
# Modelos de proceso de software

Elizabeth Suescún Monsalve, PhD
esuescu1@eafit.edu.co

# Agenda

**Modelos de procesos Evolutivos e Iterativos:**

- Modelo en Espiral
- Proceso Unificado de Rational (RUP)
- Modelos de construcción de prototipos
- Conclusiones

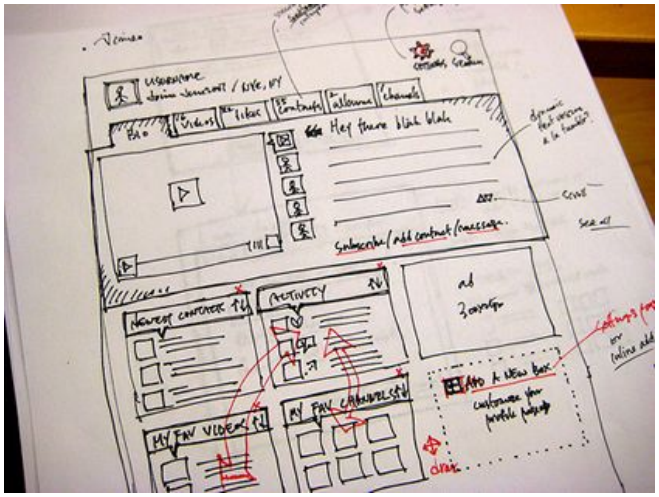
# Modelos de Procesos Evolutivos

# Modelos de procesos Evolutivos

- El desarrollo evolutivo consta del desarrollo de una **versión inicial** que luego de exponerse se va **refinando** de acuerdo a los comentarios o nuevos requisitos por parte del cliente o del usuario final.
- Los modelos **evolutivos son iterativos**, se caracteriza por la forma en que permiten a los ingenieros en software desarrollar **versiones cada vez más completas** del software.

# Modelos de procesos Evolutivos

- Algunos de los modelos que se clasifican en esta categoría.
  - Construcción de prototipos
  - Modelos en espiral
  - Modelo de desarrollo concurrente



# Construcción de prototipos

# Construcción de prototipos

- En Ingeniería de software la construcción de prototipos pertenece a los modelos de desarrollo **evolutivo**.
- El **prototipo debe ser construido en poco tiempo**, usando los programas adecuados y **no se debe utilizar mucho dinero** pues a partir de que este sea **aprobado** es que el desarrollo puede iniciar.

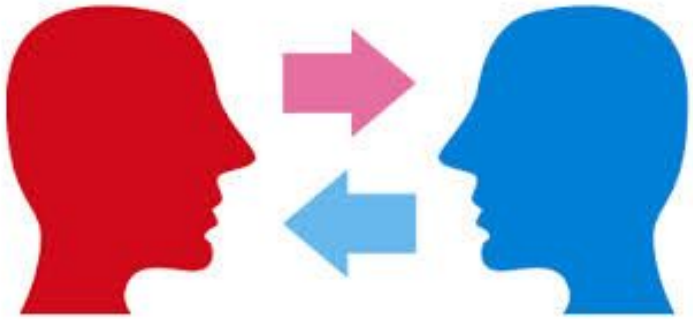
# Construcción de prototipos

- El diseño rápido se basa en una representación de aquellos "**aspectos del software que serán visibles para el cliente o el usuario final**" (por ejemplo, la configuración de la interfaz con el usuario y el formato de los despliegues de salida).
- El diseño rápido conduce a la construcción de un prototipo, el cual es evaluado por el cliente o el usuario para una **retroalimentación**; **gracias a ésta se refinan los requisitos del software que se desarrollarán.**

# Construcción de prototipos

Entender los requisitos:

**La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente**. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.



# Construcción de prototipos

Sirve para la comunicación:

**Se puede utilizar como un modelo del proceso independiente.**

**Ayuda al desarrollador de software y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos.**

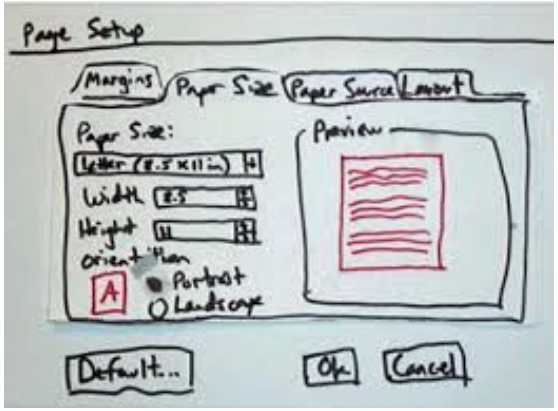
# Construcción de prototipos descartables

