CAPITULO 3

3. DESARROLLO DEL PROYECTO

Una vez se ha considerado una idea como viable, se comienza con la fase de desarrollo del proyecto. Como ya se mencionó en el capítulo 1, en esta fase se lleva a cabo todo lo referente a la planificación del proyecto y programación de sus actividades, actividades fundamentales para el éxito del proyecto. Una mala planificación garantiza una mala programación y un desenlace desastroso del proyecto.

¿*Pero que se considera un desenlace desastroso de un proyecto*? La respuesta es bastante sencilla: que no se entrega a tiempo y sus costos son mucho mayores de los estimados. Aunque el anterior es el peor de los panoramas posibles, una mala planificación garantiza que esto se cumpla. Cuando se elabora un proyecto, el cliente o la gerencia siempre estarán preocupadas por dos cosas: *¿para cuando estará terminado?* y *¿cuánto costará todo?*, respuestas que deberán quedar claramente respondidas cuando el equipo de proyecto termine esta etapa.

3.1. Planificación del proyecto

En esta primera parte de la fase de desarrollo se definen las actividades que se deben llevar a cabo, los recursos necesarios y la estructura de administración que tendrá el proyecto. Para lo anterior, el primer paso debe ser establecer las características propias del proyecto y restricciones propias y con base en esto definir las actividades necesarias para que se lleve a cabo.

Para poder llevar a cabo lo anterior, es importante *definir claramente el problema*, y una vez identificado plantear una serie de objetivos que permitan llegar a una solución satisfactoria. ¿*Por qué se habla de problema y no de proyecto*? Es un asunto metodológico. Al decidir llevar a cabo un proyecto las personas que toman decisiones en él se enfrentarán a una serie de necesidades y limitaciones, las cuales implican establecer un plan que permita solucionarlas y satisfacerlas y, por tanto, se las puede tratar como un problema.

Dicho lo anterior, podríamos definir que un problema existe cuando hay tres elementos, cada uno claramente definido:

- Una situación inicial. La propuesta de un proyecto. ¿De qué se dispone y qué se necesita?

- Una situación u objetivo final a ser alcanzado.

- Restricciones o pautas respecto de métodos, actividades, tipos de operaciones, etc. Sobre los cuales hay acuerdos previos.

Resolver un problema implica realizar tareas que demandan procesos de razonamientos más o menos complejos y no simplemente una actividad asociativa y rutinaria.

En todo proceso de decisiones se hace sumamente importante definir muy claramente cuál es el problema de decisión. En la informática es común que los clientes sepan en general qué desean... pero no sean capaces de concretarlo. Es común que los clientes hagan una solicitud del desarrollo de un software para una función determinada pero no saben definir los procesos dentro de él con claridad y por tanto no definen claramente los alcances de lo que desean.

Dado lo anterior, para la definición del problema el primer paso será el definir con el cliente los requerimientos detallados del proyecto y sus funcionalidades y establecer una documentación final con las especificaciones del proyecto. En el caso de un proyecto de software se trata de las especificaciones del sistema solicitado. Es muy importante que esta documentación esté cerrada de forma definitiva en el momento de definir actividades a desarrollar debido a que cambios futuros y frecuentes de estos requerimientos afectan las estimaciones iniciales de tiempo y recursos (esto es cierto en las metodología “pesadas” de desarrollo, no así en las “ágiles”).

Definidas las especificaciones, el siguiente paso es definir los objetivos y metas del proyecto. El plantear objetivos, de tipo general y especifico, permite dimensionar donde, cuando y como se quiere el trabajo a realizar. Una vez establecidos los objetivos se deben definir las metas o pasos a cumplir para llegar a dichos objetivos. Los

objetivos y metas son la base con la que se comienza a determinar las actividades necesarias par a llevar a cabo el proyecto, así como los tiempos necesarios.

3.1.1. Work Breakdown Structure

Esta es una herramienta que permite la definición de estas actividades, ajustándose a los objetivos y metas planteados, es conocida como **Estructura de Descomposición del Trabajo** o **Estructura de Desglose del Trabajo (EDT)** o también en inglés **Work Breakdown Structure, WBS.** Se trata de una herramienta para el tratamiento de problemas complejos en el contexto de la planificación o basada en la estrategia de "*descomposición jerárquica*" de la complejidad inicial, entendiendo complejidad inicial como una gran actividad general que se irá descomponiendo en partes que al cumplirse darán por desarrollada la actividad. Para lograr establecer la WBS del proyecto, se tienen que organizar las ideas alrededor de lo que se pretende hacer en el proyecto, es decir, definir un título del proyecto que esté de acuerdo con las metas y objetivos que se pretenden alcanzar.

