



# Producción de Software

# Introducción

# Nelson Monzón López

**nelson.monzon@ulpgc.es**

# Agustin Salgado de la Nuez

agustin.salgado@ulpgc.es

# Daniel Santana Cedrés


daniel.santanacedres@ulpgc.es

# ¿Qué es el software?

---

*Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.*

**Extraído del estándar 729 del IEEE8**

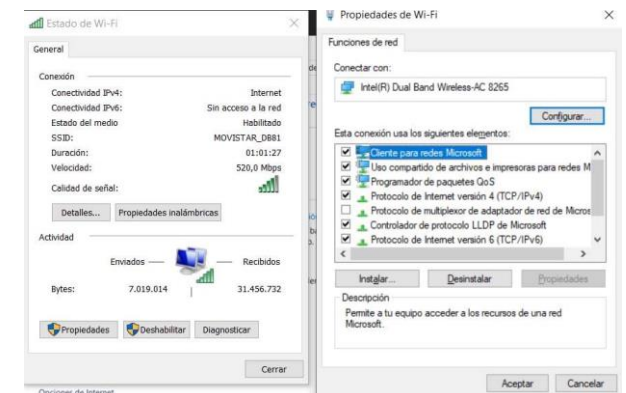
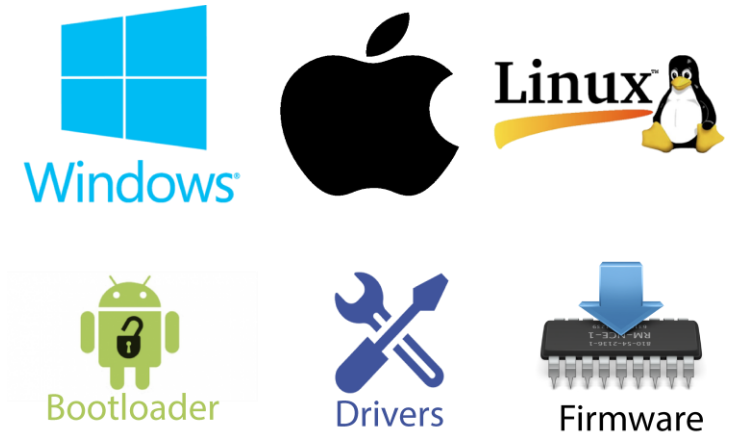
- Software de aplicación.
  - Software de programación.
  - Software de sistema.
- 

# Software de sistema

**Definición:** Programas que gestionan y controlan los recursos del hardware del ordenador, proporcionando una plataforma para que el software de aplicación funcione.

## Ejemplos:

- Microsoft Windows
- Ubuntu (y más distribuciones de Linux)
- Android
- iOS
- Unix
- BIOS
- Firmware
- Controladores de red
- Controladores de audio y video



# Software de programación

**Definición:** Es el conjunto de herramientas que permiten a los desarrolladores escribir, probar y mantener software. Incluye editores de código, compiladores, intérpretes y entornos de desarrollo integrados (IDE). También conocido como software de desarrollo.

Es software que nos permite crear más software 😊

## Ejemplos:

- Visual Studio Code (Editor de código)
- GCC (Compilador de C y C++)
- Python (Lenguaje de programación e intérprete)
- Eclipse (IDE para Java y otros lenguajes)
- NetBeans (Entorno de desarrollo para múltiples lenguajes)



# Software de aplicación

**Definición:** Es aquel **diseñado** para realizar tareas específicas para el **usuario final**, facilitando actividades como la gestión de documentos, edición de imágenes o navegación web.

Escritorio, web, mobile...

## Ejemplos:

- Aplicaciones ofimáticas
- Software educativo
- Software empresarial
- Almacenamiento de datos
- Videojuegos
- Software médico
- .....



# Software de aplicación: Características Operativas

---

**Corrección:** debe cumplir los objetivos para los cuales fue creado.

**Usabilidad:** fácil de aprender y entender.

**Integridad:** es decir, que no sea un software con efectos secundarios.

**Fiabilidad:** es decir, que el software no posea ningún defecto y no falle en la ejecución.

**Eficiencia:** la manera en la cual el software utiliza los recursos que tiene a su disposición, como vendría a ser el caso del espacio de almacenamiento, por ejemplo.

**Seguridad:** resistente a acciones y ataques externos



# Software de aplicación: Características de transición

---

**Interoperabilidad:** representa la disposición para intercambiar información con otras aplicaciones.

**Reutilización:** poder usar el código de software, con diversas modificaciones y para propósitos diversos.

**Portabilidad (compatibilidad):** que resulte accesible desde diversos equipos.



# Software de aplicación: Características de revisión

---

**Capacidad de mantenimiento:** al momento de crear el software, este debe ser fácil de mantener.

**Flexibilidad:** que tenga la disponibilidad para ser modificado por los desarrolladores.

**Extensibilidad:** es fácil de incrementar nuevas funciones.

**Escalabilidad:** el software debe ser fácil de actualizar para más trabajo.

**Modularidad:** tiene que estar compuesto por módulos y unidades independientes entre sí.

**Capacidad de prueba:** debe ser fácil la prueba del software.





# ¿Cómo crear software?

---

## **Análisis empresarial.**

Detectar necesidades del negocio (nicho de mercado)

¿Cómo podemos aportar valor y en dónde?

¿Por qué y para quién?

Presupuesto

Modelo de negocio



# ¿Cómo crear software?

---

## **Análisis empresarial.**

Detectar necesidades del negocio (nicho de mercado)

¿Cómo podemos aportar valor y en dónde?

¿Por qué y para quién?

