Guía 5.2

PROCESO DE DESARROLLO DE SOFTWARE

Renzo Falcone

Franco Turri

Tomas Pascual

Wensley Blaise

Primera Parte:

1. ¿Por qué el análisis de los requisitos es una de las tareas más complejas en el desarrollo de software?

Porque los requisitos pueden cambiar a medida que el software se vaya desarrollando, el cliente puede precisar más o menos funciones del software en cualquier momento del desarrollo de este.

1. ¿Cuáles son las ventajas de la utilización de un ciclo de vida para el desarrollo de sistemas?

Al utilizar un ciclo de vida en el desarrollo de nuestro sistema, nos permite tener una mejor administración de las etapas.

1. Explique las principales diferencias entre los ciclos de vida presentados

Ciclo de vida Cascada: Propone un desarrollo secuencial

Herramientas 4GL: Proponen un desarrollo rápido de software

Prototipo: Es desechable, interactúa con el usuario y es evolutivo

Ciclo de vida espiral: Es evolutivo e incluye el análisis de riesgo

1. Detalle los ámbitos de aplicación más adecuados para cada modelo de ciclo de vida

Ciclo de vida Cascada: Sistemas simples y pequeños, donde los requisitos sean fácilmente identificables

Herramientas 4GL: El ámbito de aplicación de esta técnica está restringido, casi exclusivamente, al software de gestión

Prototipo: En general, cualquier aplicación que presente mucha interacción con el usuario, o que necesite algoritmos que puedan construirse de manera evolutiva, yendo de lo más general a lo más específico es una buena candidata

Ciclo de vida espiral: Se adapta bien a proyectos complejos, dinámicos e innovadores.

1. Investigue algunos lenguajes de cuarta generación, explique sus características más

importantes y sus limitaciones.

Los lenguajes de cuarta generación - 4GL, son lenguajes que consisten en declaraciones similares a las declaraciones hechas en un lenguaje humano. Los lenguajes de cuarta generación se usan comúnmente en la programación de bases de datos y los ejemplos de scripts incluyen Unix Shell, Perl, PHP, Python, Ruby y SQL.

**Python**: Python es uno de los lenguajes de programación más demandados y utilizados en la actualidad. Su fundador, Guido Van Rossum, lo confeccionó con el objetivo de crear un lenguaje sencillo, polivalente y eficaz aplicable a prácticamente cualquier proyecto de desarrollo. El lenguaje de codificación de Python tiene una sintaxis muy sencilla y utiliza palabras en inglés. Al ser un lenguaje de alto nivel, las líneas de código escritas en Python también son fáciles de leer e interpretar para rápidamente entender el objeto del fragmento de código analizado.

**SQL**:

*Integridad de los datos*: Cuando hablamos de integridad de los datos, nos referimos a asegurar que los datos sean válidos, correctos y completos. SQL y las bases de datos relacionales tienen la función de preservar esta integridad.

*Lenguaje estandarizado*: Que SQL sea un lenguaje estandarizado quiere decir que se pueden desplegar implementaciones del mismo lenguaje SQL en diferentes sistemas. De esta manera se puede utilizar el mismo código para todos ellos.

*Sencillez y claridad*: SQL es un lenguaje integral desde el punto de vista conceptual, esto quiere decir que SQL es un lenguaje unificado, claro y simple, de fácil comprensión.

*Flexibilidad*: Una de las razones por las que SQL es un lenguaje tan utilizado es por su flexibilidad, dada su versatilidad a la hora de implantar soluciones y por permitir definir diferentes formas de ver los datos para satisfacer las especificaciones requeridas por parte del usuario.

1. ¿Con cuáles modelos de ciclos de vida complementaría a los lenguajes de cuarta generación?

Con las Herramientas 4GL, ya que son un conjunto muy diverso de métodos y herramientas que tienen por objeto facilitar el desarrollo de software.

1. ¿Por qué el término “mantenimiento” no es el más adecuado?

Porque cuando decimos “mantenimiento” nos estamos a refiriendo a preservar el correcto funcionamiento de algo. En este caso, nuestro objetivo es que el software sea de utilidad para el cliente y este satisfaga las acciones necesarias. También que el mismo pueda adaptarse a nuevas necesidades.

1. ¿De qué manera complementaría el ciclo de vida en cascada con prototipos?

Se podrían implementar prototipos antes de llegar a la fase final del desarrollo, y que el cliente pueda ver que los requisitos se acercan a lo que desea.

1. ¿Por qué no dejaría un prototipo como sistema final?

Porque el objetivo del prototipo es mostrarle al cliente cómo será el proyecto, el prototipo es una versión incompleta del proyecto final que tiene como objetivo darle una previsualización de lo que será la versión final al cliente o para quien se esté realizando el sistema.

Segunda Parte:

1. ¿Qué es un proceso de desarrollo de software?

Un proceso de desarrollo del software es el proceso de dividir el trabajo de [desarrollo](https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software) del software en distintas fases para mejorar el diseño, la gestión del producto, y la gestión de proyecto. Es también conocido como el ciclo de vida del desarrollo de software. La metodología puede incluir la predefinición de [entregas](https://es.wikipedia.org/wiki/Entregable) concretas y artefactos que son creados y completados por un equipo del proyecto para desarrollar o mantener una aplicación.

