase de prueba del ciclo de vida del desarrollo de software . Aquí, los probadores de control de calidad tienen que pasar por el código base para encontrar errores y errores.

Si se informan problemas, el producto se devuelve a los desarrolladores para que rectifiquen las fallas y lo implementen nuevamente. Esta fase se repite hasta que el producto se vuelve impecable.

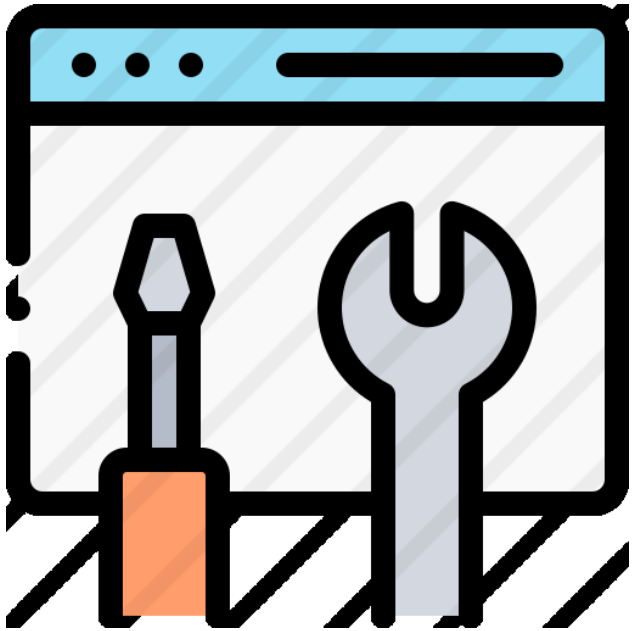
# 6. Despliegue

Once all the errors are removed, the product is rolled out to the market automatically.



# 7. Mantenimiento y Mejoras

Después de la implementación, debe observar cómo reacciona el mercado a su producto. Luego, en función de los comentarios que recibe, crea informes sobre lo que debe mejorarse.



*"¿Necesita actualizar la versión del software? ¿Necesitas agregar más funciones? ¿Necesita hacer la interfaz más simple e intuitiva?"*