# Tipos de prototipos

- **De Interface:** modelo en papel o bajo algún programa que permite mostrar pantallas, menús e/o navegación.
- **De comportamiento:**
  - En **anchura**: muestra menús y algún tipo de comportamiento.
  - En **profundidad**: cubre ciertas funciones.
- **Completo** pero de baja calidad o rendimiento.

https://www.youtube.com/watch?v=iWny4LLVHAw

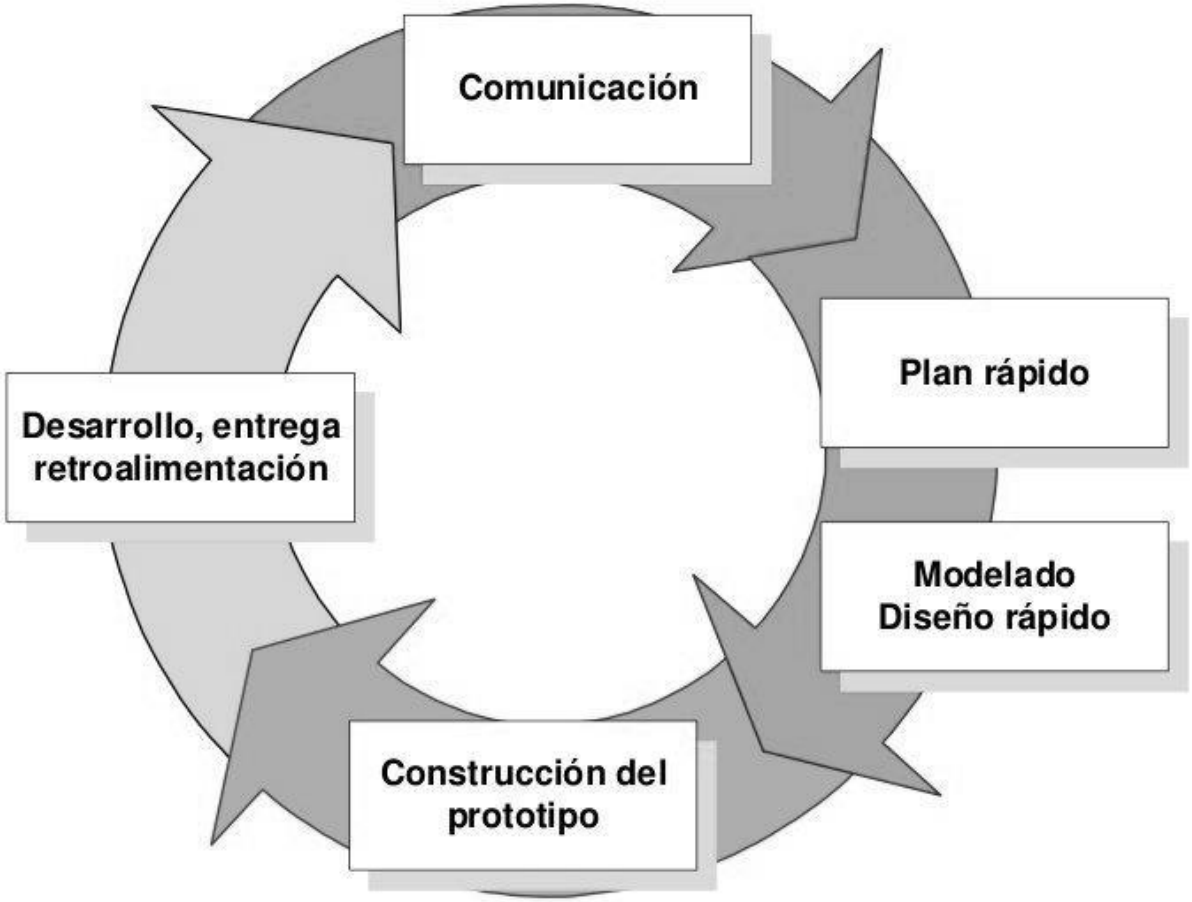
https://www.youtube.com/watch?v=E174-opnsPQ



# Construcción de prototipos

- **Etapas**:
  - Plan rápido.
  - Modelado, diseño rápido.
  - Construcción del Prototipo.
  - Desarrollo, entrega y retroalimentación.
  - Comunicación.

# Construcción de prototipos



# Construcción de prototipos

- **Ventajas**:
  - Este modelo es útil cuando **el cliente conoce los objetivos generales para el software, pero no identifica los requisitos** detallados de **entrada, procesamiento o salida.**
  - Ofrece un mejor enfoque cuando **el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo**, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.

# Construcción de prototipos

- **Ventajas**:
  - Reduce el riesgo de construir productos que no **satisfagan** las necesidades de los usuarios.
  - Reduce costos y aumenta la **probabilidad de éxito**.
  - Exige disponer de las **herramientas adecuadas.**
  - **Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.**

# Construcción de prototipos

- **Desventajas**:
  - **El usuario tiende a crearse unas expectativas cuando ve el prototipo de cara al sistema final.**
  - A causa de la intención de crear un prototipo de forma rápida, se suelen **desatender** aspectos importantes, tales como la **calidad** y el **mantenimiento a largo plazo**, lo que obliga en la mayor parte de los casos a **reconstruirlo** una vez que el prototipo ha cumplido su función.

