

# DESARROLLO DE LA METODOLOGÍA IDEAL PARA DESARROLLO DE SOFTWARE

E. ESPINOSA. Escuela Politécnica del Ejército Sede Latacunga, Latacunga, Cotopaxi, Ecuador

**RESUMEN:** El trabajo que se detalla es una propuesta metodológica para desarrollar software, se ha tomado para el mismo las mejores propuestas metodológicas de los paradigmas estructurado y orientado a objetos, Web, para lo cual se ha realizado el estudio de las diferentes técnicas, métodos, modelos, así como los estándares aplicados actualmente y se ha creado la metodología denominada Ideal, la misma que será aplicada en las TICs. Está basada en el proceso de desarrollo de Software y contempla las fases de análisis, diseño, implementación y pruebas, implantación y mejora.

Como parte del trabajo, también se realiza un análisis del marco teórico relativo a los principios de la Ingeniería del Software así como el enfoque del proceso Software orientado al producto.

## 1. INTRODUCCIÓN AL PROCESO SOFTWARE

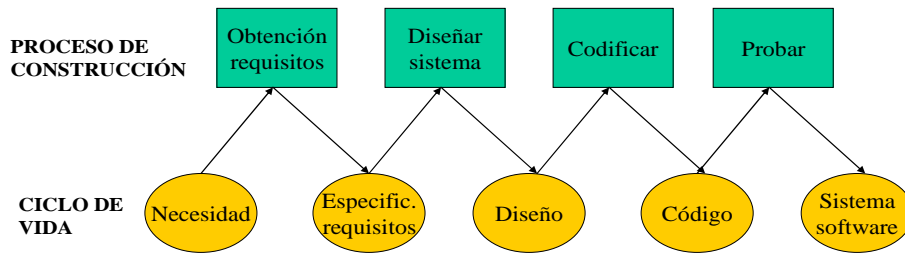
El proceso software corresponde a un conjunto de actividades que comienza con la identificación de necesidades y termina con el retiro del software.

De allí que el proceso Software básico esta formado por seis etapas definidas que son: Obtención de requisitos software, diseño, implementación, pruebas, Implantación, y Mantener y mejorar.

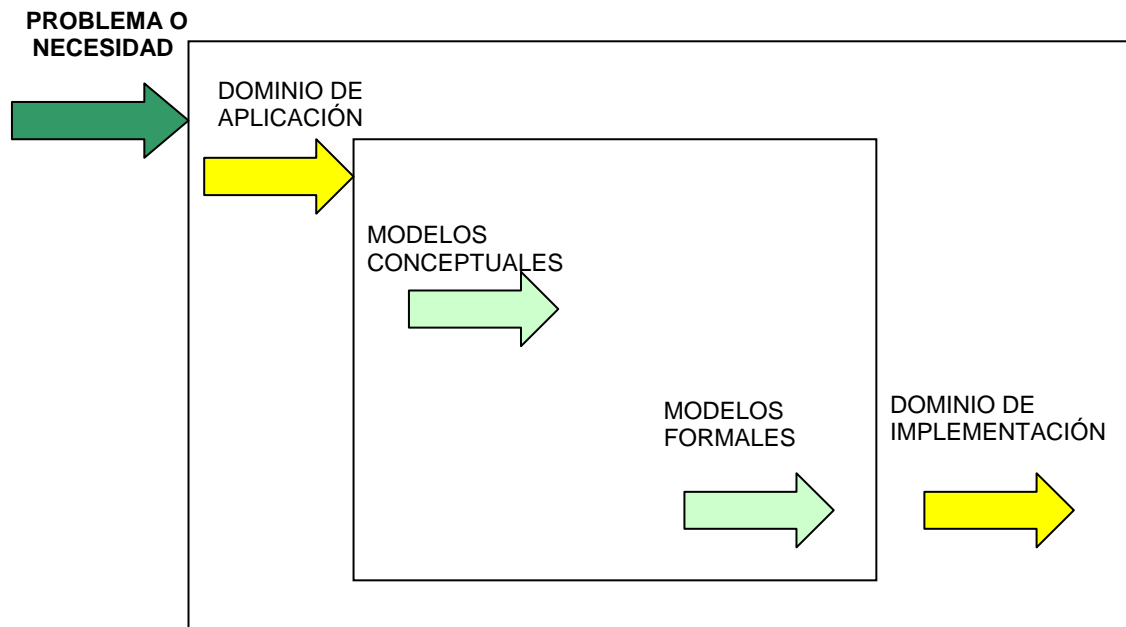
Si consideramos que el software que se obtiene al aplicar el proceso es visto como producto que sale del proceso, puede considerarse que cierta materia entra al proceso y se transforma a lo largo del mismo, generando varios productos que pueden ser documentos, diagramas, código fuente, objeto, entre otros.

