# CICLO DE VIDA DEL SOFTWARE Cursada Año 2007

**Realizado por:**

**SAMIL LEONEL SANCHEZ ACEVEDO**

**Tecnología de programación**

**Unidad 1: Conceptos Básicos en el Desarrollo de Software**

# Trabajo Práctico Nro. 1

**Introducción a las Metodologías y Modelos de Ciclos de Vida de un Sistema**

## Lista de Conceptos Tratados:

Metodología de desarrollo de software; Modelo de ciclo de vida; Etapa; Rol; Modelo de características de un sistema de software; Tipos de sistemas de software.

## Ejercicio 1.1

Describa brevemente qué significan los siguientes términos:

1. **Metodología** de desarrollo de software.

**R//:** Es un modelo sistemático de realizar, gestionar y administrar un Proyecto para llevarlo a cabo con altas posibilidades del éxito mediante procesos a seguir sistemáticamente

1. **Modelo de Ciclo de Vida** para el desarrollo de software.

**R//:** es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso.

1. **Etapa** dentro de un ciclo de vida.

**R//:** Es una forma de verificar las fases de un producto o proyecto de software están dividas en cuatro etapas las cuales son:

* Etapa de introducción. Es el momento en que el producto se introduce en el mercado.
* Etapa de crecimiento. En esta etapa aumentan las ventas, al aumentar el interés del cliente. Los beneficios empiezan a crecer y el producto necesita mucho apoyo para mantenerse
* Etapa de madurez. El crecimiento de las ventas se ralentiza y estabiliza en el mercado. El producto está asentado y consolidado en el mercado y los beneficios son altos.
* Etapa de declive. Las ventas comienzan a decrecer significativamente y el producto se prepara para salir del mercado normalmente ya saturado. La causa principal suele ser la obsolescencia.

1. **Rol** que puede cumplir una persona en el desarrollo de software.

**R//:** En el desarrollo de software se requiere de distintas capacidades es por eso que cada persona debe tener un rol dentro del grupo, que viene dado por su experiencia y capacidades personales. Estos roles son: administrador de proyecto, analista, diseñador, programador, téster, asegurador de calidad, documentador, ingeniero de manutención, ingeniero de validación y verificación, administrador de la configuración y por último, el cliente

cada rol debe estar definido su alcance para así no tener problemas redundantes y no generar afectaciones al proyecto, esto hace que el proceso de desarrollo sea mas efectivo en la elaboración del producto

1. **Modelo/diagrama** de las características de un sistema de software y sus partes componentes.

**R//:** Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Los diagramas de componentes son más parecidos a los diagramas de casos de usos, éstos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes.

## Ejercicio 1.2

Describa brevemente en qué situaciones es imprescindible seguir una metodología, para el desarrollo de software, y en cuáles situaciones no lo sería tanto.

**R//:** Cuando el proyecto es grande y están participando varias personas se recomienda implementar una metodología ágil, cuando es pequeño y casi no hay cambios en el los requerimientos de proyecto se recomienda utilizar una metodología estructurada, la elección de la metodología se da en el levantamiento de requerimientos, los cuales nos indica que metodología podemos utilizar y si se requiere alguna metodología especifica

## Ejercicio 1.3

Enumere las categorías más comunes de metodologías existentes, para el desarrollo de software, junto con sus características principales.

**R//:**

La metodología del ciclo de vida del desarrollo de sistemas (SDLC)

* los sistemas se hayan desarrollado y documentado mediante el uso de SDLC
* sea importante documentar cada paso del proceso la administración de nivel superior se sienta más cómoda o segura si utiliza SDLC
* haya los recursos y el tiempo adecuados para completar el SDLC completo
* sea importante la comunicación en relación con la forma en que funcionan los nuevos sistemas