Presupuesto

## **Modelo de negocio**

# Modelo de negocio

***Un modelo de negocio de software describe cómo una empresa crea, entrega y captura valor a través del desarrollo y distribución de software.***

## **Ejemplo: SaaS (Software as a Service)**

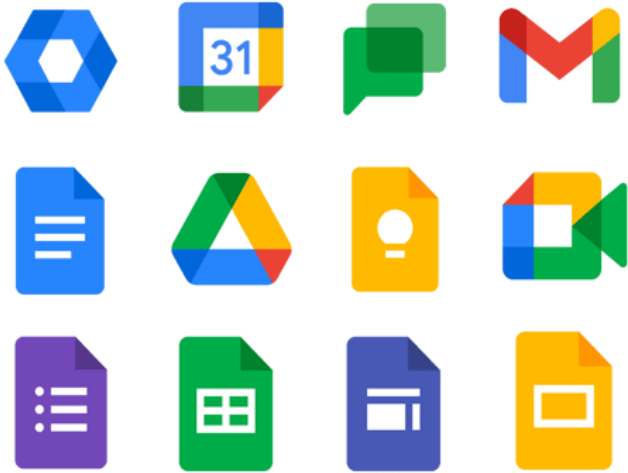
1. El software se ofrece a través de la nube (eliminar instalación local)
2. Acceso mediante suscripción (tarifa recurrente mensual, anual...)



# Ejemplos SAAS

---

Google Workspace



# Modelo de negocio

*Un modelo de negocio de software describe cómo una empresa crea, entrega y captura valor a través del desarrollo y distribución de software.*

## Ejemplo: SaaS (Software as a Service)

1. El software se ofrece a través de la nube (eliminar instalación local)
2. Acceso mediante suscripción (tarifa recurrente mensual, anual...)



### Componentes Clave:

- **Cloud:** Acceso desde cualquier lugar con conexión a internet.
- **Actualizaciones Automáticas:** “Simplifica” mantenimiento y actualizar el software sin intervención del usuario.
- **Escalabilidad:** Ajuste del servicio a las necesidades de diferentes tamaños de empresas .

*(Necesitas medir espacio en disco, medir coste de acceso/consultas a la nube, etc.)*

# Modelo de negocio

## Beneficios:

- **Reducción de Costos:** No es necesario invertir en hardware o mantenimiento (analizar coste CLOUD claro..).
- **Flexibilidad:** Los usuarios pueden acceder al software desde cualquier dispositivo.
- **Seguridad:** El proveedor “se encarga” de la seguridad y el cumplimiento de normativas (¿costes?).

## Desafíos:

- **Dependencia de Internet:** El acceso al software depende de una conexión estable a internet.
- **Privacidad de Datos:** Los datos de los usuarios se almacenan en los servidores del proveedor, lo que puede generar preocupaciones sobre la privacidad.

## Monetización



shutterstock.com · 1336157243

- **Freemium:** Funcionalidades básicas gratuitas con opción de pago para versiones premium.
- **Suscripción:** Pago mensual o anual por acceso.
- **Licenciamiento Empresarial:** Tarifas personalizadas para empresas.

# ¿Cómo crear software?

---

## **Análisis empresarial.**

Detectar necesidades del negocio (nicho de mercado)

¿Cómo podemos aportar valor y en dónde?

¿Por qué y para quién?

Presupuesto

Modelo de negocio

## **Plan de trabajo**

Recursos (personas, capacidades, tecnología,...)

Actividades (definidas en tiempo y prioridad)

Calendario (objetivos a corto, medio y largo plazo)