Con este enfoque podemos establecer los estados por los que va pasando el producto en el proceso software, siempre haciendo énfasis que todos los productos generados en el proceso son creados por la “Mente”, que es la que transforma.

## Proceso Software vs. Ciclo de Vida



En si el Proceso Software refleja cómo se usa la experiencia humana en la construcción del software y como está se aplica en un dominio concreto



La Metodología propuesta tiene las siguientes fases que están enmarcadas dentro del proceso software, se ha propuesto para el desarrollo de la misma las siguientes:

1. Análisis de requisitos
2. Diseño del Software
3. Implementación y pruebas
4. Evolución y mejora
5. Retiro del sistemas

#### 1. Análisis de Requisitos.

<b>Nombre</b>	Análisis de Requisitos
<b>Producto de Entrada</b>	Necesidad del Cliente
<b>Objetivo</b>	Obtener el documento formal de Requisitos del Software
<b>Actores</b>	Ingeniero de Requerimientos, Analista, Cliente
<b>Producto de Salida</b>	Documento de Especificación de Requisitos

Se han establecido como actividades las siguientes:

- 1.1- Obtención del conocimiento.
- 1.2.- Modelamiento Conceptual.
- 1.3.- Validación de Requisitos.
- 1.4.- Negociación de Requisitos.
- 1.5.- Documento de Especificación de requisitos.
- 1.6.- Gestión de Requisitos

**1.1.- Obtención del conocimiento.-** En esta actividad están involucrados el cliente (Usuario final del sistema) y el ingeniero de sistemas, su principal objetivo es obtener toda la información necesaria para poder comprender el dominio del problema, para lo cual utilizaremos las técnicas de encuestas, entrevista, observación, análisis de casos entre las más utilizadas, Estas generarán un conjunto de instrumentos que son indispensables para obtener información, siempre tomando en cuenta que se deberá planificar, ejecutar, validar y actualizar en función de los objetivos que plantee la técnica, la salida de esta actividad es un documento preliminar de requisitos.

**1.2.- Modelamiento Conceptual.-** El responsable de esta actividad es el Ingeniero de Sistemas, esta actividad tiene como objetivo generar un conjunto de diagramas que permitan consolidar el conocimiento del dominio del problema.

La entrada es el documento preliminar de requisitos, en esta actividad debemos realizar tres tareas que se detalla a continuación.

**1.2.1.-** Determinar los conceptos del problema y sus relaciones.

**1.2.2.-** Diagramar el modelo conceptual.

**1.2.3.-** Diagramar casos de uso.

**1.2.1.- Determinar los conceptos del problema y sus relaciones.**

Se utiliza el siguiente instrumento que se presenta a continuación.