# 3. Metodologías de Desarrollo

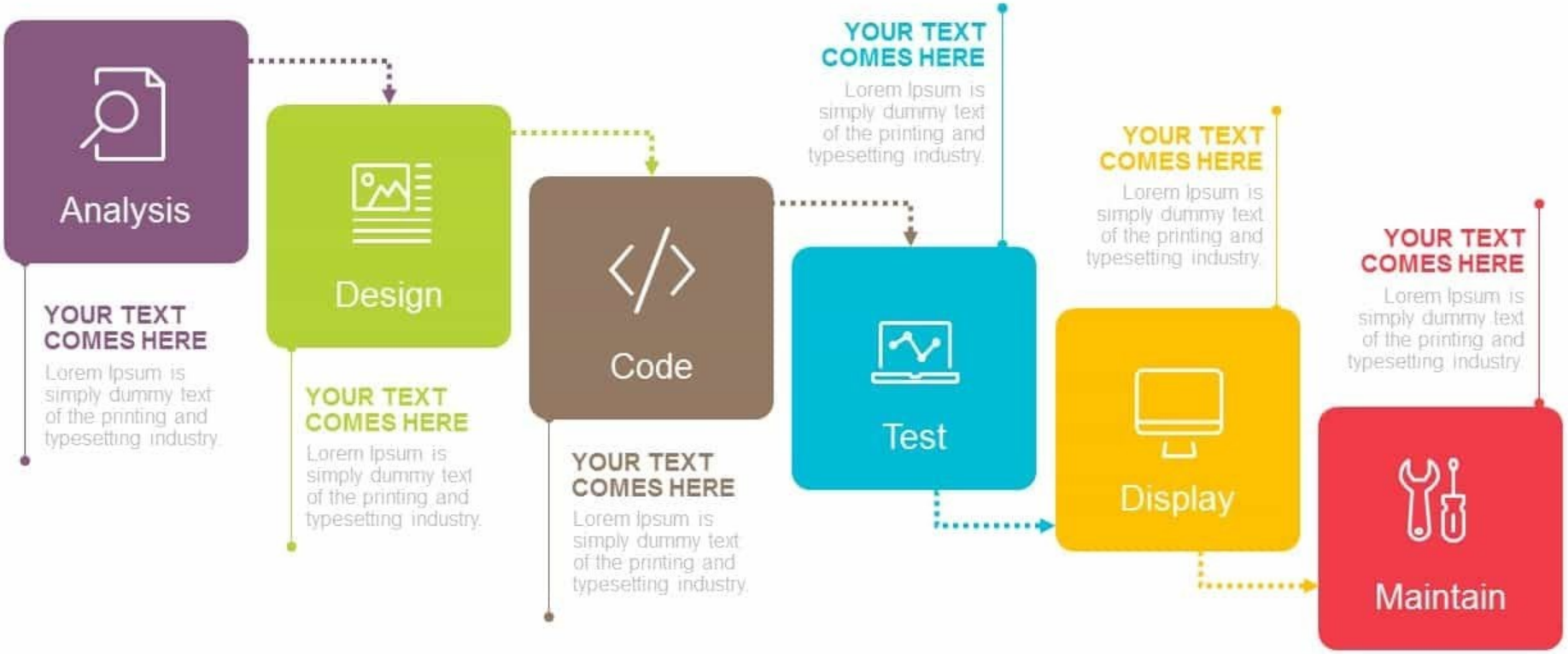
# Metodologías

- Cascada  (Waterfall model).
- El modelo V ( V model).
- Modelo iterativo (Iterative model.)
- Modelo Espiral (Spiral model).
- Modelos Agiles (The Agile model).
- Metodología Scrum (Scrum methodology).
- Metodología De Programación Extrema(The Extreme Programming methodology).
- El modelo RAD (The Rad Model).
- Modelo de Prototipos de Software (The Software Prototype model).
- Modelo Bing Bang (The Big Bang model).