El WBS ayuda a realizar el desglose en las tareas en que se descomponen las actividades y procesos de forma clara y fácil de entender, planificada en el tiempo y también permite que se puedan identificar recursos materiales y estimar el nivel de asignación necesaria en cada fase del desarrollo o ejecución y finalmente identificar y estimar actividades humanas y dotación de recursos humanos para las distintas tareas a desarrollar. También permite asignar responsabilidades sobre partes del proyecto asegurando que se incluyen en el plan todas las tareas necesarias sin duplicar trabajo, organizando el flujo de trabajo y establecer las bases para controlar el avance del trabajo en referencia a un plan maestro, algo de lo que se hablará más en detalle en el capitulo de control.

Para elaborar un WBS es importante que se tengan en cuenta estos aspectos:

- Que la WBS cubra todo el ámbito abarcado por el proyecto. Las tareas no incluidas no forman parte del proyecto o plan.

- La WBS debe incluir todos los resultados finales

- La suma de los elementos de cada nivel es igual ala totalidad del siguiente nivel superior (y la suma de los costes del nivel inferior iguala al coste del nivel-padre superior). Y viceversa: el trabajo de cada elemento de la estructura equivale a la suma de las tareas de los elementos subordinados.

- La articulación de elementos subdividiendo tareas debe seguir una lógica en que se refleje claramente la naturaleza del producto, servicio o sistema

- Cada elemento de la WBS debería representar un elemento discreto de trabajo que será descrito en el diccionario WBS.

- Cada elemento de la WBS debe poseer un único identificador.

- La descripción de los elementos de la WBS deberá utilizar nombres (aunque y de ser necesario se emplearán adjetivos como modificadores.

- El trabajo en cada elemento de WBS podrá ser descrito detalladamente en un diccionario o glosario propio del plan o proyecto que serán de suma utilidad para la documentación del proyecto y posteriores documentos.

- El desarrollo de una WBS debería incluir los proveedores o socios en producción, clientela, consultoras, etc.

Algo importante de recordar es que la WBS documenta el alcance del proyecto, no su plan de ejecución.

A continuación un ejemplo de una WBS para la elaboración de un proyecto de software contable:

1. Definición de las actividades:

0. Proyecto Contabilidad.

1. Especificar necesidades.

    1.1. Estudiar Sistema Actual.

    1.2. Añadir Nuevas Características.

2. Analizar Contabilidad.

    2.1 .Estudiar Procesos.

