

TEMAS

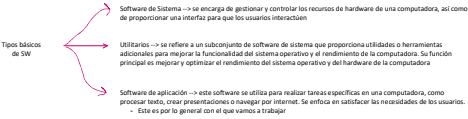
• Unidad 1 completa (incluye paper No silver bullet)

- Introducción a la Ingeniería del Software. ¿Qué es?
- Estado Actual y Antecedentes. La Crisis del Software.
- Disciplinas que conforman la Ingeniería de Software.
- Ejemplos de grandes proyectos de software fallidos y exitosos.
- Ciclos de vida (Modelos de Proceso) y su influencia en la Administración de Proyectos de Software.
- Procesos de Desarrollo Empíricos vs. Definidos.
- Ciclos de vida (Modelos de Proceso) y Procesos de Desarrollo de Software.
- Ventajas y desventajas de Cú de los ciclos de vida. Criterios para elección de ciclos de vida en función de las necesidades del proyecto y las características del producto.
- Componentes de un Proyecto de Sistemas de Información.
- Vínculo proceso-proyecto-producto en la gestión de un proyecto de desarrollo de software.

- Gestión de Producto
- Requerimientos Ágiles - User Stories
- Estimaciones Ágiles
- SCM

¿QUÉ ES EL SOFTWARE?

El software no es solo código, se tiende a hacer una mala asociación entre que el software es sólo el programa, pero es importante incorporar que el software es un set de programas y la documentación que lo acompaña. Saber programar más, no alcanza para hacer software.



SOFTWARE Y MANUFACTURA

No hay manera de comparar software con una línea de producción. Tenemos 5 razones principales:

1. El software es **menos producible**, no como los productos tangibles en una línea de producción
2. Con **mejor producto de software es igual a otro**, ya que de alguna manera los R o necesidades de los usuarios que van a hacer uso del mismo van a ser distintas
3. **No todas las fallas en el SW son errores**. El tratamiento de errores en SW es totalmente distinto en software que en la producción.
4. **El software no se gasta**. Cuando hablamos de productos, hablamos de tiempo, en el caso de este producto se desgasta. Si bien el SW debe ir adaptándose a los cambios y las necesidades del usuario (evolutivo), específicamente no tiene la calidad de durabilidad.
5. El software **«obviamente» no está gobernado por las leyes de la física**.

Evidentemente, todas las diferencias están ligadas a la intangibilidad del software como producto. A demás de estas razones hay muchas más

PROBLEMAS AL DESARROLLAR/CONSTRUIR SOFTWARE

1. Más tiempo y costo que los presupuestados
  - problema más repetido al desarrollar SW
2. La versión final del producto no satisface las necesidades del cliente
  - ¿quién es el que el cliente pidió, y no es que nosotros entendimos mal, sino que el cliente puede haberse explicado mal, puede que se haya olvidado de algo, o que cambie de opinión
3. Problema en la escalabilidad y/o adaptabilidad del software
  - agregar una funcionalidad en otra versión es casi un riesgo imposible. Muchas veces se parte de un prototipo que se va mejorando y haciendo que funcione, entonces en términos arquitectónicos es muy difícil hacer que eso escale???
4. Mala documentación
  - desactualizada y que no acompaña al software, por ende trae inconsistencias en la gestión o mantenimiento del mismo.
  - documentación es desde lo más pequeño, como un comentario en el código.
5. Mala calidad del SW
  - muchas veces el software de calidad se cree relacionado con la cantidad de testing que se realizó sobre él, pero no es así.



Estas son las premisas de las nuevas formas de trabajar—como los frameworks ágiles, scrum, lean, etc.— Como se descubrió que estos eran los factores de éxito dentro del desarrollo de SW, se buscó hacer que los mismos quedaran embudados dentro del proceso de desarrollo

Uno de los aspectos claves que tiene que ver con los R, no solo con dejarlos claros, sino también con incentivar a que el usuario tenga participación dentro del proceso para ir alcanzando los mismos, ya que la R es "interactiva" y medida que avanzamos el desarrollo del sistema. También debemos tener en cuenta la retroalimentación, algo súper importante para ir midiendo el avance del sistema y contar con información relevante para poder corregir fallas o errores y hacer mejoras

El manifiesto Ágil y metodologías como Scrum nos hablan sobre equipos capacitados y con habilidades cuya participación dentro del desarrollo del sistema sea competente.