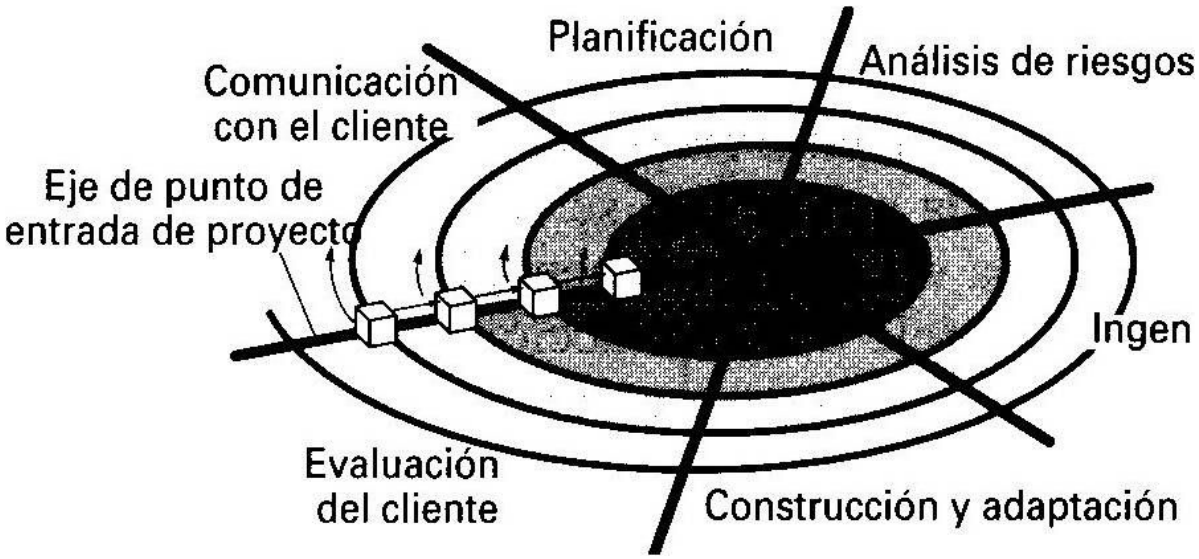
# Construcción de prototipos

- **Desventajas**:
  - El desarrollador suele tomar **algunas decisiones de implementación poco convenientes** (por ejemplo, elegir un lenguaje de programación incorrecto porque proporciona un desarrollo más rápido).
  - Con el paso del tiempo, el desarrollador puede olvidarse de la razón que le llevó a tomar tales decisiones, con lo que se corre el riesgo de que dichas elecciones pasen a formar parte del sistema final.

# Construcción de prototipos

- **Alerta**:

  Es frecuente que **el usuario** se muestre reacio o **pida que sobre ese prototipo se construya el sistema final**, lo que lo convertiría en un **prototipo evolutivo**, pero partiendo de un estado poco recomendado.




# Modelo en Espiral

# Modelo en Espiral

- Representa, en forma de espiral, una secuencia de actividades.
- Originalmente propuesto por Boehm en **1988**, y se diferencia de los demás modelos por considerar el **riesgo**.
- Para la ingeniería de software es actualmente **el enfoque más realista para el desarrollo de software y de sistemas a gran escala.**
  - Utiliza un enfoque evolutivo, permitiendo al desarrollador y al cliente **entender y reaccionar ante los riesgos** en cada nivel evolutivo.

# Modelo en Espiral

El modelo en espiral se divide en un número de **actividades estructurales**, también llamadas **regiones de tareas**, según Sommerville el ciclo de vida del modelo en espiral se divide en cuatro sectores:

1. **Definición de objetivos**. En esta fase se identifica las **restricciones** del proceso y el producto, y algunos **riesgos** para trazar objetivos y respectivamente **planes estratégicos**.

# Modelo en Espiral

2. **Evaluación y reducción de riesgos**. Se hace un análisis detallado para cada riesgo y se establece los pasos para reducirlo.
3. **Desarrollo y validación**. Después de evaluar los riesgos, se elige un modelo para el desarrollo del sistema.
4. **Planificación**. El proyecto se revisa y se toma la decisión de si debe continuar con un ciclo posterior de la espiral.

# Modelo en Espiral

- Características:
  - El software se desarrolla en una serie de **versiones incrementales.**
  - La versión incremental podría ser un modelo en papel o un prototipo.
  - Con cada incremento, se producen **versiones cada vez más completas.**

# Modelo en Espiral

- **Ventajas:**
  - Puede **adaptarse** y aplicarse a lo largo de la vida del software.
  - Como el software evoluciona, a medida que progresa el proceso, **el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos** en cada uno de los niveles evolutivos.