Metodologías ágiles

* haya un defensor de proyectos de métodos ágiles en la organización
* haya que desarrollar aplicaciones rápidamente en respuesta a un entorno dinámico
* haya que realizar un rescate (el sistema falló y no hay tiempo de averiguar qué salió mal)
* el cliente está satisfecho con las mejoras incrementales
* los ejecutivos y analistas están de acuerdo con los principios de las metodologías ágiles

Metodologías orientadas a objetos

* los problemas modelados se prestan a sí mismos para convertirlos en clases
* una organización ofrece apoyo para aprender UML
* es posible agregar sistemas en forma gradual, un subsistema a la vez
* la reutilización de software escrito con anterioridad es una posibilidad
* es aceptable hacer frente a los problemas difíciles primero

## Ejercicio 1.4

Enumere los modelos de ciclo de vida más comunes, para el desarrollo de software, junto con sus características principales.

## Un modelo de ciclo de vida define el estado de las fases a través de las cuales se mueve un proyecto de desarrollo de software.

## *Modelo cascada*

## Este es el más básico de todos los modelos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. La visión de la modelo cascada del desarrollo de software es muy simple; dice que el desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuye a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la fase. El modelo de ciclo de vida cascada captura algunos principios básicos o características:

* Planear un proyecto antes de embarcarse en él.
* Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
* Documentar los resultados de cada actividad.
* Diseñar un sistema antes de codificarlo.
* Testear un sistema después de construirlo.
* Una de las contribuciones más importantes del modelo cascada es para los administradores, posibilitándoles avanzar en el desarrollo, aunque en una escala muy bruta.

## *Modelo de desarrollo incremental*

### El desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema. Típicamente, un documento de requerimientos es escrito al capturar todos los requerimientos para el sistema completo, no demanda una forma específica de observar el desarrollo de algún otro incremento. El modelo de desarrollo incremental provee algunas características significativas para los proyectos:

* Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
* Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
* Si un error importante es realizado, sólo la última iteración necesita ser descartada.
* Reduciendo el tiempo de desarrollo de un sistema (en este caso en incremento del sistema) decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.
* Si un error importante es realizado, el incremento previo puede ser usado.
* Los errores de desarrollo realizados en un incremento pueden ser arreglados antes del comienzo del próximo incremento

## *Modelo Espiral*

## El modelo espiral de los procesos software es un modelo del ciclo de meta-vida. En este modelo, el esfuerzo de desarrollo es iterativo. Tan pronto como uno completa un esfuerzo de desarrollo, otro comienza. Además, en cada desarrollo ejecutado, puedes seguir estos cuatros pasos: Determinar qué quieres lograr. Determinar las rutas alternativas que puedes tomar para lograr estas metas. Por cada una, analizar los riesgos y resultados finales, y seleccionar la mejor, y sus principales características son:

* El modelo espiral captura algunos principios básicos o características:
* Decidir qué problema se quiere resolver antes de viajar a resolverlo.
* Examinar tus múltiples alternativas de acción y elegir una de las más convenientes.
* Evaluar qué tienes hecho y qué tienes que haber aprendido después de hacer algo.
* No ser tan ingenuo para pensar que el sistema que estás construyendo será "EL" sistema que el cliente necesita, y
* Conocer (comprender) los niveles de riesgo, que tendrás que tolerar.
* El modelo espiral no es una alternativa del modelo cascada, ellos son completamente compatible

## *Modelo de prototipos*

## El Modelo de prototipos, en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos. El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará

* Etapas: Plan rápido. Modelado, diseño rápido Construcción del Prototipo Desarrollo, entrega y retroalimentación Comunicación Entrega del desarrollo final
* 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. También 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 Se puede reutilizar el código   Inconvenientes: 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. Es frecuente que el usuario se muestre reacción a ello y 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. En aras de desarrollar rápidamente el prototipo, el desarrollador suele tomar algunas decisiones de implementación poco convenientes (por ejemplo, elegir un lenguaje de programación incorrecto porque proporcione 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

## *Modelo orientado a objetos*

## Los tipos de Modelos de ciclos de vida normalmente se basan el el análisis y diseño estructurados, pero los objetos tienen una particularidad, y es que están basados en componentes que se relacionan entre ellos a través de interfaces, o lo que es lo mismo, son más modulares y por lo tanto el trabajo se puede dividir en un conjunto de mini proyectos. Además, hoy en día la tendencia es a reducir los riesgos, y en este sentido, el ciclo de vida en cascada no proporciona muchas facilidades. Debido a todo esto, el ciclo de vida típico en una metodología de diseño orientado a objetos es iterativo e incremental.