SOFTWARE NO EXITOSO

- Podríamos decir que el software no es exitoso o resulta en costos más elevados cuando hacemos lo contrario a lo anterior.
- Tenemos R incompletos 13,1%
  - R cambiantes 8,5%
  - Cuando hay poco involucramiento del usuario 12,4%
  - Cuando no tenemos los recursos suficientes 10,6%
  - Cuando ponemos expectativas poco realistas y somos demasiado optimistas 9,3%
  - Cuando no tenemos suficiente apoyo por parte de la gerencia 8,7%



EL PROCESO DE SOFTWARE

Un proceso de software se define como un conjunto **estructurado** de actividades que, a raíz de un conjunto de entradas o inputs, producen una salida. Estas actividades varían dependiendo el tipo de sistema y la organización que lo desarrolla. El proceso debe ser explícitamente modelado o sea a ser administrado y llevado a cabo.

- Hay diferentes percepciones de lo que es un proceso de desarrollo de SW, pero el objetivo siempre va a ser obtener un producto o servicio de SW a partir de los inputs.

No solo vamos a tener como entrada los R, que nos van a definir el alcance, sino también personas, que son el principal recurso/capital que se consume en el desarrollo, energía, equipamiento, materiales, y procedimientos.

- Hay diferentes percepciones de lo que es un proceso de desarrollo de SW, pero el objetivo siempre va a ser obtener un producto o servicio de SW a partir de los inputs.

Nuestro objeto es a raíz de que todas estas entradas colaboren para generar un producto de SW que aporte valor.

IEEE -> un **proceso** es una secuencia de pasos ejecutados para un propósito dado

- esta definición no es lo suficientemente clara y completa, en general. Por lo que nos basamos en...

OMM -> un **proceso** es un conjunto de actividades, métodos, prácticas y transformaciones que se usan para desarrollar o mantener software y sus productos asociados.

Dentro de esta definición podemos vincular más conceptos

1. Procedimientos y métodos: definición de cómo vamos a llevar a cabo las actividades de desarrollo
2. Personas: con habilidades, entrenamiento y motivación
3. Herramientas y equipo

Las personas hacen uso de las herramientas, equipos, procedimientos y métodos y producen un producto de SW

¿Qué define el proceso de desarrollo de SW? La organización, en función de los objetivos y de las características del SW



PROCESOS DEFINIDOS

Estos se basan en una concepción basada en el modelo industrial (ingratos en líneas de producción), en la cual tengo un conjunto de inputs que van a ser utilizados para un conjunto de actividades, que van a producir el mismo resultado una y otra vez, siempre que los inputs sean los mismos.

Este tipo de procesos asumen que vamos a ser utilizados para un conjunto de actividades, que van a producir el mismo resultado una y otra vez, siempre que los inputs sean los mismos.

Este tipo de procesos asumen que vamos a ser utilizados para un conjunto de actividades, que van a producir el mismo resultado una y otra vez, siempre que los inputs sean los mismos.

No podemos simplificar el SW a algunos inputs, que usados para determinadas aplicaciones producen el mismo output, es decir que el output es predecible

Muchos comentarios y procesos corren en estas premisas para llevar a cabo un proyecto de SW, como por ejemplo el PUD.

PROCESOS EMPÍRICOS

Viene del empirismo, es aquella corriente en la cual centramos el foco en la experiencia. No sólo la experiencia del usuario, sino también la experiencia en el negocio la técnica, en cómo voy a manipular las herramientas y aprender de los errores.

Los tomamos en contraste con los procesos definidos

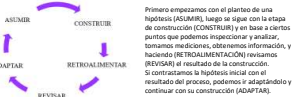
Confiamos en que las variables no pueden ser definidas o controladas en su totalidad, estamos dentro de un ambiente complejo donde guiado no todo lo que está en juego, está bajo nuestro control y esto nos implica la necesidad de adaptación para poder garantizar la producción de un software de calidad.

Se basan en la experiencia y el aprendizaje, y la principal diferencia con los procesos definidos es la forma en la cual se puede mejorar el proceso. Mientras que en un proceso definido el punto de mejora es el central (las actividades/etapas intermedias, es decir mis entradas y salidas son fijas), el proceso empírico no tiene un punto de mejora tan tangible, más bien voy a tener que ir trabajando sobre distintos aspectos, siempre centrándome en las personas, que son la clave del empirismo, son quienes capitalizan la experiencia.