CONCEPTO	RELACIÓN

Para registrar la información en este instrumento, utilizamos como entrada el documento obtenido en la etapa anterior, del cual vamos registramos los conceptos y las relaciones:

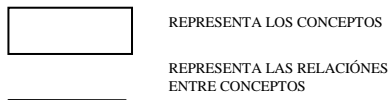
Ejemplo:

Sistema Académico

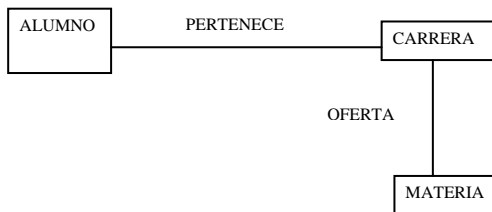
CONCEPTO	RELACIÓN
Alumno	Pertenece
Carrera	Oferta
Materia	

### 1.2.2.- Diagramar el modelo conceptual.

Una vez lleno el instrumento procedemos a crear el diagrama de conceptos, en el cual intervienen los conceptos y las relaciones registradas utilizando la siguiente simbología:

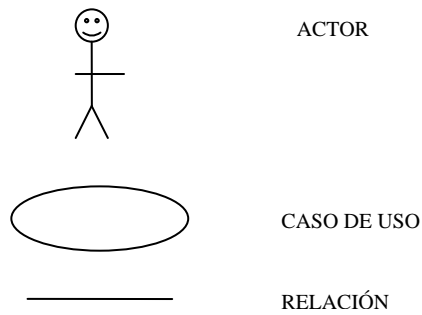


Ejemplo:

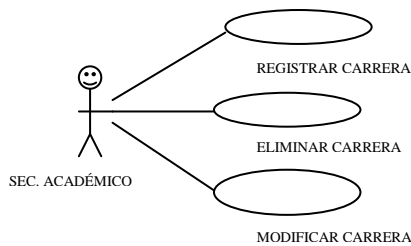


### 1.2.3.- Diagramar casos de uso.

Para realizar este diagrama utilizamos como entrada el modelo conceptual y el documento preliminar de la especificación de requisitos, los casos de uso representan las funcionalidades que en el sistema se implementará, los símbolos que se utilizan se detallan a continuación.



Ejemplo:



### 1.3.- Validación de Requisitos.

Para la actividad de validación de los requisitos de software utilizamos el modelo conceptual, el diagrama de casos de uso y el documento de especificación de requisitos de software, este proceso tiene como objetivo, verificar el grado de conocimientos obtenidos por parte del ingeniero de sistemas con el cliente, para identificar en forma clara los problemas en el documento de requisitos.

### 1.4. Negociación de Requisitos.

En todo proceso de requisitos intervienen distintos tipos de individuos (Clientes, equipo de desarrollo, equipo de gestión), etc por lo cual se crean intereses que conlleva a la generación de conflictos, los cuales se sugiere, nunca se deben resolver por decreto, sino debe ser fruto de conciliación y negociación, siempre haciendo conocer que el éxito de un buen levantamiento de requisitos permitirá crear un sistema que satisfaga las necesidades y que este cumpla todas las expectativas planteadas por el cliente.

### 1.5 Documento de Especificación de requisitos (Documento formal IEEE 830).

En esta actividad se genera el “Documento de Requisitos Formal”, en el modo habitual de guardar y comunicar los requisitos. Una guía para digitalizar y presentar requisitos es la aplicada según los estándares IEEE 830, IEEE 1362.

### 1.6 Gestión de Requisitos.

Esta actividad como todas las anteriores es muy importante, aquí nos centramos en administrar los cambios de los requisitos, permitiendo asegurar su consistencia trazabilidad.