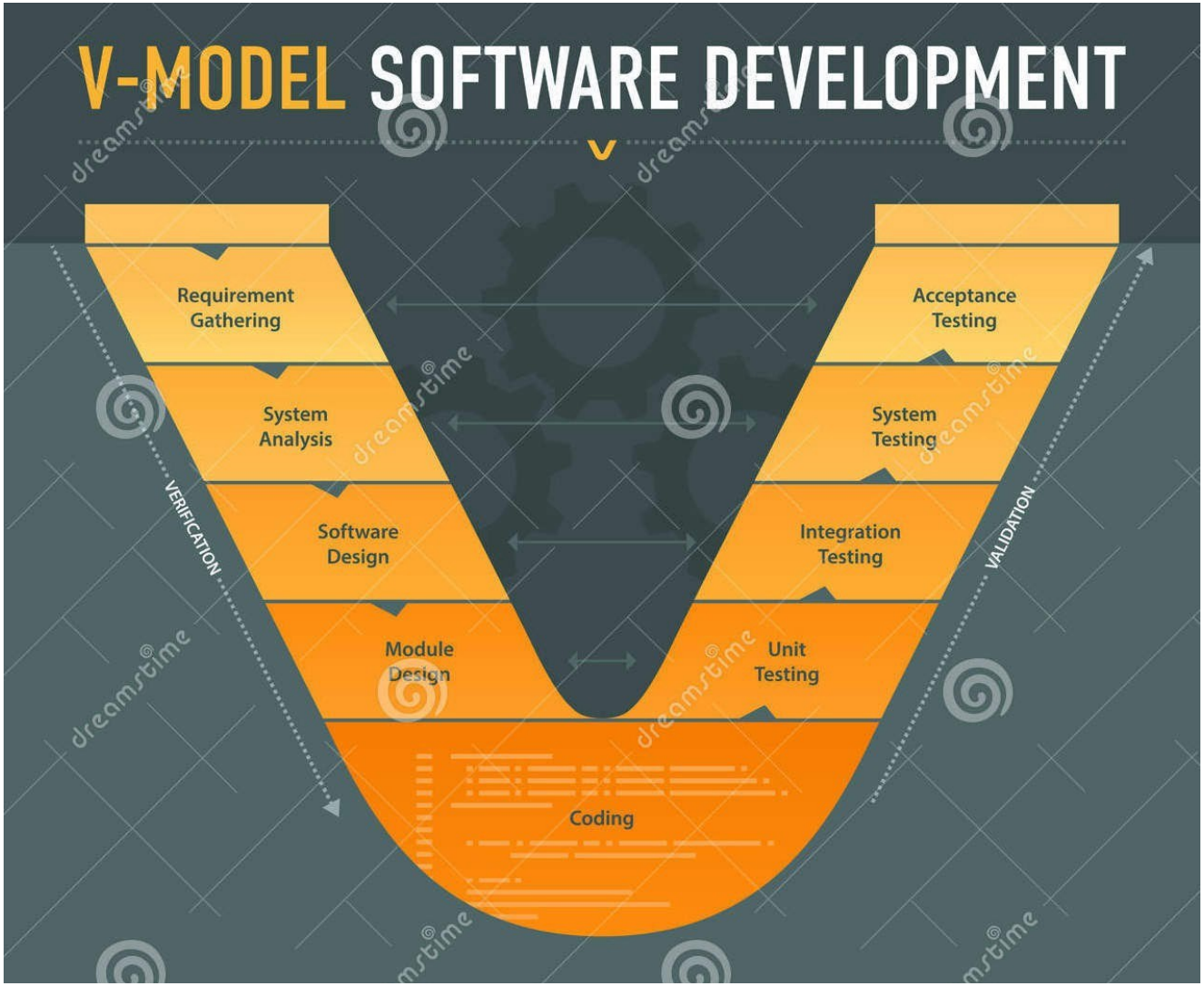
# Modelo Cascada

Model

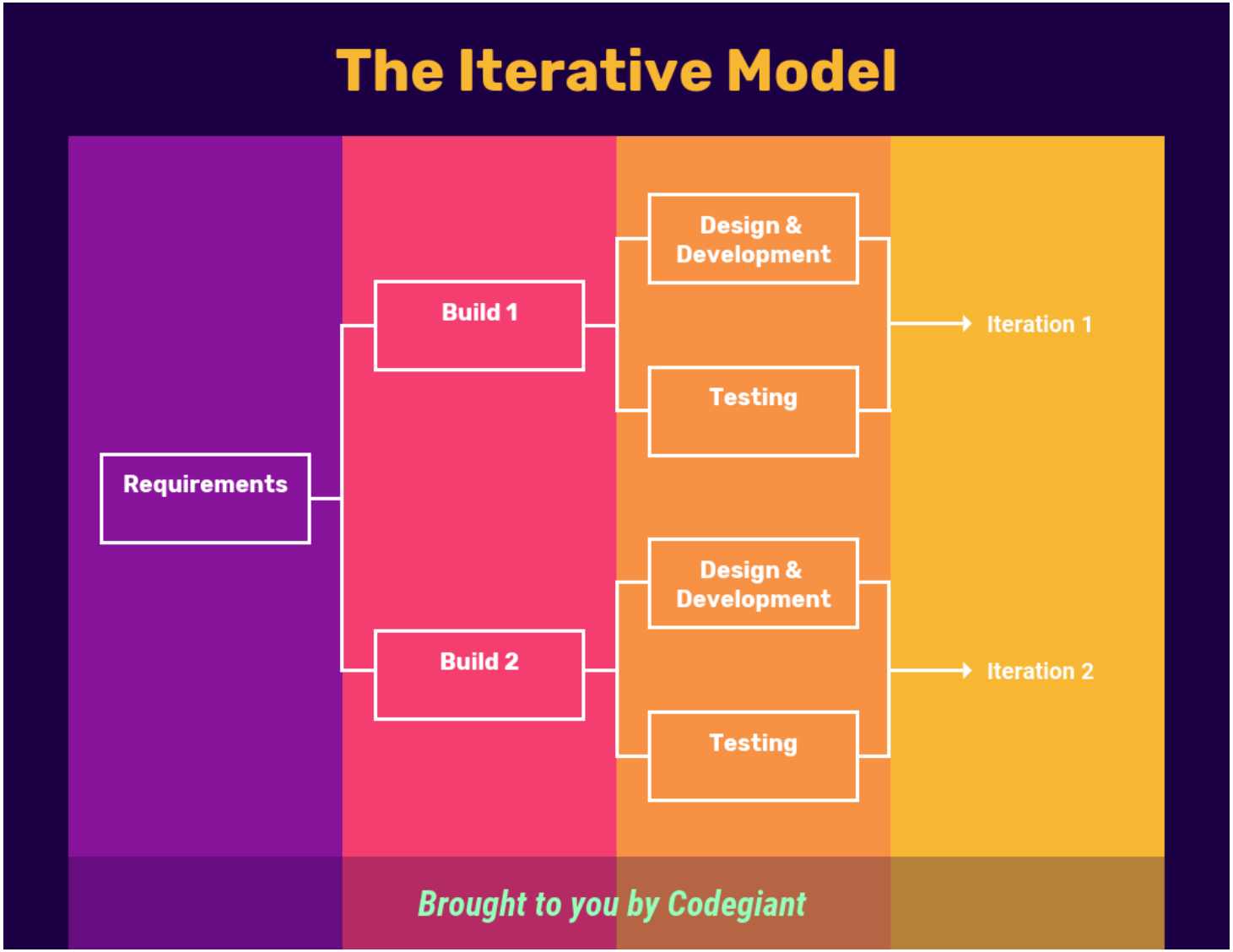
## SOFTWARE WATERFALL MODEL



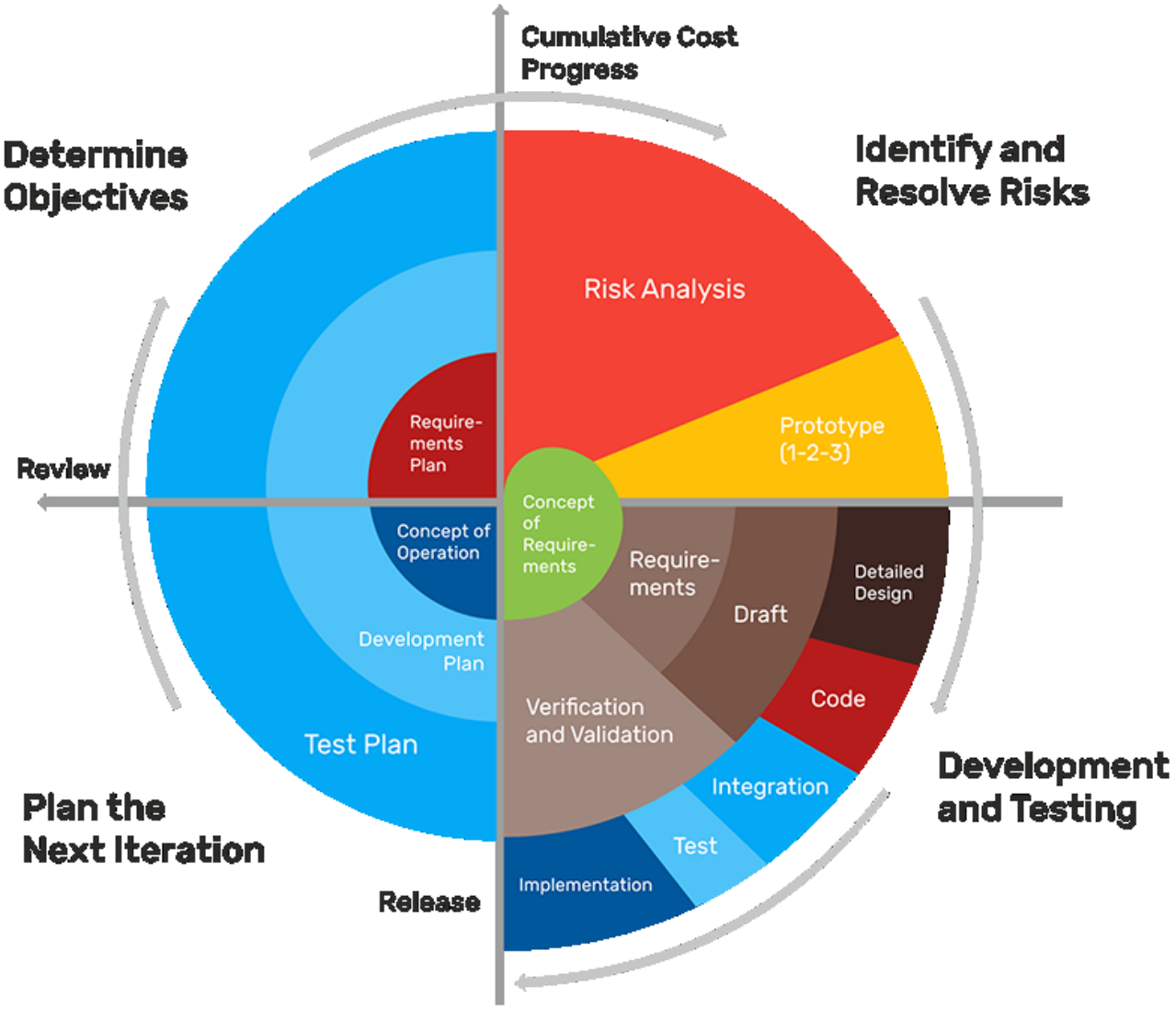
# Modelo V


V-MODEL SOFTWARE DEVELOPMENT
Requirement Gathering
System Analysis
Software Design
Module Design
Coding
Unit Testing
Integration Testing
System Testing
Acceptance Testing
VERIFICATION
VALIDATION