Esta capitalización de lo que se ha ido se hace en un ciclo de retroalimentación que permita la adaptación y mejora **aprovechando de la experiencia**, como información, la contrasto con lo que se esperaba obtener, y en base a eso voy ir haciendo adaptaciones y mejoras.

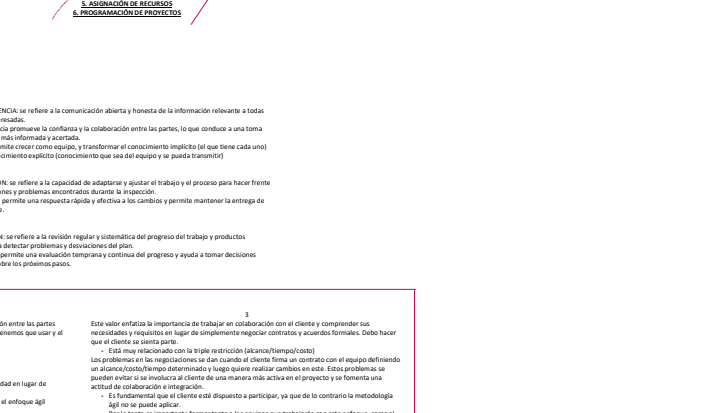
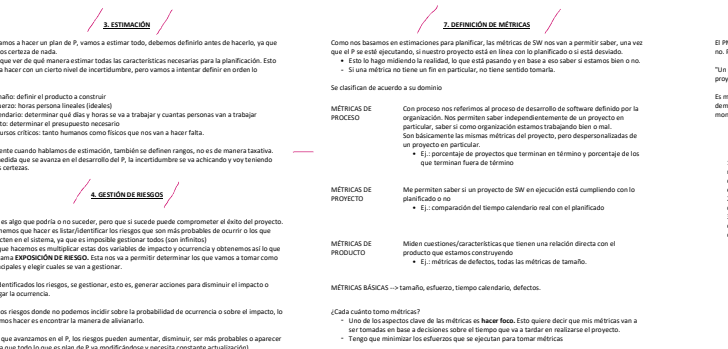