## MATRIZ DE GESTIÓN DE REQUISITOS

	4	Gestión de Alumnos Nuevos		
Número	4			
Nombre	Gestión de Alumnos Nuevos			
Origen	Secretaría Académica			
Estabilidad	Alta	Media	Baja	
	x			
Prioridad	Primario	Secundario	Opcional	
	x			
Fecha de Creación	18/05/00			
Fecha de Cambio				

## 2. DISEÑO DEL SOFTWARE

Para esta fase se requiere realizar:

### 2.1 Diseño de Alto Nivel

- Diagrama Conceptual
- Definición de Casos de Uso de alto nivel
- Diagrama de Secuencia de Iteración con el sistema

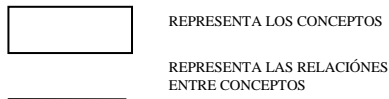
### 2.2 Diseño de la Base de Datos

<b>Nombre</b>	Diseño
<b>Producto de Entrada</b>	Especificación de Requisitos Software
<b>Objetivo</b>	Obtener el documento de Diseño del Software (Alto y Bajo Nivel)
<b>Actores</b>	Ingeniero de Software, Cliente
<b>Producto de Salida</b>	Documento de Diseño de Software

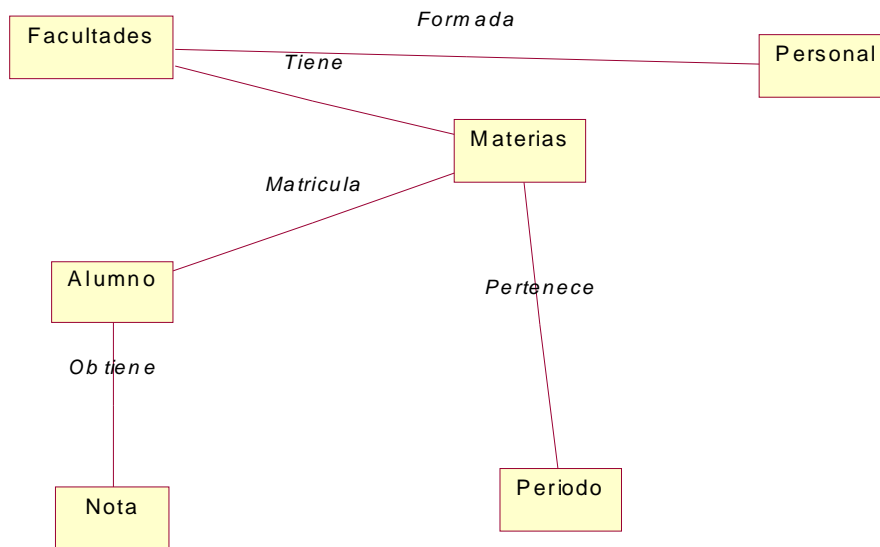
### 2.1 DISEÑO DE ALTO NIVEL

#### Diagramar el modelo conceptual.

En el diagrama de conceptual, se registra los conceptos y las relaciones utilizando la siguiente simbología:



Ejemplo:



## CASOS DE USO ALTO NIVEL

### Gestión Facultad

**Caso de uso** : Ingresar facultad

**Actores** : Secretario/secretario

**Tipo** : Primario

**Descripción** : Secretario solicita la operación de ingresar Facultad, el sistema despliega pantalla para ingreso de Facultad, el secretario ingresa los datos uno a uno solicitados en pantalla.