# Modelo en Espiral

- **Ventajas:**
  - Permite a quien lo desarrolla aplicar el **enfoque de construcción de prototipos** en cualquier etapa de evolución del producto.
  - Demanda una **consideración directa de los riesgos** técnicos en todas las etapas del proyecto. Reduce los riesgos antes de que se conviertan en problemas.

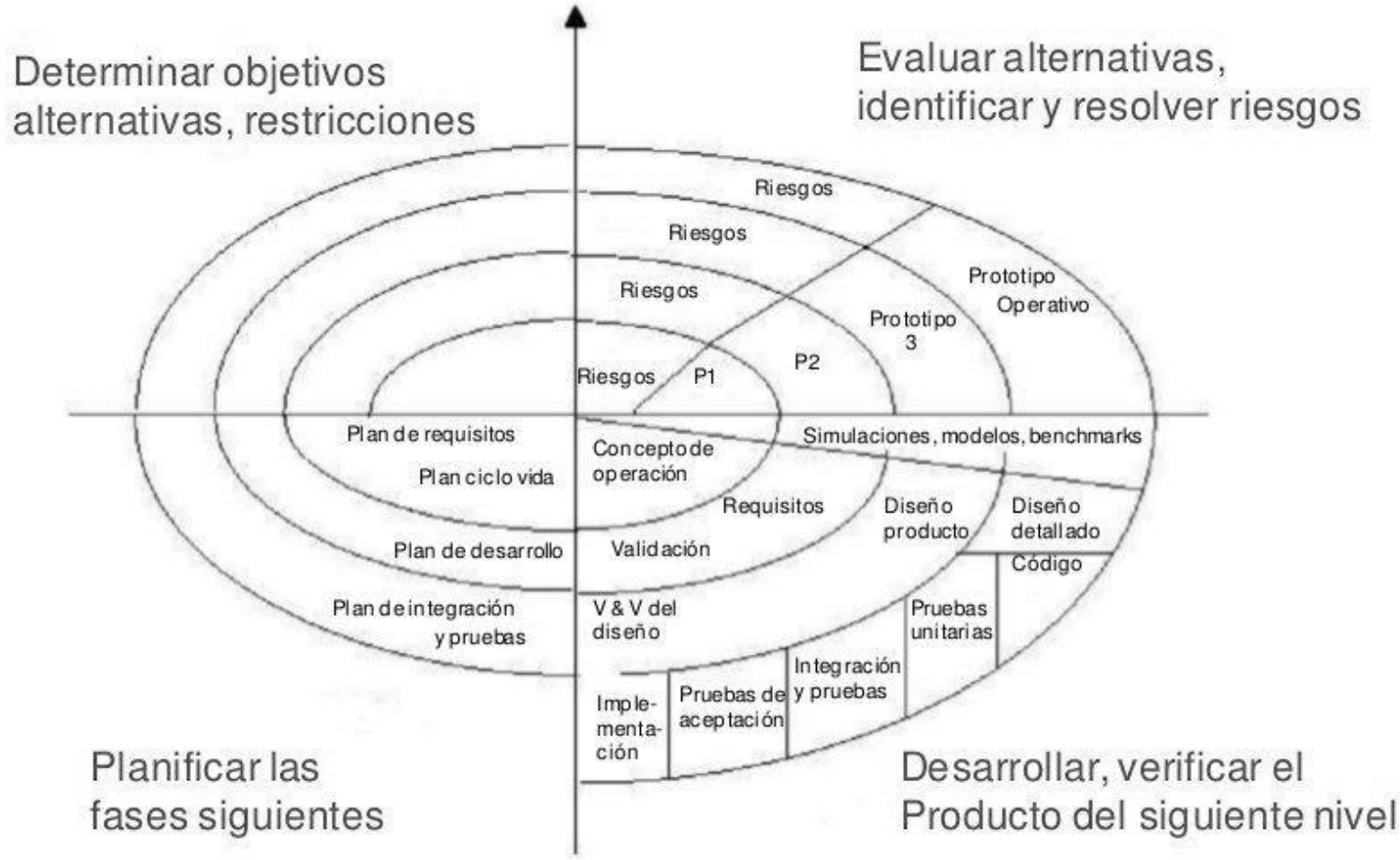
# Modelo en Espiral

- **Desventajas:**
  - Puede resultar difícil **convencer a los clientes** de que el enfoque evolutivo es controlable (particularmente en situaciones de contrato).
  - **Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas.**

# Modelo en Espiral

- **Desventajas:**
  - Aún no es un modelo muy utilizado.
  - Se requiere de experiencia en la identificación de riesgos y refinamiento para su uso generalizado.