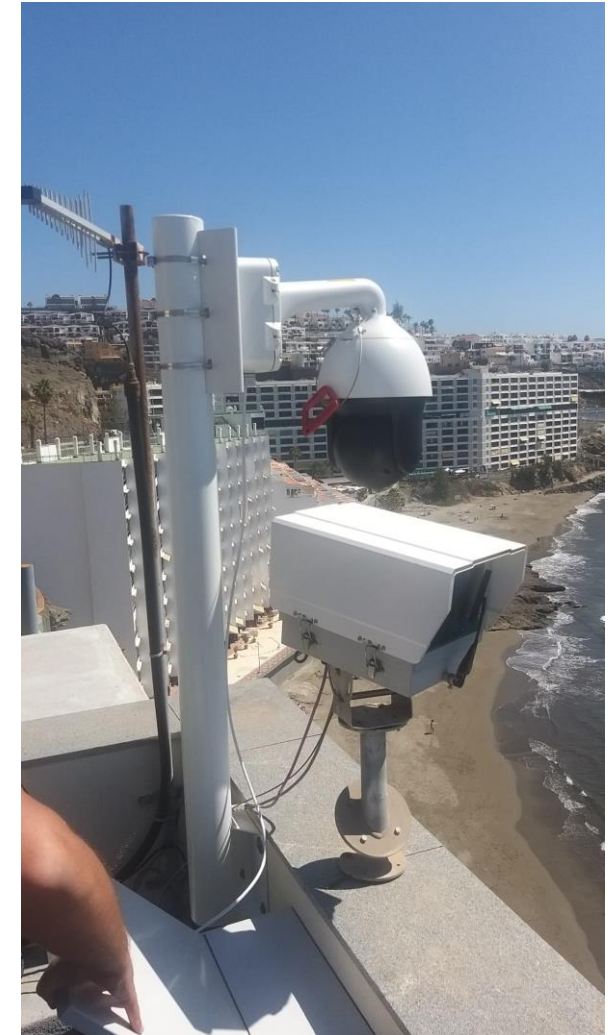


# Proyecto Investigación Industrial

## Estrategias Precisas de Visión por Computador para la Gestión Inteligente de la Costa y el Litoral

Duración: 3 años

Proyecto Investigación Industrial

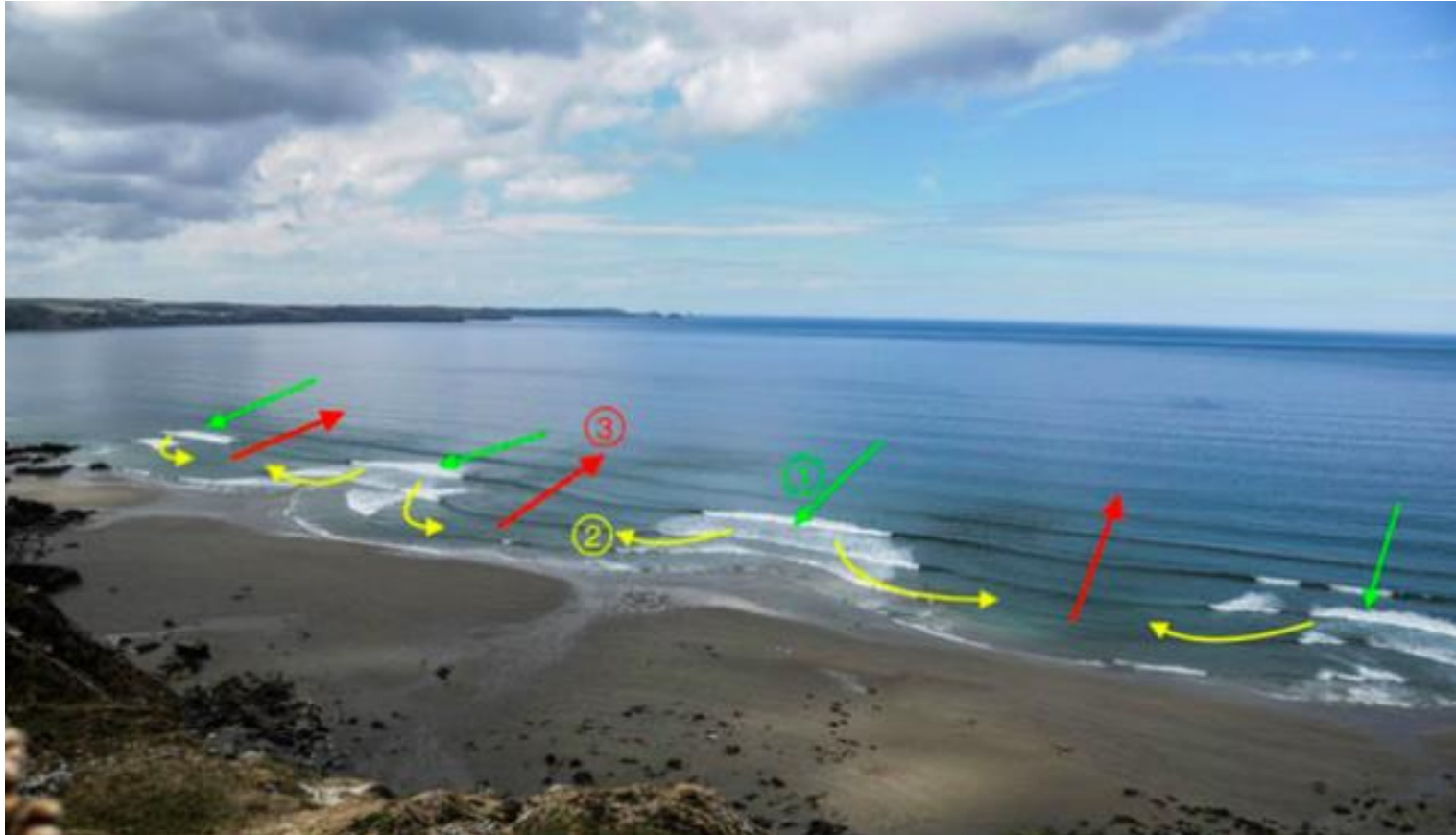




# Cámaras PTZ



# Detectar peligros en la costa y el mar





# Detectar peligros en la costa y el mar



# Plan de trabajo(s)

---

## Objetivo 1: Mejora de imágenes y vídeo

Objetivo 1.1 Mejora de la percepción de las escenas

Objetivo 1.2. Estabilización de vídeo



## Objetivo 2: Detección de peligros en el entorno marítimo

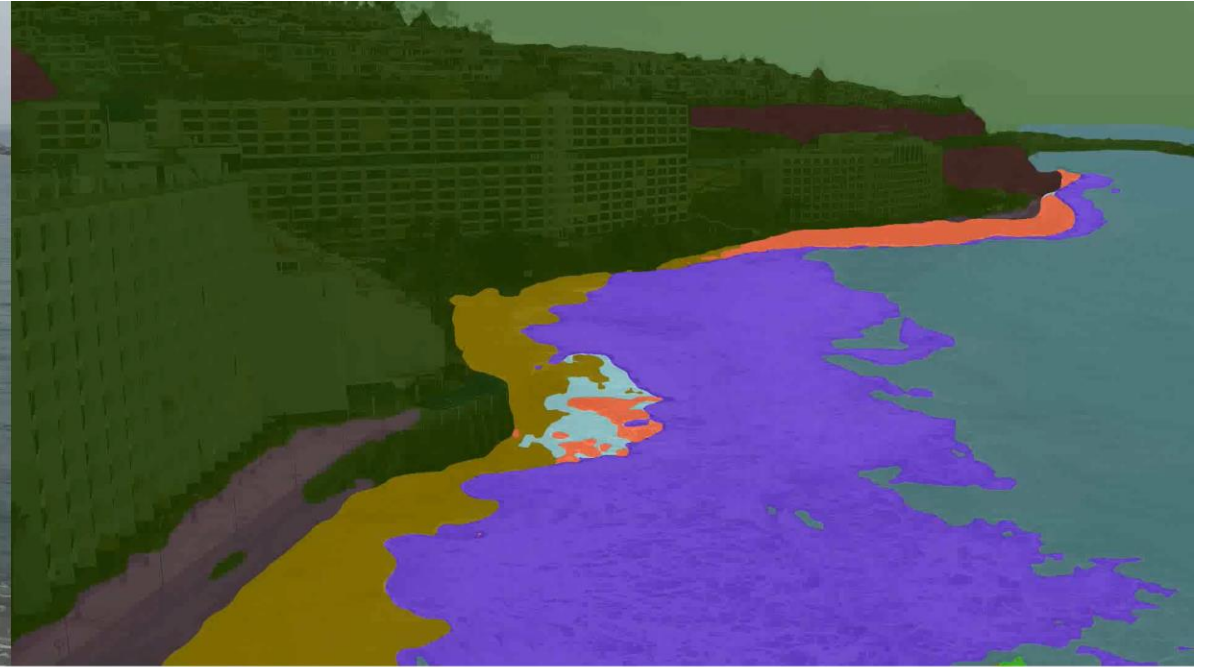
Objetivo 2.1: Detección de zonas de riesgo por oleaje y corrientes de resaca