* COMPONENTES DEL MODELO DE ESTRUCTURA DE OBJETOS El componente básico del OSM es la clase de objetos. Se distinguen tres tipos de clase: \*Objetos Entidad. \*Objetos de Interfaz. \*Objetos de Control.
* Este modelo identifica: \*las clases de objetos en la aplicación. \*como las clases de objetos se asocian unas con otras. \*como se comunican los objetos. \*Los detalles de cada clase de objetos, incluyendo atributos y operaciones. Durante el proceso de análisis y diseño, el OSM es definido en sucesivos niveles incrementales de detalle, hasta que el nivel necesario para la complementación es alcanzado
* Durante el ciclo de desarrollo se aportan los siguientes elementos al modelo: Análisis del Negocio: se reconocen objetos claves del negocio y generan las abstracciones en las clases apropiadas (objetos entidad). Análisis de Requerimientos: se identifican asociaciones estructurales entre objetos y nuevas clases (entidad). Diseño lógico: Se incorporan todas las clases necesarias para la aplicación incluyendo los objetos de interfaz y de control. Diseño Físico: se incorporan todos los detalles remanentes para la complementación física de cada clase de objetos.

## *Modelo ágil*

### El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto

## Algunos métodos ágiles de desarrollo de software: Adaptive Software Development (ASD) Agile Unified Process Crystal Clear Feature Driven Development (FDD) Lean Software Development (LSD) Kanban (desarrollo) Open Unified Process (OpenUP) Programación Extrema (XP) Método de desarrollo de sistemas dinámicos (DSDM) Scrum G300 6D-BUM

## Ejercicio 1.5

Enumere las etapas más comunes que comprenden los diferentes modelos de ciclo de vida existentes para el desarrollo de software. Describa brevemente el propósito de cada una.

**R//:**

### Cualquier sistema de información va pasando por una serie de fases a lo largo de su vida. Su ciclo de vida comprende una serie de etapas entre las que se encuentran las siguientes:

* ***Planificación***

Realizar una serie de tareas previas que influirán decisivamente en la finalización con éxito del proyecto.

* ***Análisis***

Averiguar qué es exactamente lo que tiene que hacer el sistema. La etapa de análisis en el ciclo de vida del software corresponde al proceso mediante el cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema.

* ***Diseño***

Se han de estudiar posibles alternativas de implementación para el sistema de información que hemos de construir y se ha de decidir la estructura general que tendrá el sistema (su diseño arquitectónico). El diseño de un sistema es complejo y el proceso de diseño ha de realizarse de forma iterativa.

* ***Implementación***

Seleccionar las herramientas adecuadas, un entorno de desarrollo que facilite nuestro trabajo y un lenguaje de programación apropiado para el tipo de sistema que vayamos a construir. La elección de estas herramientas dependerá en gran parte de las decisiones de diseño que hayamos tomado hasta el momento y del entorno en el que nuestro sistema deberá funcionar.

* ***Pruebas***

Tiene como objetivo detectar los errores que se hayan podido cometer en las etapas anteriores del proyecto (y, eventualmente, corregirlos). La búsqueda de errores que se realiza en la etapa de pruebas puede adaptar distintas formas, en función del contexto y de la fase del proyecto.