    2.2. Estudiar Datos.

3. Diseñar Aplicación.

    3.1. Diseño B.D.

    3.2. Diseño Programas.

4. Codificación.

    4.1. Construcción del esquema.

    4.2. Codificación de los Programas

5. Pruebas

    5.1. Prueba de Unidades

    5.2. Prueba del Sistema

2. El diagrama asociado a el sería como el que se puede ver en la figura 1:

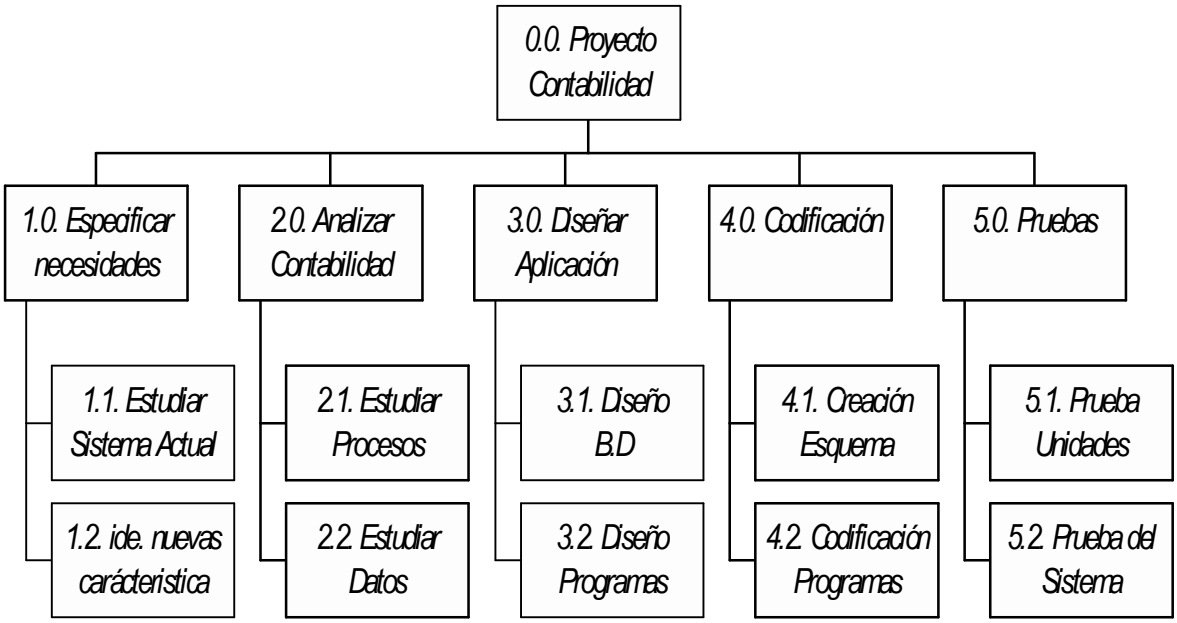

Figura 1. Diagrama WBS proyecto contable

3. Desarrollo de fichas detalladas en las que se describen detalladamente las especificaciones de cada tarea. Un ejemplo:

Especificación de tarea
Número: 3.1.
Nombre: Diseño B.D.
Descripción: Se diseñara la base de datos, partiendo del modelo entidad-relación propuesto en el análisis y con el objetivo de tener un sistema funcionando sobre DB2.
Esfuerzo Estimado: 2 semanas/hombre
Entregables: Estructura de implementación de la Base de Datos.

3.1.2. Ciclo del vida del software

Dado que este libro esta enfocado a los proyectos de software, es importante hablar de el ciclo de vida del software, dado que el ciclo que se escoja en el proceso de desarrollo del software será determinante en la forma de trabajar y documentar y por tanto en la definición de actividades. El apartado anterior habla de WBS para establecer una forma de organizar las actividades pero no de establecerlas.

En el desarrollo de software hablamos de varios formas o ciclos de vida, las más comunes son las que se mencionarán a continuación, donde la decisión de con cual trabajar dependerá de muchos factores (esto ya es dominio de la ingeniería de software).

3.1.3. Definición de ciclo de vida

El término **ciclo de vida del software** describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de establecer un ciclo de vida es el definir las distintas fases intermedias que se requieren para **validar** el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y **verificación** de los procedimientos de desarrollo. Se asegura de que los métodos utilizados son apropiados.

Por tanto, trabajar con un ciclo de vida permite:

- La organización, planificación y gestión de personal, corrección de errores, distribución del tiempo.

- Establecer la documentación que debe presentarse a los clientes así como documentar información necesaria para los usuarios del software y para el mantenimiento y futuros desarrollos.

- Determinar las herramientas y metodologías en las distintas actividades.

- La elaboración de software de más calidad. Durante el desarrollo del software es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación. El ciclo de vida permite que los errores se detecten lo antes

posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos establecidos previamente.

- ♣ Permite planificar el trabajo, por la definición misma de las actividades necesarias.

Todos los ciclos de vida, sin importar su forma tienen las siguientes partes:

3.1.3.1. Definición de necesidades iniciales:

Esta etapa tiene como objetivo la consecución de un primer documento en que queden reflejados los requerimientos y funcionalidades que ofrecerá al usuario del sistema a desarrollar, en otras palabras, qué - y no cómo - se va a desarrollar.

Dado que normalmente se trata de necesidades del cliente para el que se creará la aplicación, el documento resultante suele tener como origen una serie de entrevistas cliente-proveedor, definición de especificaciones y funcionalidades, situadas en el contexto de una relación comercial, siendo que debe ser comprendido por ambas partes.

3.1.3.2. Definición de especificaciones

Ahora se trata de formalizar los requerimientos; el documento obtenido en la etapa anterior se tomará como punto de partida para esta fase. Su contenido es aún insuficiente y lleno de imprecisiones que será necesario completar y depurar.

Por medio de esta etapa se obtendrá un nuevo documento que definirá con más precisión el sistema requerido por el cliente, el empleo de los casos de uso (*use cases*) de Jacobson es una buena elección para llevar a cabo la especificación del sistema.