Objetivo 2.2: Detección de objetos en el entorno marítimo



[illegible]

# Plan de trabajo(s)

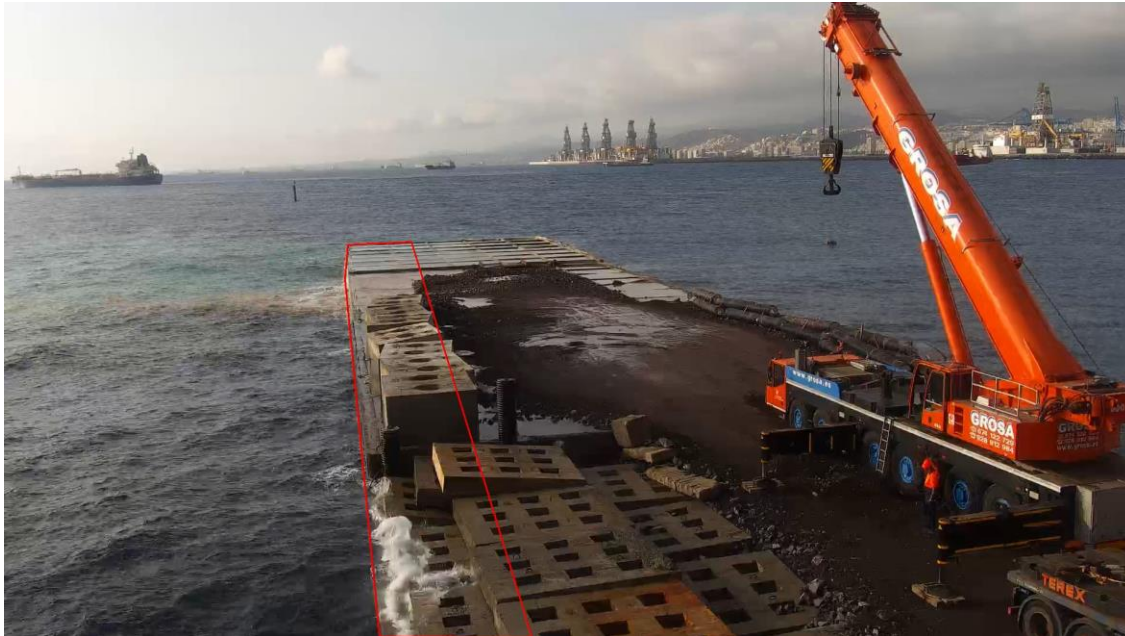


Name	Colour	Name	Colour	Name	Colour	Name	Colour	Name	Colour
void		foam		pier		person		building	
beach		sea		dike		object		sky	
sand		ship		rock		floor		mountain	



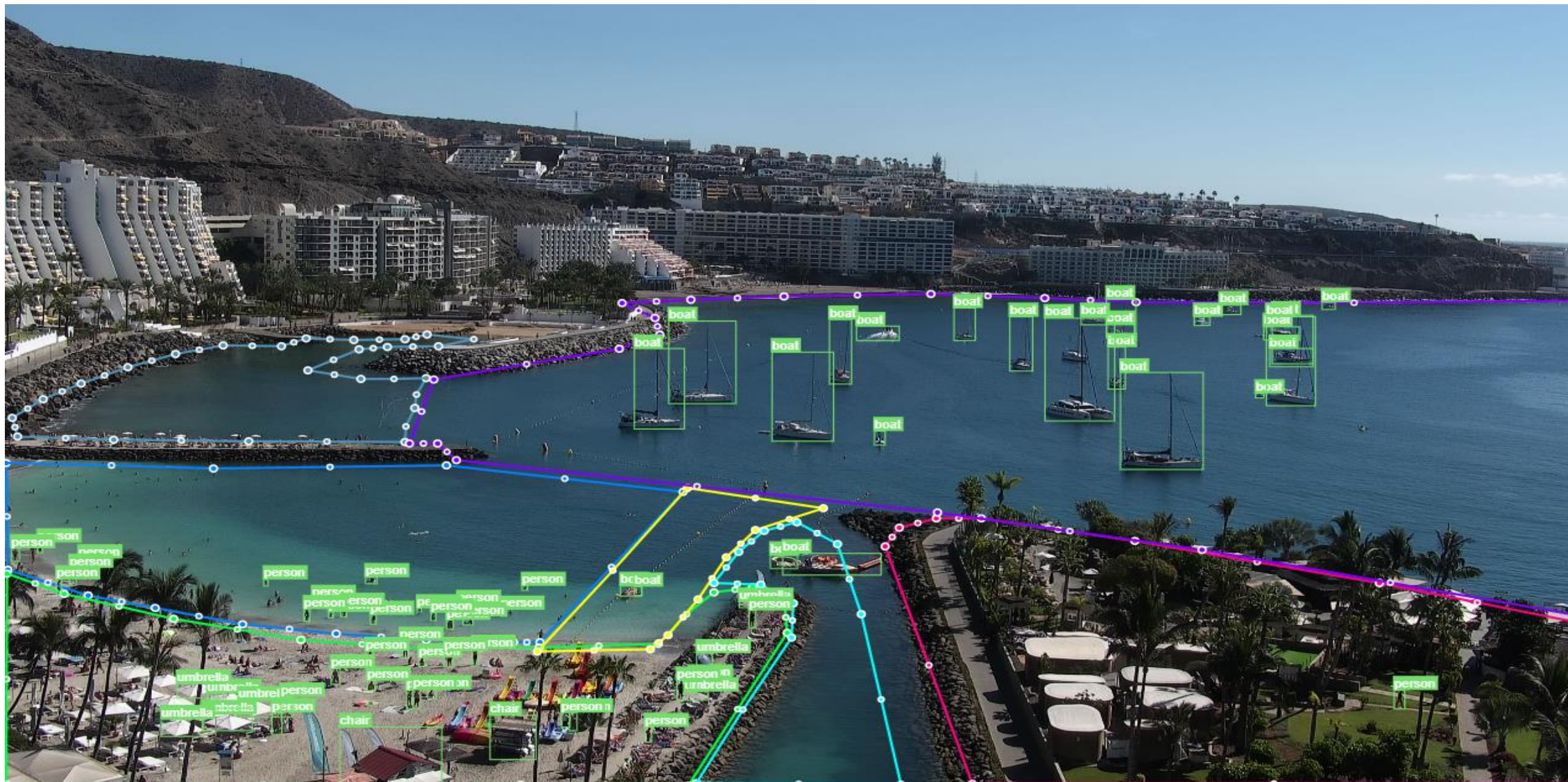
# Resultados del plan de trabajos...