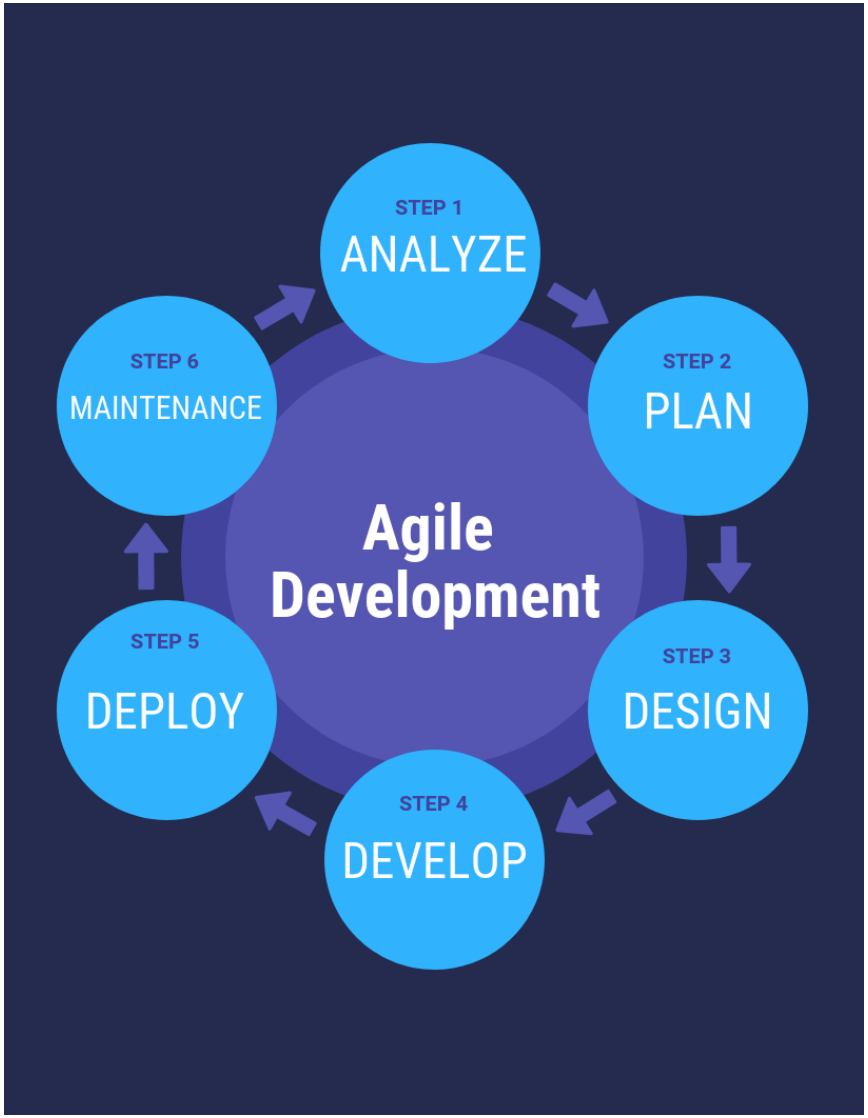
# Modelo Iterativo



# Modelo Espiral



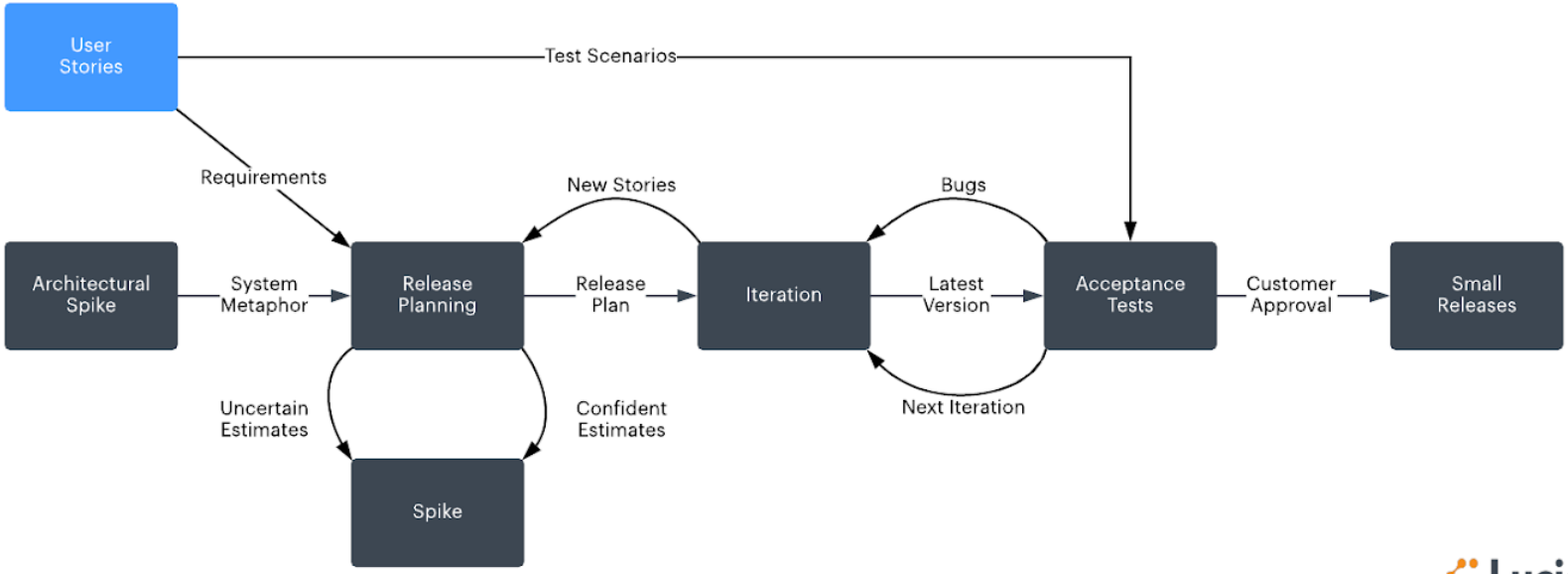