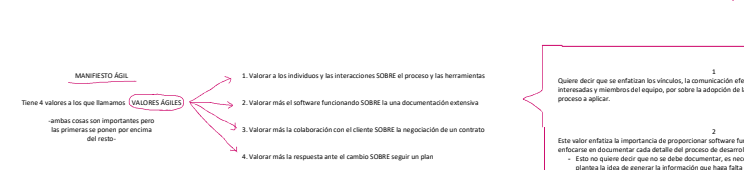
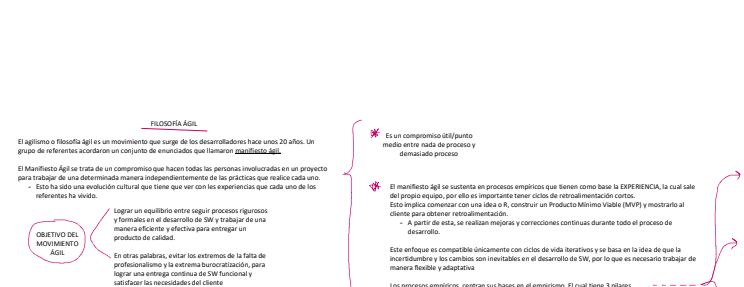
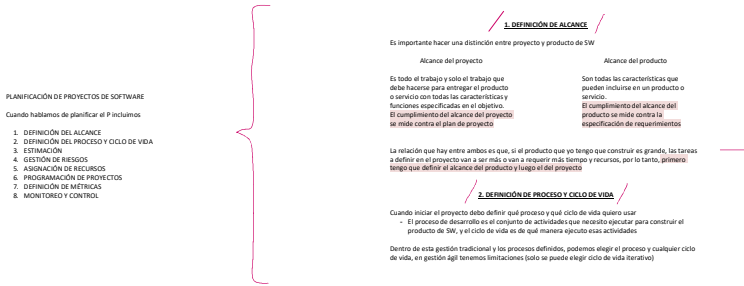
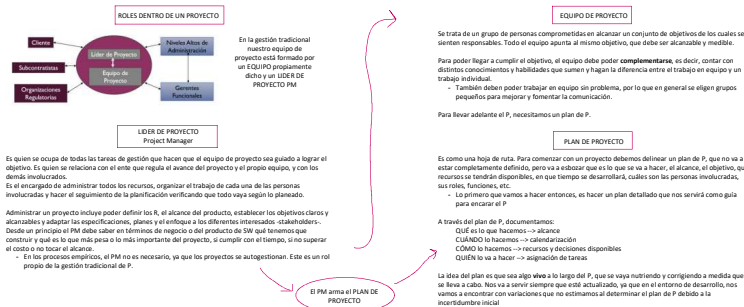
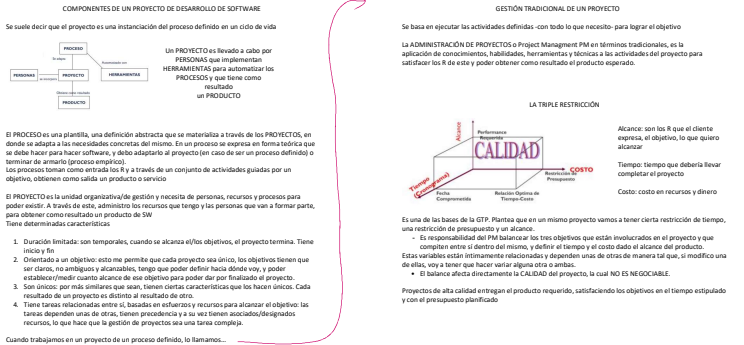
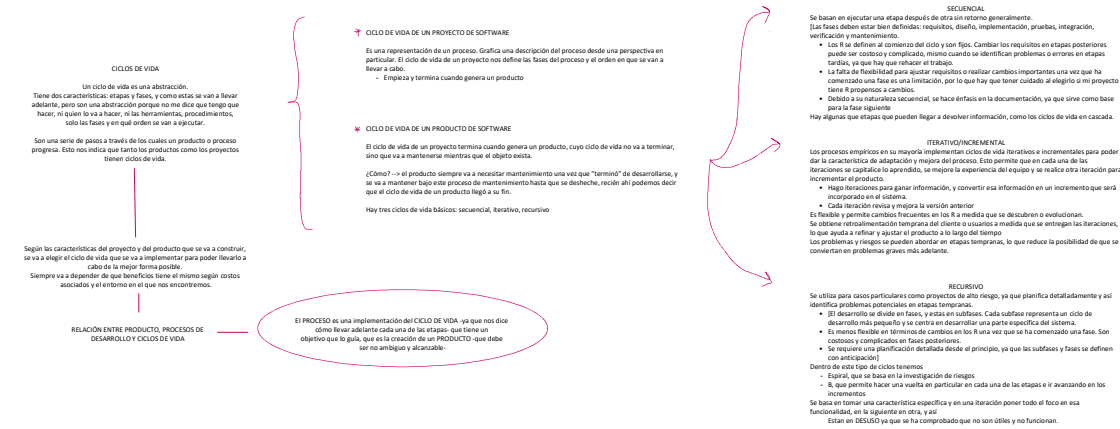
Experiencia técnica -> una de las claves para producir un producto de calidad, es necesario conocer un poco de todo.