# Modelo en Espiral




# Proceso Unificado de Rational (RUP)

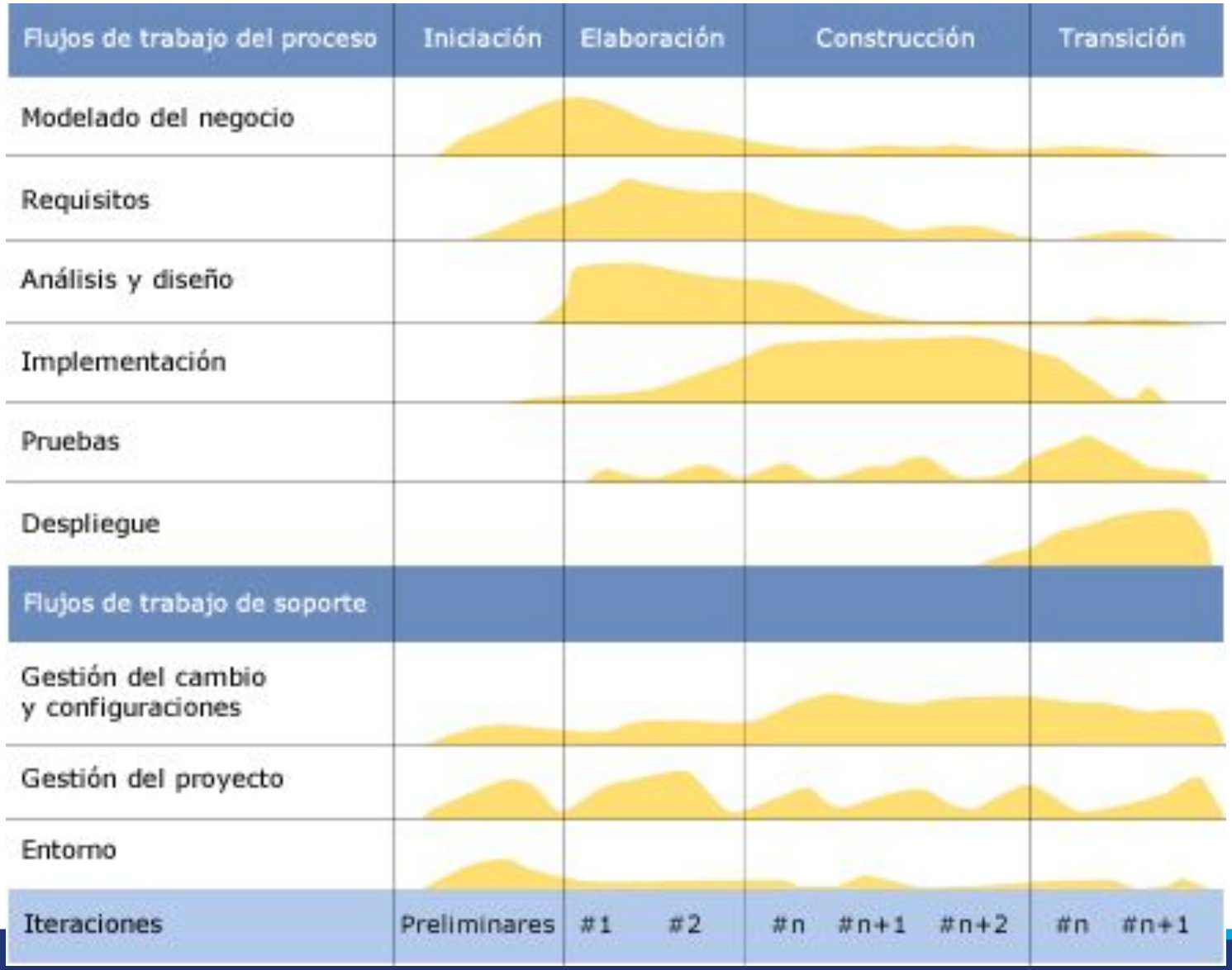
# Proceso Unificado de Rational (RUP)

- Es una **metodología** de desarrollo de **software orientada a objetos** creada por Rational Software Corporation y actualmente IBM es su propietaria.

- La **metodología** estándar **más utilizada** para el análisis, diseño, implementación y documentación de sistemas **orientados a objetos.**

## Fases

**Actividades o Disciplinas**



# Mejores Prácticas de RUP

- **Desarrollo de software de manera iterativa**
- **Gestión de requisitos**
- **Usar arquitectura basada en componentes**
- **Software modelado visualmente**
- **Verificar la calidad del software**
- **Controlar los cambios al software**