Name	Colour	Name	Colour	Name	Colour	Name	Colour	Name	Colour
void		foam		pier		person		building	
beach		sea		dike		object		sky	
sand		ship		rock		floor		mountain	





# Resultados del plan de trabajos...





# ¿Cómo crear software?

---

## **Análisis empresarial.**

Detectar necesidades del negocio (nicho de mercado)

¿Cómo podemos aportar valor y en dónde?

¿Por qué y para quién?

Presupuesto

Modelo de negocio

## **Plan de trabajo**

Recursos (personas, capacidades, tecnología,...)

Actividades (definidas en tiempo y prioridad)

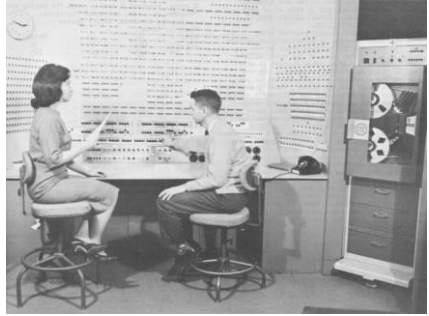
Calendario (objetivos a corto, medio y largo plazo)

## **Políticas de mejora continúa**

Reflexionar sobre buenas prácticas y estrategias que mejoren procesos



# Evolución del sector tecnológico



1950

1960

1970

1980

1990

2000

2010

**Crisis  
del  
Software  
(1968)**



---

Funcionalidad deficiente  
Estimaciones incorrectas  
Desbordamiento de costes  
Objetivos no definidos desde un inicio  
No somos adivinos!!

- Es imposible predecir los impedimentos
- Cambios de última hora

Poca o nula comunicación

- El trabajador no sabe que se le pide
- El cliente no tiene realmente claro lo que quiere

Necesidad de  
**Gestión de  
proyectos**

# Ingeniería del software

---

- Definiciones:

*Es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información*

*Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar software nuevo*

- Una metodología representa el camino para desarrollar software de una manera sistemática.
- 

# Ingeniería del software

---

Reutilizar conocimientos.

Proceso estándar en la organización.

Aplicar procesos de forma sistemática.

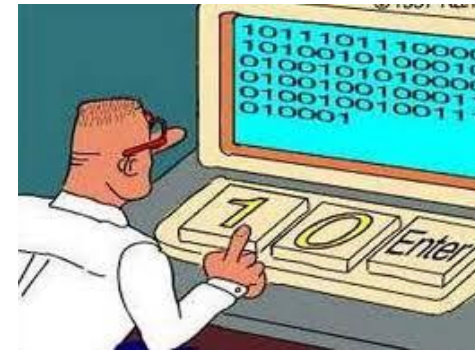
Acelerar y profesionalizar el desarrollo.

No reinventar la rueda a menos que necesitemos que vuele.