* ***Instalación o despliegue***

Debemos de planificar el entorno en el que el sistema debe funcionar, tanto hardware como software: equipos necesarios y su configuración física, redes de interconexión entre los equipos y de acceso a sistemas externos, sistemas operativos y bibliotecas.

Estas etapas son un reflejo del proceso que se sigue a la hora de resolver cualquier tipo de problema.

* ***Uso y mantenimiento***

La etapa de mantenimiento consume típicamente del 40 al 80 por ciento de los recursos de una empresa de desarrollo de software. De hecho, con un 60% de media, es probablemente la etapa más importante del ciclo de vida del software.

* Eliminar los defectos que se detecten durante su vida útil, lo primero que a uno se le viene a la cabeza cuando piensa en el mantenimiento de cualquier cosa.
* Adaptarlo a nuevas necesidades cuando el sistema ha de funcionar sobre una nueva versión del sistema operativo o en un entorno hardware diferente.
* Añadirle nueva funcionalidad, cuando se proponen características deseables que supondrían una mejora del sistema ya existente.
* ***Delimitación del ámbito del proyecto***

Determinar los aspectos abarcados por el proyecto como fijar aquellos aspectos que no se incluirán en el proyecto. Estos últimos han de indicarse explícitamente. Si es necesario, se puede especificar todo aquello que se posponga hasta una versión posterior del sistema.

* ***Estudio de viabilidad***

Con recursos ilimitados (tiempo y dinero), casi cualquier proyecto se podría llevar a buen puerto. Por desgracia, en la vida real los recursos son más bien escasos, por lo que no todos los proyectos son viables.

* ***Análisis de riesgos***

### Siempre se produce algún contratiempo que eche por tierra la mejor de las planificaciones. Es algo inevitable con lo que hemos de vivir y para lo cual disponemos de una herramienta extremadamente útil: la gestión de riesgos, que tradicionalmente se descompone en evaluación de riesgos y control de riesgos

## Ejercicio 1.6

Enumere los roles más comunes que puede cumplir una persona en el desarrollo de software. Describa brevemente las responsabilidades principales de cada uno.

**R//:**

Un equipo de desarrollo puede ser una sola persona, o 50, pero en cualquier equipo existen una serie de roles(funciones), que pueden ser identificados.En un equipo pequeño, puede que una persona cubra múltiples roles, mientras que, en equipos más grandes, es más común tener funciones dedicadas, independientemente del caso, la identificación de los roles en el equipo ayudará a estructurar el mismo, y a crear conciencia de las responsabilidades**.** Por ejemplo, si nadie se siente responsable de probar el software, será inevitable que se encuentren errores en la versión final y evalué cómo son tratadas dentro de su equipo las distintas responsabilidades descriptas.

* ***El Cliente***

## Se puede pensar que tratar al cliente como parte del equipo de desarrollo es extraño, pero en realidad, no lo es: El cliente es un factor importante en el éxito de un proyecto, tanto como cualquier otro miembro del equipo, por eso es importante contar con la participación del cliente dentro del proyecto.

* ***El Analista***

## El analista es alguien que es responsable de entender las necesidades del cliente, y asegurarse de que la solución que está siendo desarrollada se ajusta a esas necesidades. Las actividades típicas de un analista incluyen la elicitación de requisitos, reuniones con clientes y la redacción de especificaciones funcionales.

* ***El Arquitecto de Software***

## El papel del arquitecto de software es traducir los requisitos, tal como se define por el analista, en una solución técnica. Él puede crear un diseño técnico, o simplemente algunos bocetos a mano alzada, de cómo el sistema va a estar estructurado. En cualquier caso, es la responsabilidad del arquitecto a pensar en el sistema antes de que se desarrolle. Si se hace bien, durante la fase de diseño que se abordarán correctamente todos los problemas que se enfrenten en el desarrollo de la solución.