PATRÓN DE CONDOMENTO DE PROCESOS EMPÍRICOS



Esto se basa en una concepción que necesita de repeticiones, es por esto que dentro de los procesos empíricos damos uso a un ciclo de vida específico





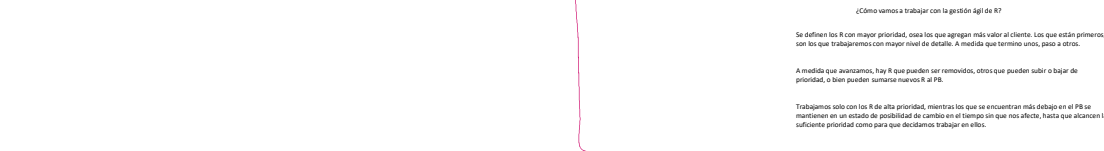
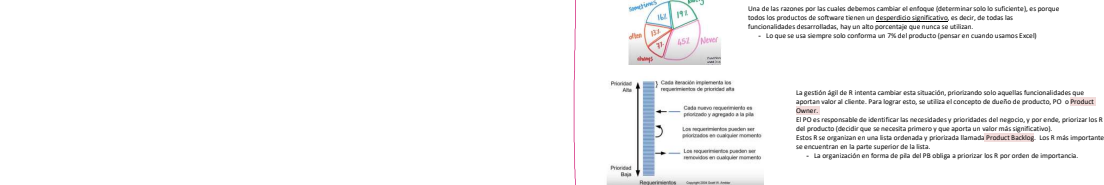
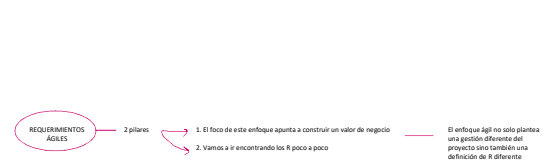
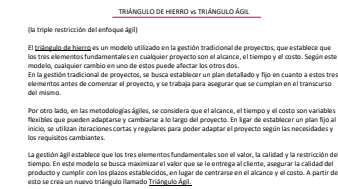
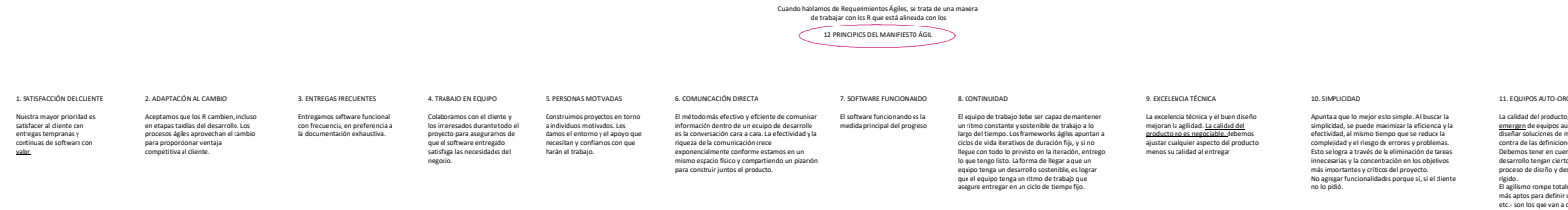
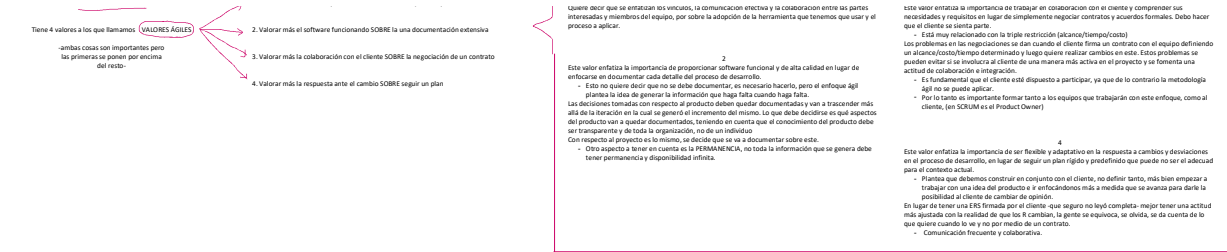
8. MONITOREO Y CONTROL

trabajar sobre lo definido en el plan de P y determinar si se está cumpliendo o no a lo largo del plan y en las métricas.

en un día a la vez" → esto quiere decir que si puedo corregir los desvíos de mi plan a lo largo del tiempo, entonces voy a poder cumplir el objetivo.

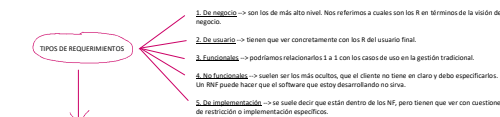
comparar lo planificado con lo real. Como desarrolladores es muy común ser demasiado optimistas al hacer estimaciones, por lo que es totalmente necesario hacer seguimiento del proyecto.