# Artefactos - Fases

**Inicio**: Documento Visión, Diagramas de caso de uso, Especificación de Requisitos, Diagrama de Requisitos

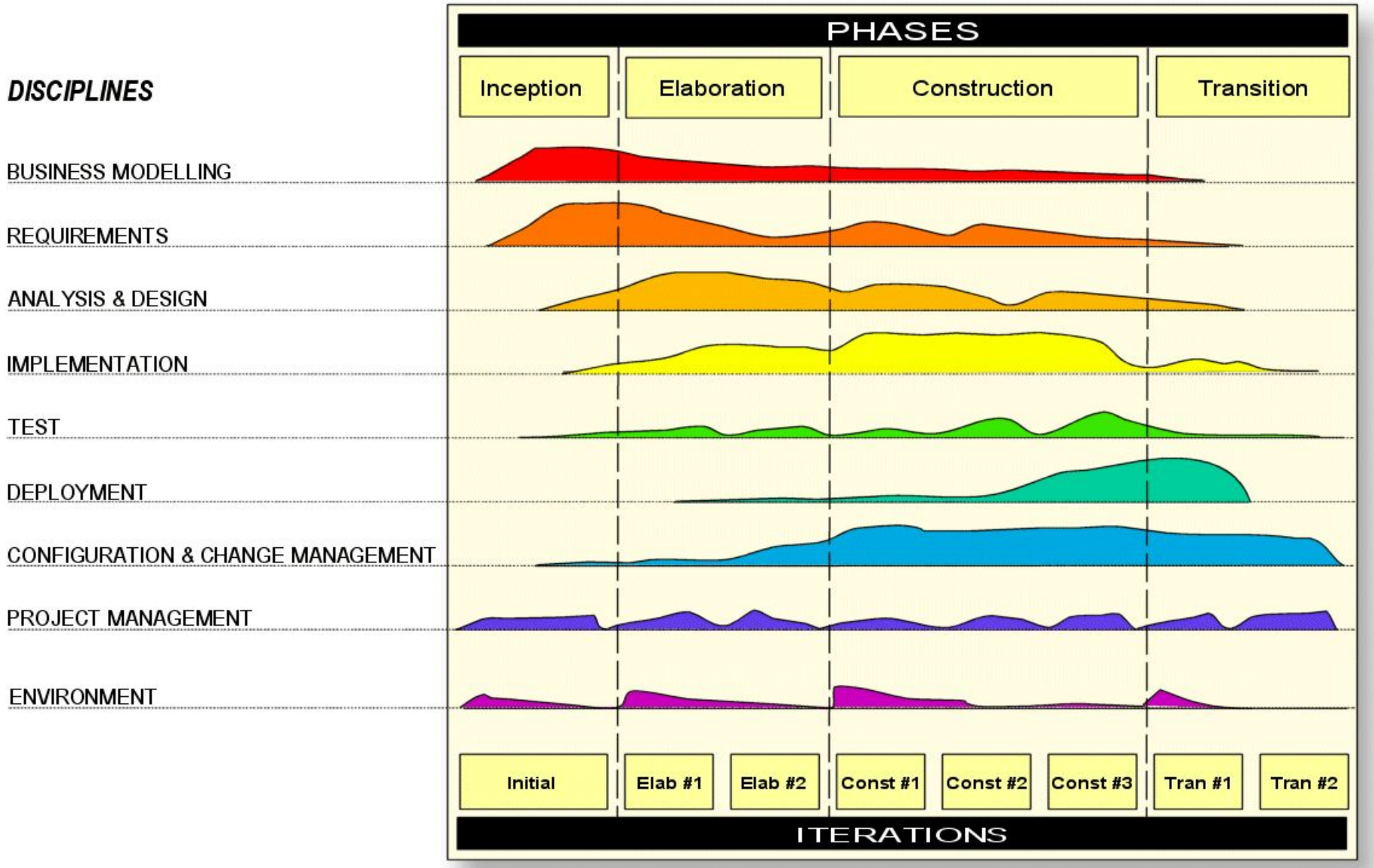
**Elaboración**: Documento Arquitectura que trabaja con las siguientes vistas:

- **Vista Lógica**
  Diagrama de clases, Modelo E-R (Si el sistema así lo requiere)
- **Vista de Implementación** Diagrama de Secuencia, Diagrama de estados, Diagrama de Colaboración
- **Vista Conceptual**
  Modelo de dominio
- **Vista física**
  Mapa de comportamiento a nivel de hardware. Diseño y desarrollo de casos de uso, o flujos de casos de uso arquitectónicos, Pruebas de los casos de uso desarrollados, que demuestran que la arquitectura documentada responde adecuadamente a requerimientos funcionales y no funcionales.