* ***El Arquitecto del Sistema***

## Al igual que el arquitecto de software, el Arquitecto del Sistema es responsable de pensar el sistema antes de construirlo. Así como el arquitecto de software es responsable para el software, un arquitecto del sistema es responsable del hardware. Muchas aplicaciones ejecutan completamente en un único servidor. Muchos otros sin embargo se ejecutan en grupos de servidores, con servidores dedicados de bases de datos, servidores web y balanceadores de carga. Un arquitecto del sistema tiene en cuenta los requisitos de rendimiento y disponibilidad, el número de usuarios / visitantes, etc. y en base a esto, diseña una infraestructura de servidores y una red.

* ***El Desarrollador de Software***

## El desarrollo efectivo de una aplicación es hecho por los desarrolladores del equipo. Pero un desarrollador tiene más responsabilidades que solo escribir código. Él es a menudo responsable de hacer el seguimiento de su propio progreso, e informar al jefe de proyecto de los problemas a los que se enfrenta. Él es también quien implementa las ideas del arquitecto, y como tal, puede tener que discutir las (in)posibilidades de la implementación con el arquitecto. Una responsabilidad importante es documentar el código.

* ***El Jefe de Desarrolladores***

## Un desarrollador líder, que tiene las mismas responsabilidades que los otros desarrolladores, pero también tiene añadidas algunas más. Un desarrollador líder debe entrenar a los otros desarrolladores, y ayudarles a resolver los problemas que puedan enfrentar. Este desarrollador, que suele ser el miembro del equipo más experimentado, también será capaz de asegurarse de que la ejecución sigue de cerca al diseño planteado, y no se dé lugar a lo que se denomina “invasión de características” durante el desarrollo. El desarrollador líder tiene una gran influencia en la calidad del código.

* ***El Diseñador Gráfico***

## “Lo de dentro es lo que cuenta.”, es tan cierto, como que también la percepción de los usuarios depende mucho de la mirada y la sensación que le produce una aplicación o sitio web. No importa lo buena que la aplicación sea, si la interfaz es inconsistente, se sentirá menos robusto****.****

* ***El Tester***

## Las pruebas son una parte importante para asegurar que el software funciona de la manera que debería. El papel de ‘tester’ se realiza a menudo por los desarrolladores para los aspectos técnicos y los usuarios para los aspectos funcionales. Un problema que surge de hacer a los desarrolladores probar su propio código es que, no importa lo bueno que sean, se ven influidos por la forma de su código fue creado. Cuando se prueba, se tendrá en cuenta esas mismas situaciones que ya se tuvieron en cuenta a la hora de escribirlo.

* ***El Gerente del Proyecto***

## Un gerente de proyecto tiene muchas responsabilidades. Es responsable de la planificación del proyecto, de mantener el proyecto dentro del presupuesto, y de la solución de problemas. En resumen, él resuelve cualquier problema que ponga en peligro el progreso del proyecto. Muchas de las tareas del gerente del proyecto tienen que ver con la comunicación, la comunicación al cliente sobre el progreso del proyecto y la comunicación con todos los miembros del equipo. Incluso en los proyectos de desarrollo que no cuentan con un gerente de proyecto, es conveniente asignar el rol de gerente de proyecto a alguien, para que quede claro quién es responsable de la ejecución de este.

* ***El Administrador de Cuentas***

## Si usted desarrolla proyectos para clientes, sus proyectos pueden beneficiarse de las funciones de un Administrador de Cuentas. Un administrador de cuentas cultiva la relación con el cliente. Aunque la gestión de proyectos y administración de cuentas se hace a menudo por la misma persona dentro de un proyecto, hay situaciones en las que ayuda a dividir estos roles.

* ***El Administrador de sistemas***

## El sistema en que la aplicación será instalada es creado por un administrador del sistema, se arman los servidores, se instala el sistema operativo, un servidor web, PHP, una base de datos y cualquier software adicional que se requiera. Incluso antes de que el proyecto se haya terminado, un administrador del sistema puede tener que construir los entornos de desarrollo y ambientes de prueba, más adelante en el proyecto, se ocupara de mantener los sistemas operando.