| PARA EL ÉXITO  | CAUSAS DE FRACASOS                                    |
|--|---|
| 1. Tener un plan claro y generar acciones concretas y medibles.  | 1. Falta al definir el problema                       |
| 2. Tener un plan claro y generar acciones concretas y medibles.  | 2. Plan basado en datos insuficientes                 |
| 3. Tener un plan claro y generar acciones concretas y medibles.  | 3. La planificación la hizo el grupo de planificación |
| 4. Tener un plan claro y generar acciones concretas y medibles.  | 4. No hay seguimiento del plan de P                   |
| 5. Tener un plan claro y generar acciones concretas y medibles.  | 5. Plan de P pobre en detalles                        |
| 6. Tener un plan claro y generar acciones concretas y medibles.  | 6. Planificación de recursos inadecuada               |
| 7. Tener un plan claro y generar acciones concretas y medibles.  | 7. Las estimaciones se basaron en "supuestos"         |
| 8. Tener un plan claro y generar acciones concretas y medibles.  | 8. Falta de datos históricos                          |
| 9. Tener un plan claro y generar acciones concretas y medibles.  | 9. Mala estimación de carga                           |
| 10. Tener un plan claro y generar acciones concretas y medibles. | 10. Mala comunicación                                 |



LA DIFERENCIA MÁS MARCADA ENTRE LA GESTIÓN TRADICIONAL Y LA GESTIÓN ÁGIL SE ENCUENTRA EN LA TRIPLE RESTRICCIÓN

| METODOLOGÍA TRADICIONAL vs METODOLOGÍA ÁGIL |  |
|---|--|
| Prioridad                                   | TRADICIONAL  |
| Enfoque                                     | Simple y plan  |
| Definición                                  | Enfoque "Como" (¿Por qué? ¿Para qué?)  |
| Participación                               | Detallada y controlada. Desempeño al máximo  |
| Equipos                                     | Equipos, unidades de mayor poder e interés. Análisis de negocio, Project Manager y Áreas de Proceso. |
| Herramientas                                | Entrevistas, observación y formularios   |
| Documentación                               | Alta nivel de detalle. Mapa de Rentabilidad para los departamentos.                                  |
| Productos                                   | Definidos en detalle   |
| Procesos                                    | Estables, adversos al cambio.  |



En la gestión ágil de R se trabaja a nivel de

- R de negocio
- R de usuario

Los US son el de usuario alineados a R de negocio

En este contexto, se aplican algunos CONCEPTOS RELACIONADOS A LOS PRINCIPIOS ÁGILES

- LOS CAMBIOS VAN A EXISTIR TODO EL TIEMPO DE FORMA CONSTANTE → sabemos que van a haber cambios, por lo que se busca priorizar los requerimientos orientados a poder aceptar el cambio. Si el PO, los clientes o los stakeholders se plantean cambios, es porque el producto no está siendo utilizado.
- DEBEMOS TENER UNA MIRADA DE TODOS LOS R QUE ESTÁN INVOLUCRADOS → todos los que tienen algo para decir del producto (stakeholders)
- EL USUARIO DICE LO QUE QUIERE CUANDO RECIBE LO QUE PIDió → es por esto que las iteraciones cortas y las entregas continuas son esenciales, ya que permiten una retroalimentación y un mejoramiento para futuras entregas.
- NO HAY TÉCNICAS NI HERRAMIENTAS QUE SIRVAN PARA TODOS LOS CASOS → a veces se necesita trabajar con más prototipos o modelos, otras veces con técnicas sencillas es suficiente.
- LO IMPORTANTE NO ES ENTREGAR UNA SALIDA, UN REQUERIMIENTO, SINO ENTREGAR UN RESULTADO, UNA SOLUCIÓN DE VALOR → apostamos a entregar algo útil para el negocio o el cliente minimizando el desperdicio.



ANIZADOS

es decir las mejores arquitecturas, diseños y fl.  
organizados con libertad de tomar decisiones y  
manera colaborativa. Esto de la emergencia va en  
es compacta al principio, como si uffrent.  
ta la importancia de permitir que los equipos de  
grado de autonomía y capacidad de decisión en el  
errotto en lugar de tener un enfoque paricuto y  
veniente con el enfoque tradicional y empresa que los  
aracterísticas: estimar, cuanto tiempo se necesita,  
y desarrollo.

12. MEJORA CONTINUA

El equipo reflexiona sobre como ser más  
efectivo para luego justar y perfeccionar  
su comportamiento en secuencia durante  
ciertos intervalos de tiempo