1. Establezca las similitudes y diferencias principales con el ciclo de vida en cascada y el ciclo de vida en espiral.

incorpora el factor de “riesgo del proyecto” al modelo de ciclo de vida, sumando las

mejores características de los modelos de cascada y de prototipo.

Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos.

Problemas de ciclo de vida en espiral:

•Falta de un proceso de guía explicito para determinar objetivos, limitaciones y

alternativas.

•La determinación del riesgo no es fácil.

•Es necesaria mucha experiencia en proyectos de software.

incorpora el factor de “riesgo del proyecto” al modelo de ciclo de vida, sumando las

mejores características de los modelos de cascada y de prototipo.

Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos.

Problemas de ciclo de vida en espiral:

•Falta de un proceso de guía explicito para determinar objetivos, limitaciones y

alternativas.

•La determinación del riesgo no es fácil.

•Es necesaria mucha experiencia en proyectos de software.

Ciclo de vida en espiral:

Incorpora el factor de “riesgo del proyecto” al modelo de ciclo de vida, sumando las

mejores características de los modelos de cascada y de prototipo.

Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos.

Problemas de ciclo de vida en espiral:

•Falta de un proceso de guía explicito para determinar objetivos, limitaciones y

alternativas.

•La determinación del riesgo no es fácil.

•Es necesaria mucha experiencia en proyectos de software.

**Ciclo de vida en espiral:**

Incorpora el factor de “riesgo del proyecto” al modelo de ciclo de vida, sumando las

mejores características de los modelos de cascada y de prototipo.

Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos.

**Problemas de ciclo de vida en espiral:**

•Falta de un proceso de guía explicito para determinar objetivos, limitaciones y

alternativas.

•La determinación del riesgo no es fácil.

•Es necesaria mucha experiencia en proyectos de software.

**Problemas del ciclo de vida en cascada:**

•Por lo general los desarrollos de sistemas no siguen un ciclo de vida estrictamente

secuencial.

•Hay dificultad para establecer inicialmente todos los requisitos del sistema.

* Los requisitos se van aclarando y refinando a lo largo de todo el proyecto.
* Hasta que se llega a la fase final del desarrollo, la codificación, no se dispone de una versión operativa del programa.

1. ¿Qué significa que el UP está “dirigido por casos de uso”, es “iterativo e incremental” y está “centrado en la arquitectura”?

**Dirigido por los casos de uso**

En el Proceso Unificado los [casos de uso](https://es.wikipedia.org/wiki/Caso_de_uso) se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o [escenarios](https://es.wikipedia.org/wiki/Escenarios) y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. El proceso dirigido por casos de uso es el rup. Nota: en UP se está Dirigido por requisitos y riesgos de acuerdo con el Libro UML 2 de ARLOW, Jim que menciona el tema.

**Iterativo e Incremental**

El Proceso Unificado es un marco de desarrollo [iterativo e incremental](https://es.wikipedia.org/wiki/Desarrollo_en_espiral) compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un *incremento* del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en [cascada](https://es.wikipedia.org/wiki/Desarrollo_en_cascada): Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

**Centrado en la arquitectura**

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios de este: electricidad, fontanería, etc.

1. ¿Cuáles son las ventajas de un proceso iterativo e incremental respecto de un proceso de desarrollo secuencial?

El desarrollo se organiza en una serie de mini proyectos de duración fija, llamados

iteraciones (2 a 6 semanas)

- El resultado de cada uno es un sistema que puede ser probado, integrado y ejecutado.

- Cada iteración incluye sus propias actividades de: análisis de requisitos, diseño,

implementación y prueba

- El sistema crece incrementalmente a lo largo del tiempo, iteración

* El desarrollo se organiza en una serie de mini proyectos de duración fija, llamados iteraciones (2 a 6 semanas)
* El resultado de cada uno es un sistema que puede ser probado, integrado y ejecutado.
* Cada iteración incluye sus propias actividades de: análisis de requisitos, diseño, implementación y prueba
* El sistema crece incrementalmente a lo largo del tiempo, iteración tras iteración.
* El resultado de cada iteración es un sistema ejecutable, pero incompleto

En general, cada iteración aborda nuevos requisitos y amplia el sistema incrementalmente

1. ¿Qué relación tiene el UP con UML?

Son distintos, pero se relacionan para poder describir un sistema de formas distintas.

Ya que UML no puede ser utilizado para programación.

1. ¿Qué son las fases? Explique brevemente las características principales (en qué pone énfasis) de:
2. inicio

Define el alcance/ objetivos del proyecto.

1. elaboración

Se obtiene la versión refinada del proyecto, la implementación iterativa, la resolución de riesgos y nuevos requisitos.

1. construcción

Esto abarca la evolución hasta convertirse en un producto listo incluyendo requisitos mínimos.

d. transición

El programa debe estar listo para ser probado, instalado y utilizado sin ningún problema. Luego se debe comenzar a pensar futuras novedades para la misma.

1. ¿Con qué ciclo de vida relacionaría las disciplinas realizadas en las fases del UP?

Se asemeja al ciclo de vida de prototipo.

1. Explique el concepto de iteración

La iteración es la repetición del programa para llegar a un objetivo

1. ¿Cuál es el resultado tangible de cada iteración?

Cada iteración puede resultar con una mejora o el encaminamiento al proyecto final.