### CASO DE USO EXPANDIDO

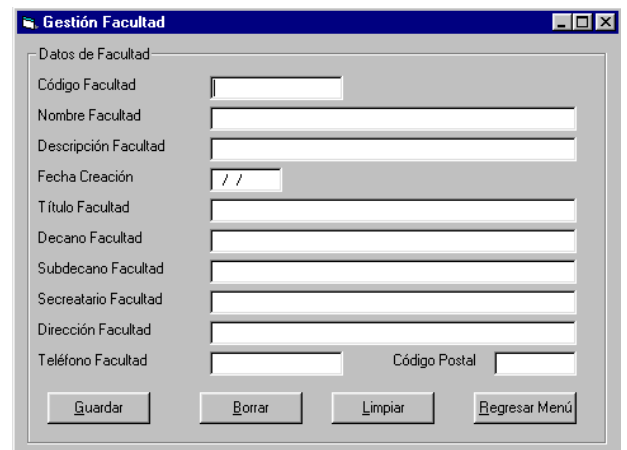
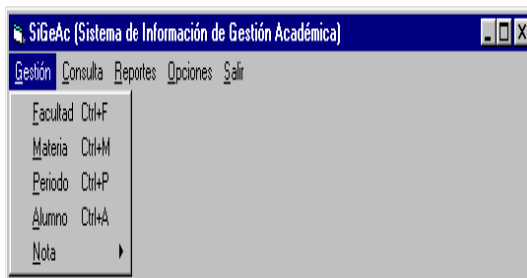
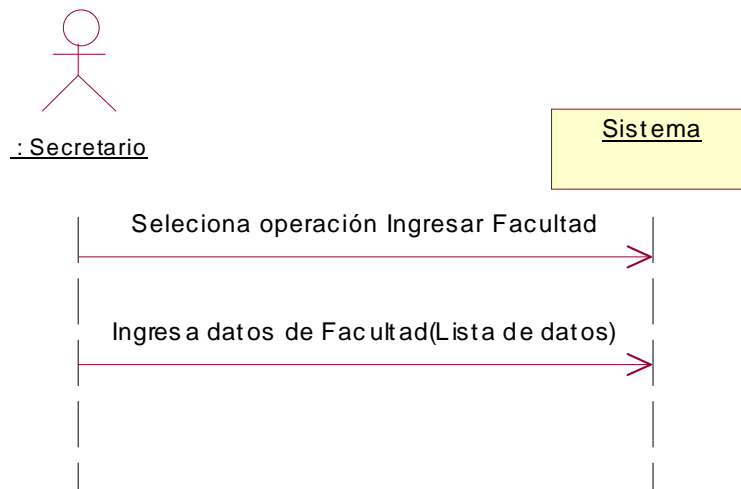
#### Caso de Uso: Ingresar Facultad

- Actores: Secretario (Iniciador)
- Propósito: Ingresar una facultad en el sistema.
- Visión general: El usuario solicita la operación de ingresar facultad, el sistema presenta el formulario de ingreso de facultad, el usuario ingresa uno a uno los datos de la facultad, sistema almacena los datos y confirma la operación.
- Tipo: Primario, esencial.
- Referencias:
- Curso Típico de eventos

Acción del Actor	Respuesta del Sistema
1. Este caso de uso comienza cuando un usuario solicita la operación de ingresar facultad	2. Presenta formulario de ingreso de facultad
3. Ingresa uno a uno los datos de la facultad.	4. Almacena la facultad y confirma la operación.

- Cursos alternativos
  - Línea 2: No existe formulario, termina el caso de uso.
  - Línea 4: Existe facultad termina caso de uso.

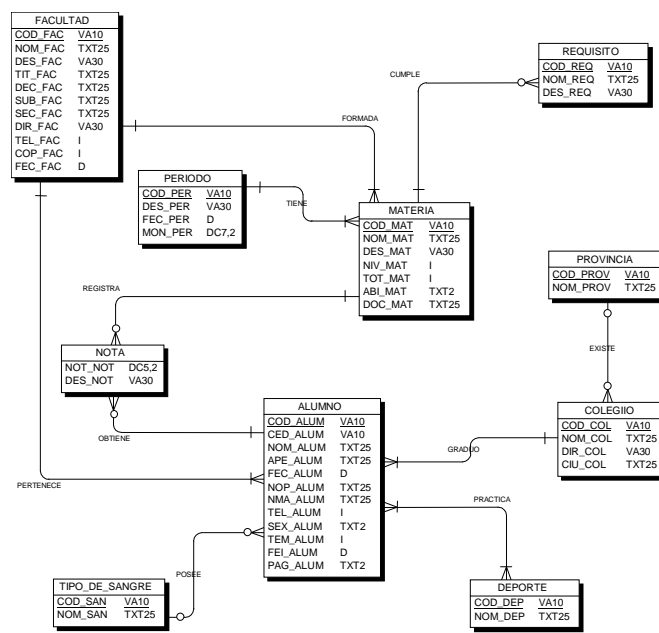
### DIAGRAMA DE SECUENCIA DE ITERACIÓN CON EL SISTEMA



## 2.2 DISEÑO DE BASE DE DATOS

En esta actividad se recogen todos los conceptos (Modelo Conceptual), para ser tratada como entidades de datos que van ser procesadas o consultadas por el sistema, todas las relaciones existentes entre las entidades.