# Metodología Agil



# Caso Especial - SCRUM

# Metodología XP

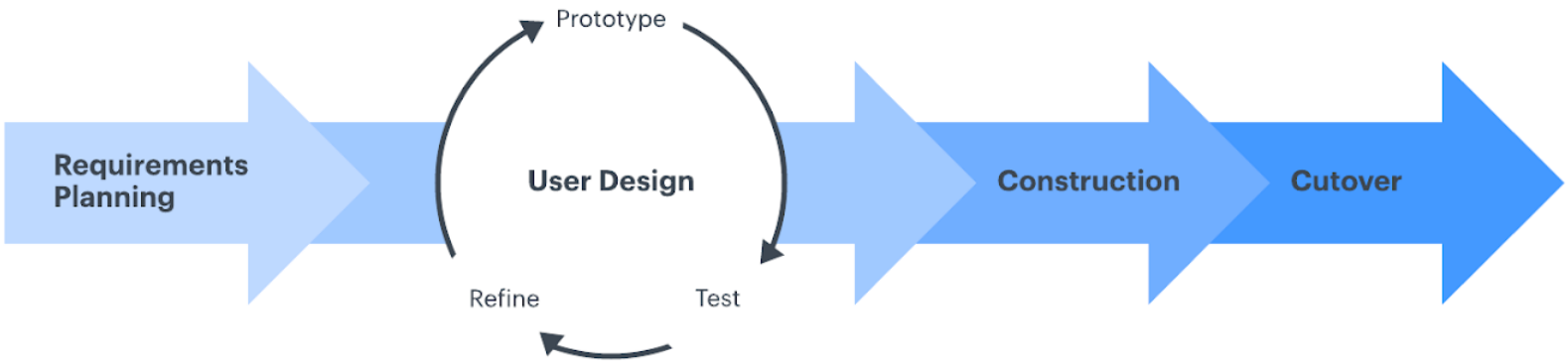
Extreme Programming (XP) Methodology