* ***El Administrador de Código***

## El Código es importante y debe ser tratado como tal, el código necesita ser gestionado. Si varios de los desarrolladores están trabajando en conjunto, el código que escriben debe integrarse en algún momento, independientemente del sistema de control de versiones utilizado. Además, cuando haya terminado, el proyecto debe ser implementado. La implementación del proyecto significa tomar el código y desplegarlo en el servidor. Aunque usualmente no hay una persona manejando esto, es importante identificar dicho rol.

* ***El Capacitador***

## Cuando un proyecto se haya completado, los usuarios pueden necesitar ser capacitados, en particular si en el proyecto se desarrollado una aplicación. No es común capacitar a los usuarios de un sitio web, pero a menudo hay un back-end que los administradores tendrán que ser aprender a usar.

## Ejercicio 1.7

Enumere los factores que influyen a la hora de elegir un modelo de ciclo de vida para el desarrollo de un sistema.

**R//:** Ningún modelo predomina sobre el otro por ello definimos el modelo que mejor se adapte a al proyecto basado en el análisis para guiarnos, en nuestra elección los factores que mas influyen en la toma de esta decisión más comunes son:

* La complejidad del problema
* El tiempo que se tiene disponible para entregar el proyecto o entregas parciales si lo requiere el cliente
* La comunicación efectiva que tiene el equipo desarrollador y el usuario
* Que certeza o incertidumbre tiene los requerimientos dados por el usuario

## Ejercicio 1.8

Considere el desarrollo de un sistema cuyo dominio de aplicación es conocido, sus objetivos y requerimientos funcionales son estables y simples de comprender desde un principio, la tecnología a utilizar ya esta predeterminada y es bien conocida por el equipo de desarrollo. ¿Qué tipo de modelo de ciclo de vida elegiría para el desarrollo de dicho sistema?.

## Ejercicio 1.9

Una vez elegido el modelo de ciclo de vida, para el desarrollo del sistema planteado en el ejercicio anterior,

¿Qué etapas escogería para dicho modelo de ciclo de vida, teniendo en cuenta que el desarrollo lo realizan una o pocas personas?.

**Metodologías de Desarrollo de Software I Trabajo Práctico Nro. 1**

## Ejercicio 1.10

Considere ahora el desarrollo de un sistema cuyo dominio de aplicación no es muy conocido por el equipo de desarrollo. En este caso, el cliente tampoco tiene muy claro qué es lo que quiere, de manera que los objetivos y requerimientos funcionales del sistema son inestables y difíciles de comprender. Además, el equipo de desarrollo va a utilizar una tecnología que le resulta completamente nueva. Discuta qué modelo de ciclo de vida es más apropiado y qué etapas se deberían utilizar para desarrollar este sistema.

## Ejercicio 1.11

Considere ahora que el dominio del sistema a desarrollar es el de Control de Tráfico Ferroviario de una gran ciudad. ¿En cuál de los tipos de sistemas que conoce ubicaría a este sistema?; ¿Qué tipo de metodología de desarrollo de software utilizaría en este caso?; ¿Por qué?; ¿Cuáles son los factores a tener en cuenta para elegir este tipo de metodologías?.

## Ejercicio 1.12

Han surgido los métodos ágiles, ¿Cuál es su objetivo/filosofía? ¿Qué problemas de desarrollo de software intentan solucionar?

## Bibliografía de Apoyo Sugerida

1. – Herramientas de Análisis y Diseño Estructurado. Apunte de la cátedra Metodologías de Desarrollo de Software I. C. Marcos y E. Belloni. DCyS, Fac. de Cs. Exactas, UNICEN. Revisión año 2003. Versión electrónica disponible en: <http://www.exa.unicen.edu.ar/catedras/metodol1/> (Sección Apuntes).
   * Ver Capítulo de Introducción.

Página 2