<b>Nombre</b>	Diseño de la Base de Datos
<b>Producto de Entrada</b>	Especificación de Requisitos de Software
<b>Objetivo</b>	Obtener el documento de Diseño de la Base de Datos
<b>Actores</b>	Ingeniero de Requerimientos, Analista
<b>Producto de Salida</b>	Documento de Diseño de Base de Datos



## 3. IMPLEMENTACIÓN Y PRUEBAS

Esta fase tiene como objetivo la construcción y las pruebas de los componentes del sistema, se deberá dar a conocer el lenguaje, el entorno y el desarrollo de pruebas.

<b>Nombre</b>	Implementación y Pruebas
<b>Producto de Entrada</b>	Especificación de Requisitos de Software, Casos de Uso expandidos, Diseño de la Base de Datos, Diagrama de Secuencia de Iteración con el sistema, diseño de pruebas, normas de Programación, Lista de chequeo para programas e interfaces
<b>Objetivo</b>	Obtener el código y aplicar los diseños de prueba
<b>Actores</b>	Ingeniero de Requerimientos, Ingeniero de Pruebas, Programador
<b>Producto de Salida</b>	Código Fuente, Objeto, Documento de resultado de las pruebas

#### 4. EVOLUCIÓN/MEJORA

Durante el ciclo de vida de un software, se cambia para corregir errores de requisitos del sistema original y para implementar nuevos requerimientos que surgen, estos cambios propuestos tienen que analizarse y tienen que estar en coordinación con la gestión de configuración del software.

<b>Nombre</b>	<b>EVOLUCIÓN/MEJORA</b>
<b>Producto de Entrada</b>	Documentos de Gestión de Configuración del Software, Riego y calidad, productos del proceso software sujetos a modificación
<b>Objetivo</b>	Implantar nuevas funcionalidades al sistema
<b>Actores</b>	Administrador del proyecto, personal encargado de los cambios
<b>Producto de Salida</b>	Productos del proceso software modificados

#### 5. RETIRO

El objetivo de esta fase es preparar el entorno de retiro del software que cumplió su ciclo de vida para poner fuera de servicio.

<b>Nombre</b>	<b>RETIRO DEL SISTEMA</b>
<b>Producto de Entrada</b>	Productos del Proceso software
<b>Objetivo</b>	Desmantelar el software
<b>Actores</b>	Administrador del proyecto, personal encargado del retiro
<b>Producto de Salida</b>	Cronograma de retiro del programa

### 3 CONCLUSIONES

- La adopción de una metodología de desarrollo y la aplicación de misma, permite la construcción del producto y es una guía permanente que permite la planificación, ejecución y control durante las diferentes fases del proyecto.
- Es posible el combinar los diferentes paradigmas para generar una metodología que ayude al proceso de desarrollo de software.
- La Metodología desarrollada permite la construcción de software haciendo énfasis en revisiones.

### 5. RECOMENDACIONES

- Aplicar la metodología desarrollada en la construcción del Software.

### 6. BIBLIOGRAFÍA

- [1] F.P.J.r. Brooks. The Mythical Man-Month. Addison-Wesley, 1975.
- [2] A. Davis. Software Requirements. Objects, Functions and States. Prentice-Hall, 1993.

- [3] M. Jackson. Software Requirements and Specifications: A Lexicon of Practice. Addison-Wesley, 1995.
- [4] IAN SOMMERVILLE. Ingeniería del Software, 2004