Made in Lucidchart

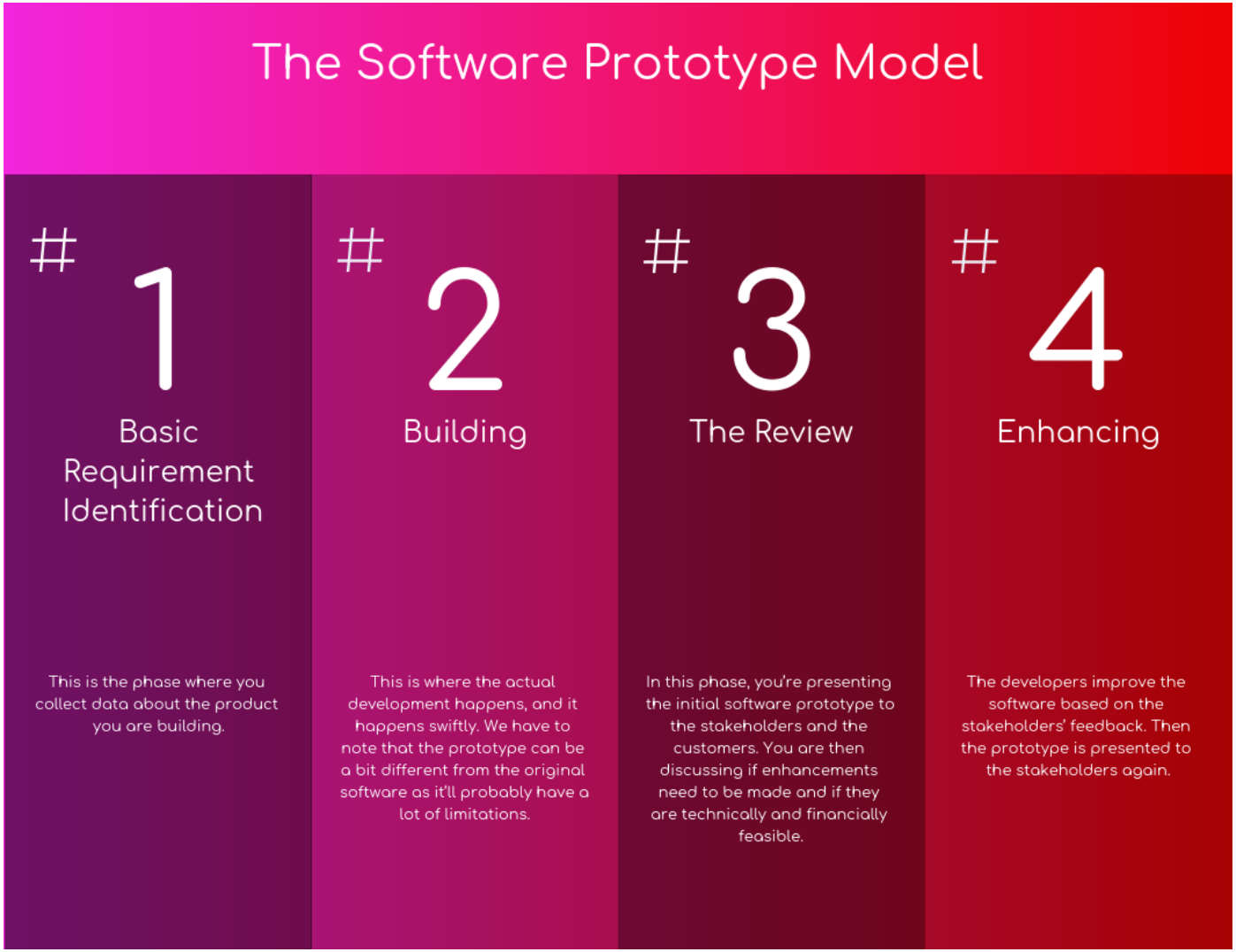
# Modelo RAD

Rapid Application Development (RAD)