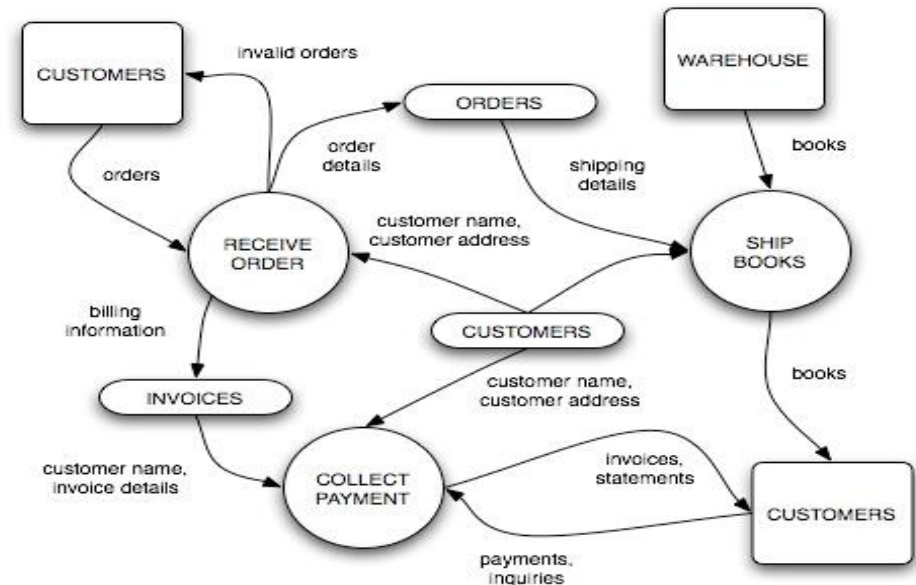
# Modelos de trabajo

- Años 50 - Desarrollo convencional
  - Desarrollo artesanal
  - No existían metodologías de desarrollo
  - Desarrolladores = Programadores
  - Principales problemas:
    - Los resultados eran impredecibles
    - No se controlaba el proyecto
    - Los cambios organizativos afectaban al proceso
- Necesidad de grandes sistemas software



# Modelos de trabajo

- Años 70 – Desarrollo estructurado
  - De artesanal a metodológico
  - Programación estructurada
    - Se define forma estática y dinámica de un programa
    - Normas para estructuras de datos y control
    - Técnicas de programación estructurada de Warnier y Jacobson
  - Diseño y Análisis estructurado
  - Aparecen los primeros métodos
    - 75 – Myers, Yourdon y Constantine
    - 80 – Page y Jones
  - Nivel de abstracción más amplio



# Modelos de trabajo

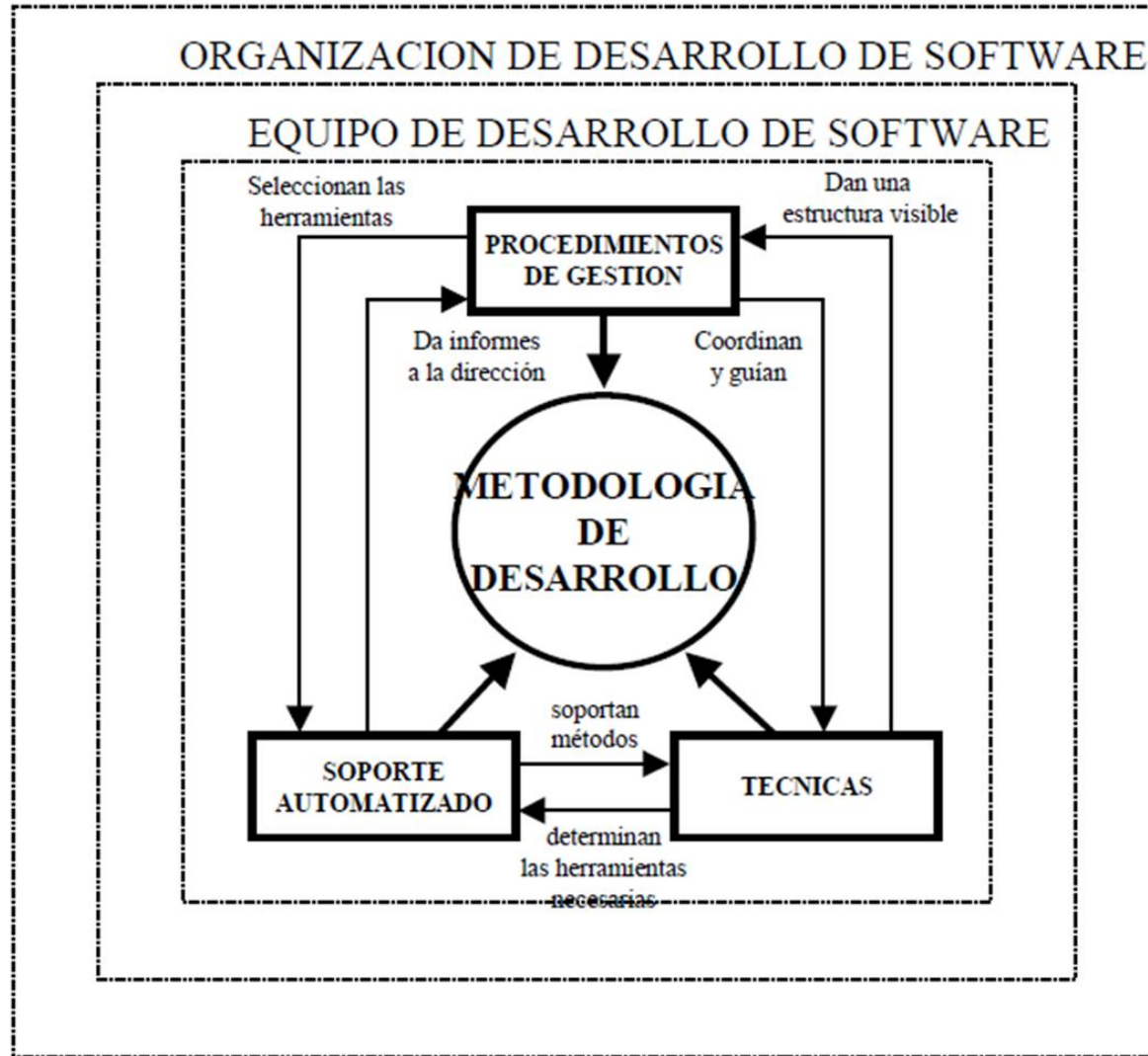
---

- Años 80 – Desarrollo Orientado a Objetos
  - Se tratan procesos y datos de forma conjunta
  - Lenguajes orientados a objeto
  - Métodos *evolutivos y revolucionarios*
  - Booch, OOSD, OMT, Objectory,
- Años 90 – Metodologías OO y Ágiles
  - 1995 – Scrum
  - 1997 – UML
  - 1998 – Proceso Unificado
  - 1999 – XP
- Años 2000 – Consolidación de Metodologías Ágiles
  - 2001 – Manifiesto Ágil ([agilemanifesto.org](http://agilemanifesto.org))