# Artefactos - Fases

**Construcción**: Especificación de requisitos faltantes, Diseño y desarrollo de casos de uso y/o flujos de acuerdo con la planeación iterativa, Pruebas de los casos de uso desarrollados, y pruebas de regresión según sea el caso

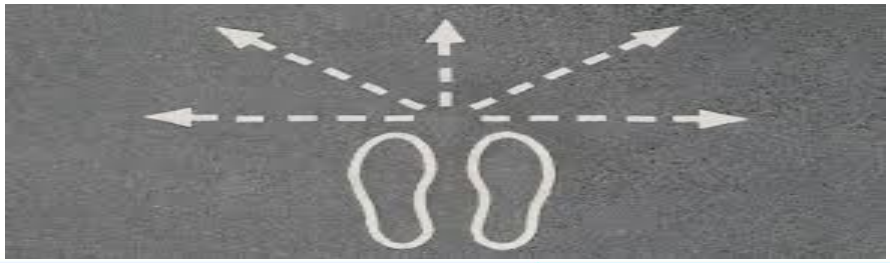
**Transición**: Pruebas finales de aceptación, Puesta en producción y Estabilización


DISCIPLINES
BUSINESS MODELLING
REQUIREMENTS
ANALYSIS & DESIGN
IMPLEMENTATION
TEST
DEPLOYMENT
CONFIGURATION & CHANGE MANAGEMENT
PROJECT MANAGEMENT
ENVIRONMENT
PHASES
Inception
Elaboration
Construction
Transition
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #3
Tran #1
Tran #2
ITERATIONS

# Resumiendo

# ¿Cómo escoger una metodología del desarrollo del software?

- ¿Cómo dividir el proyecto en etapas?
- ¿Qué tareas se deben llevar a cabo en cada etapa?
- ¿Qué restricciones deben aplicarse?
- ¿Qué técnicas y herramientas deben emplearse?
- ¿Cómo se controla y gestiona un proyecto?

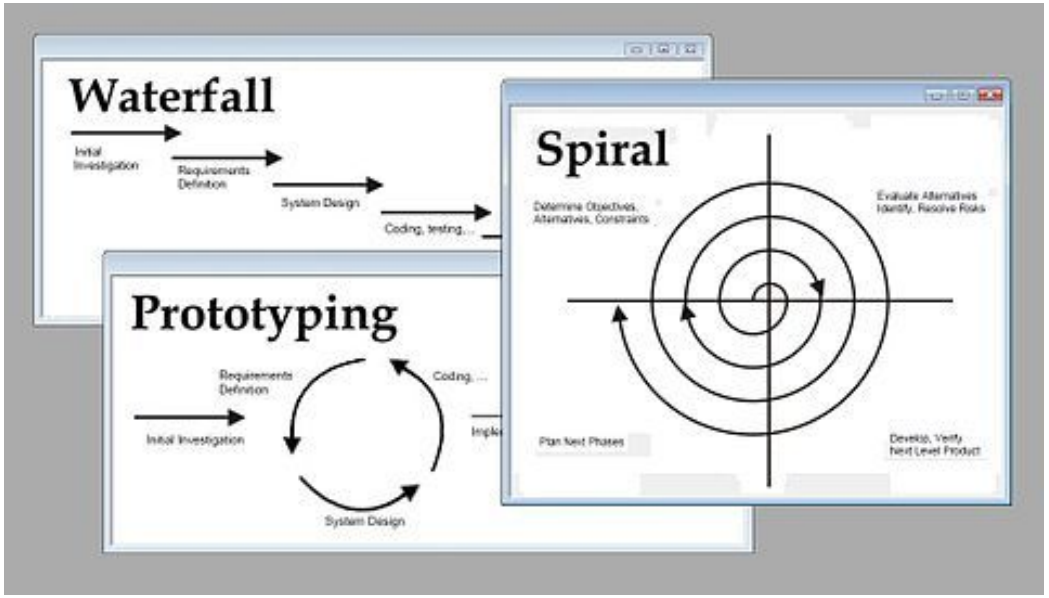


# Ninguna Metodología Garantiza el éxito de un Proyecto

- El proceso de desarrollo de software no es único.
- **No existe un proceso de desarrollo universal que sea efectivo en todos los proyectos.**
- Es difícil por ejemplo, **automatizar** todo un proceso de desarrollo de software y aún más unificar a los profesionales del área sobre la ejecución de algún tipo de proceso en particular.
- **Existen diferentes metodologías** que satisfacen las visiones principales de cómo debería asumirse el proceso de desarrollo de software.

# Criterios para elegir una metodología

- Costo/Beneficio
- Complejidad
- Robustez del software
- Conocimiento disponible
- Tipo de cliente



# Diferencias entre los modelos de proceso convencionales y ágiles

- Marcos ágiles:
  - Están basados en heurística provenientes de **prácticas de producción de código.**
  - Están **preparadas para cambios** durante el proyecto.
  - Son impuestas internamente (por el equipo).
  - **Proceso menos controlado.**

# Diferencias entre los modelos de proceso convencionales y ágiles

- Marcos ágiles:
  - No existe **contrato** tradicional.
  - Son bastante **flexibles**.
  - El cliente es parte del equipo de desarrollo.
  - **Grupos pequeños** y trabajando en el mismo sitio.
  - **Menos énfasis en la arquitectura** del software al inicio.