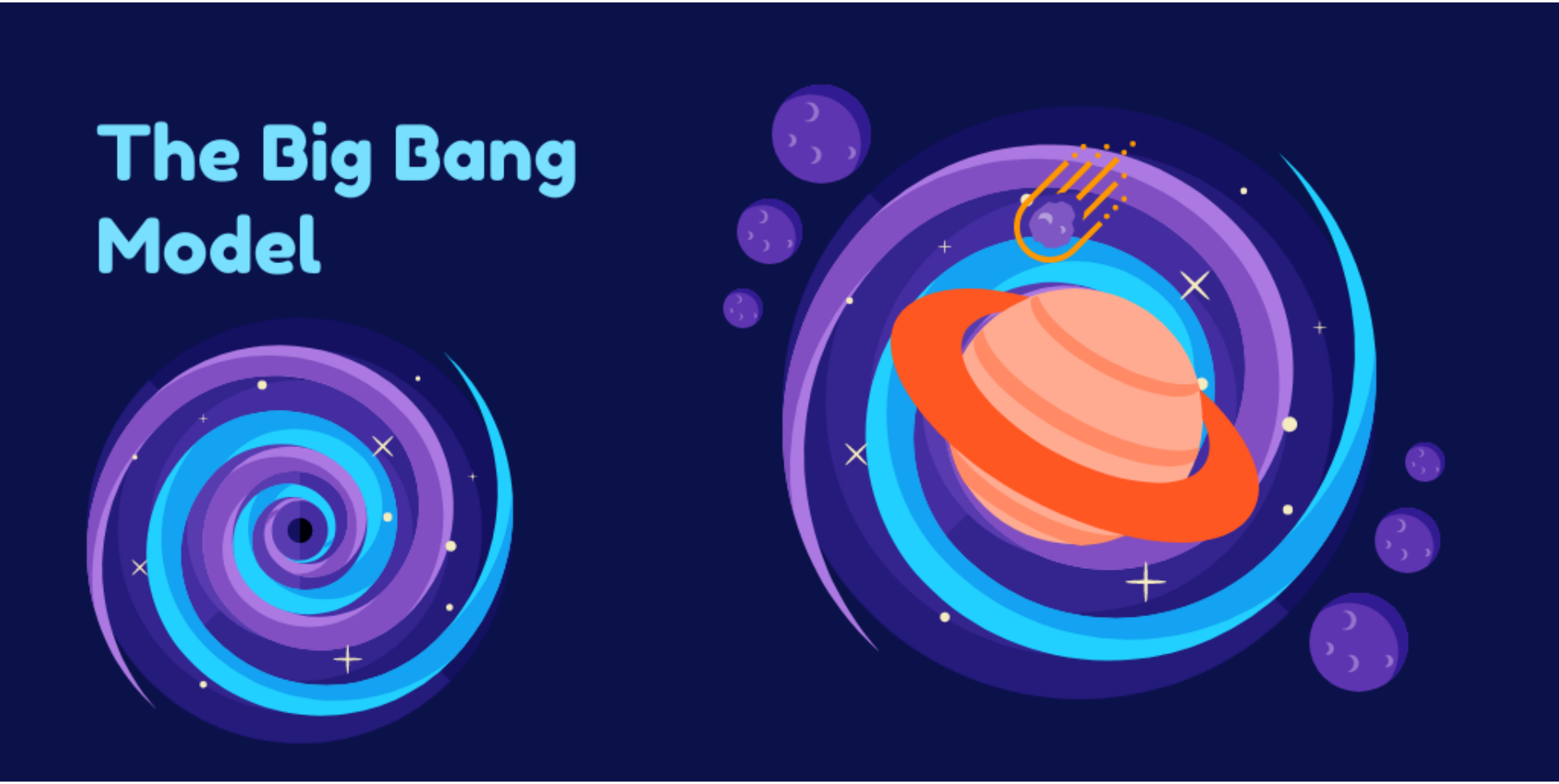



Made in Lucidchart

# Modelo de Prototipos



# Modelo Big Bang



# Referencias


- Langer, A. M. Guide to Software Development—Designing and Managing the Life Cycle. 2016.
- Richards, M. (2015). *Software architecture patterns* (Vol. 4). 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated
- https://blog.codegiant.io/software-development-life-cycle-the-ultimate-guide-2020-153d17bb20fb