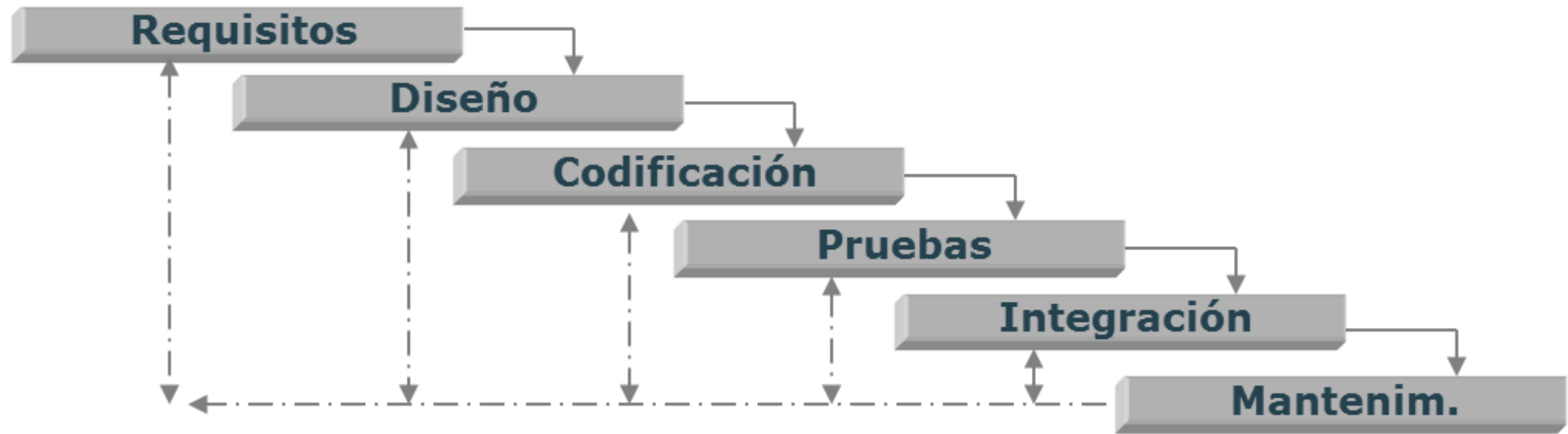


# Modelos de trabajo

## ENTORNO DE DESARROLLO DE SOFTWARE



# Gestión Predictiva




# Modelo de construcción de prototipos

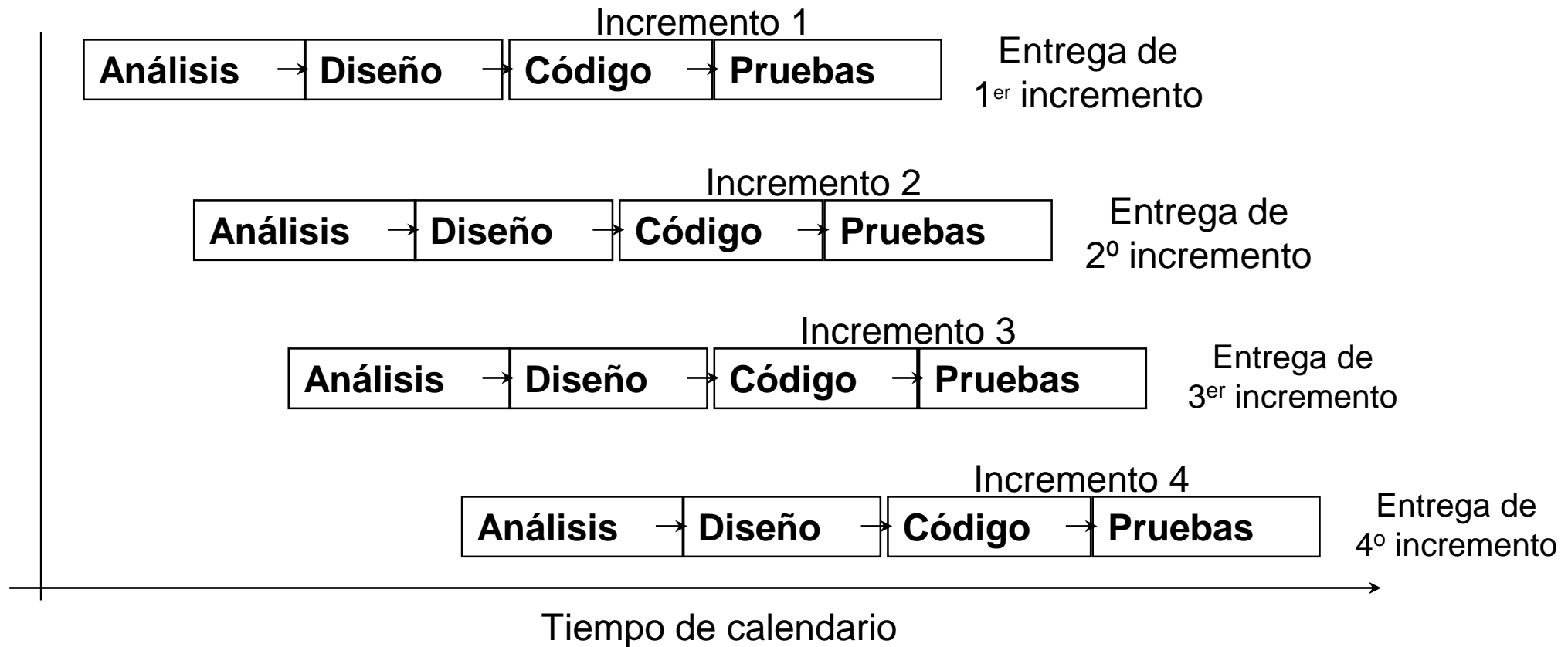


# Modelo de construcción de prototipos

---


- Ventajas
    - Ayuda a identificar los requisitos
    - Agrada tanto a los clientes como a los desarrolladores
  - Inconvenientes
    - El cliente considera al prototipo como el producto final, listo para usar.
    - la calidad del software o la factibilidad de mantenimiento no se tienen en cuenta
    - El desarrollador a menudo hace compromisos de implementación
- 