# Diferencias entre los modelos de proceso convencionales y ágiles

- Metodologías convencionales:
  - Basadas en **normas** provenientes de **estándares**.
  - Presentan cierta **resistencia a los cambios**.
  - **Impuestas externamente.**
  - Proceso mucho más controlado, con numerosas políticas.
  - Existe un contrato prefijado con $ y alcance fijo.

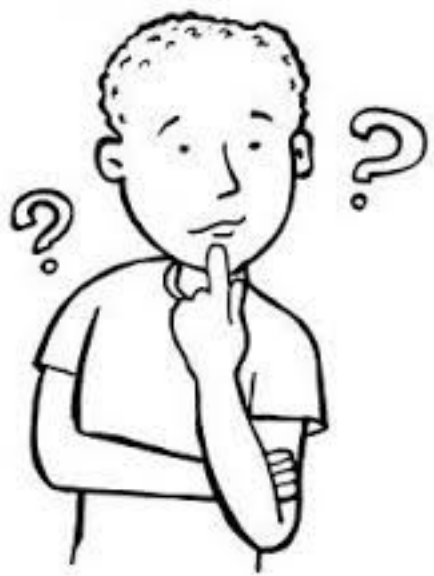
# Diferencias entre los modelos de proceso convencionales y ágiles

- Metodologías convencionales:
  - Son un poco rígidas.
  - El **cliente** interactúa con el equipo de desarrollo mediante **reuniones**.
  - **Grupos grandes y posiblemente distribuidos.**
  - ***La arquitectura del software es esencial y se expresa mediante modelos.***

# ¿Qué metodología es conveniente usar?

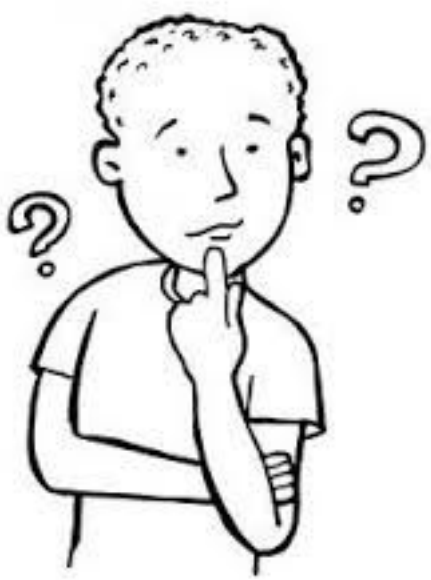
- Tener metodologías diferentes para aplicar **de acuerdo con el proyecto** que se desarrolle resulta una idea interesante.
- **En un proyecto puede involucrar prácticas tanto de marcos ágiles como de metodologías tradicionales.**

  De esta manera podríamos tener una metodología para cada proyecto, la problemática sería definir cada una de las prácticas, y en el momento preciso definir parámetros para saber cual usar.



# ¿Qué metodología es conveniente usar?

- Es importante tener en cuenta que el uso de un **marco ágil no es para todos.**
- Por otro lado, las metodologías tradicionales o convencionales permiten crear software de manera más segura ya que estas están establecidas por sus pasos.



Sistemas de Aviónica

https://www.youtube.com/watch?v=bCbaEQ00Ez4

1. Identifique detalles de este tipo de sistema.
2. Qué Metodología utilizaría para el desarrollo de este sistema?
3. Describa las etapas y que incluiría del desarrollo en cada etapa.
4. Socialice en clase.

# Conclusiones

- Una **metodología** se basa en una combinación de los **modelos de proceso** genéricos.
- La trascendencia de las metodologías se ha hecho notoria, se necesita agilizar y reducir costos.
- En el desarrollo convencional todo el programa está en un solo bloque, con ejecución secuencial de instrucciones.

# Conclusiones

- En el desarrollo estructurado los programas están divididos en distintos bloques.
- El desarrollo orientado a objetos comprende dividir un programa en clases.
- Los **marcos ágiles** fueron pensados especialmente para equipos de desarrollo pequeños, con plazos reducidos, requisitos volátiles y nuevas tecnologías.

# Referencias


- http://wiki.monagas.udo.edu.ve/index.php/Metodolog%C3%ADas_para_el_desarrollo_de_software
- Sommerville, Ian. "Software Engineering. International computer science series." (2004).
- Pressman, Roger S. *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.
- http://www.projectsmart.co.uk/docs/chaos-report.pdf
- http://es.slideshare.net/laos7/el-proceso-de-desarrollo-de-software?qid=79d8ac20-fc7a-47e1-9832-e4ae7d4d4eeb&v=default&b=&from_search=2
- http://es.slideshare.net/soreygarcia/introduccin-a-la-ingenieria-de-software-14023309