# Modelo incremental



# Modelo incremental

---

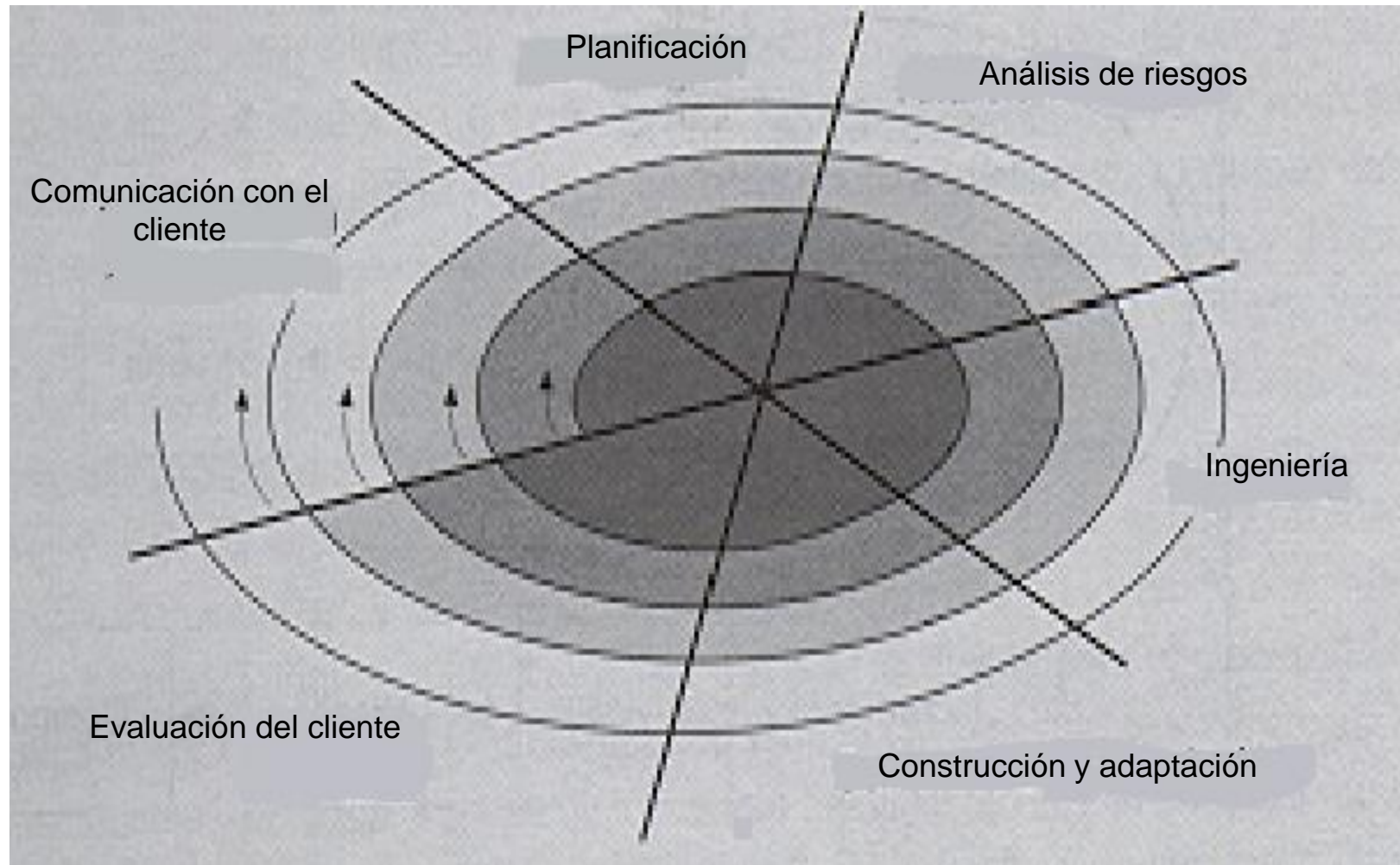
- Combina elementos del modelo lineal con la filosofía de creación de prototipos
  - El primer incremento a menudo es un producto esencial
  - A partir de la evaluación se planea el siguiente incremento y así sucesivamente
  - Es interactivo por naturaleza
  - Es útil cuando el personal no es suficiente para la implementación completa
- 

# Modelo incremental

---

- Ventajas
  - Se puede financiar el proyecto por partes
  - Apropiado para proyectos grandes de larga duración
  - No se necesita tanto personal al principio como para una implementación completa
- Inconvenientes
  - Se necesitan pruebas de regresión
  - Pueden aumentar el coste debido a las pruebas


# Modelo en espiral





# Modelo en espiral

---

- **Comunicación con el cliente:** Para establecer comunicación entre el desarrollador y el cliente.
  - **Planificación:** Para definir los recursos, el tiempo y otras informaciones relacionadas con el proyecto.
  - **Análisis de riesgos:** Para evaluar riesgos técnicos y operativos.
  - **Ingeniería:** Para construir una o más representaciones de la aplicación.
  - **Construcción y adaptación:** Para construir, probar, instalar y proporcionar soporte al usuario
  - **Evaluación del cliente:** Para obtener la reacción del cliente según la evaluación de las representaciones del software
- 

# Gestión basada en procesos





# Producción de Software

## Tema 1

Nelson Monzón López  
[nelson.monzon@ulpgc.es](mailto:nelson.monzon@ulpgc.es)