Es normal que no resulte posible obtener una buena especificación del sistema en el primer intento; serán necesarias sucesivas versiones del documento donde quedará reflejada la evolución de las necesidades del cliente, dado que no siempre sabe con certeza en los primeros contactos todo lo que quiere realmente, y pueden surgir cambios

externos que supongan requerimientos nuevos o modificaciones de los ya contemplados.

3.1.3.3. Análisis

Es necesario determinar qué elementos intervienen en el sistema a desarrollar, así como su estructura, relaciones, evolución en el tiempo, detalle de sus funcionalidades, lo que permitirá una descripción clara de qué sistema se va a construir, qué funcionalidades va a aportar y qué comportamiento va a tener

3.1.3.4. Diseño

Tras la etapa anterior se conoce qué debe hacer el sistema. Ahora es necesario determinar cómo hacerlo, establecer cómo debe ser construido el sistema, definiendo en detalle entidades y relaciones de las bases de datos. Se pasará de casos de uso esenciales a su definición como casos expandidos reales, se seleccionará el lenguaje más adecuado, el motor de bases de datos a utilizar en su caso, librerías, configuraciones hardware, redes, etc.

3.1.3.5. Implementación

Llegado este punto se empiezan a codificar los algoritmos y estructuras de datos definidos en las etapas anteriores, en el correspondiente lenguaje de programación y para un determinado sistema gestor de bases de datos. Se trata de la fase de programación propiamente dicha.

3.1.3.6. Pruebas

El objetivo de estas pruebas es garantizar que el sistema haya sido desarrollado correctamente, sin errores de diseño y/o programación. Es conveniente que sean planteadas tanto a nivel de cada módulo como de integración del sistema. Existen pruebas de caja negra y de caja blanca.

3.1.3.7. Validación

Esta etapa tiene como objetivo la verificación de que el sistema desarrollado cumple con los requisitos expresados inicialmente por el cliente y que han dado lugar al

presente proyecto. Para esta fase también es interesante contar con los casos de uso, generados a través de las correspondientes fases previas, que servirán de guía para la verificación de que el sistema cumple con lo descrito por los casos.

3.1.3.8. Mantenimiento y evolución

Finalmente la aplicación resultante se encuentra en fase de producción, en funcionamiento dentro de los procesos de negocio del cliente, cumpliendo los objetivos para los que ha sido creada. A partir de este momento nos adentramos en la etapa de mantenimiento, que supondrá pequeñas operaciones tanto de corrección como de mejora de la aplicación, así como otras de mayor importancia, fruto de la propia evolución del sistema y del negocio.

La mayoría de las veces en que se desarrolla una nueva aplicación, se piensa solamente en el ciclo de vida para su creación, olvidando la posibilidad de que esta deba sufrir modificaciones futuras, que se producirán con casi completa seguridad en la mayor parte de los casos, solo que no confundir estos posibles cambios futuros con los cambios durante el desarrollo. Cuando se habla de cambios en el desarrollo implica que los requerimientos no quedaron bien definidos en la etapa inicial generando traumas en el desarrollo del proyecto, y en caso de que estos cambios sean necesarios u obligatorios, siempre deberá hacerse una re-planificación del proyecto.

3.1.4. Tipos de ciclo de vida

3.1.4.1. Modelo Cascada

Se trata de un ciclo de vida inicialmente propuesto por Royce en 1970 y fue adaptado para el software a partir de ciclos de vida de otras ramas de la ingeniería. Es el primero de los propuestos y el más ampliamente seguido por las organizaciones (se estima que el 90% de los sistemas han sido desarrollados así). Algunas características de este modelo son:

- El producto final tarde mucho tiempo, El cliente no verá resultados hasta el final, con lo que puede impacientarse. Es comparativamente más lento que los demás y el coste es mayor también.

- Las etapas se realizan de manera lineal.

- Es rígido y restrictivo, poco flexible. Si se han cometido errores en una fase es difícil volver atrás.

- No se tiene el producto hasta el final, esto quiere decir que si se comete un error en la fase de análisis no lo descubrimos hasta la entrega, con el consiguiente gasto inútil de recursos.

- Se deben tener claros todos los requerimientos al principio, por eso es útil en proyectos complejos que se entienden bien desde el principio o aquellos para los que se dispone de todas las especificaciones desde el principio.

- Sirve como base al resto de modelos.

La estructura se muestra en la figura 2:

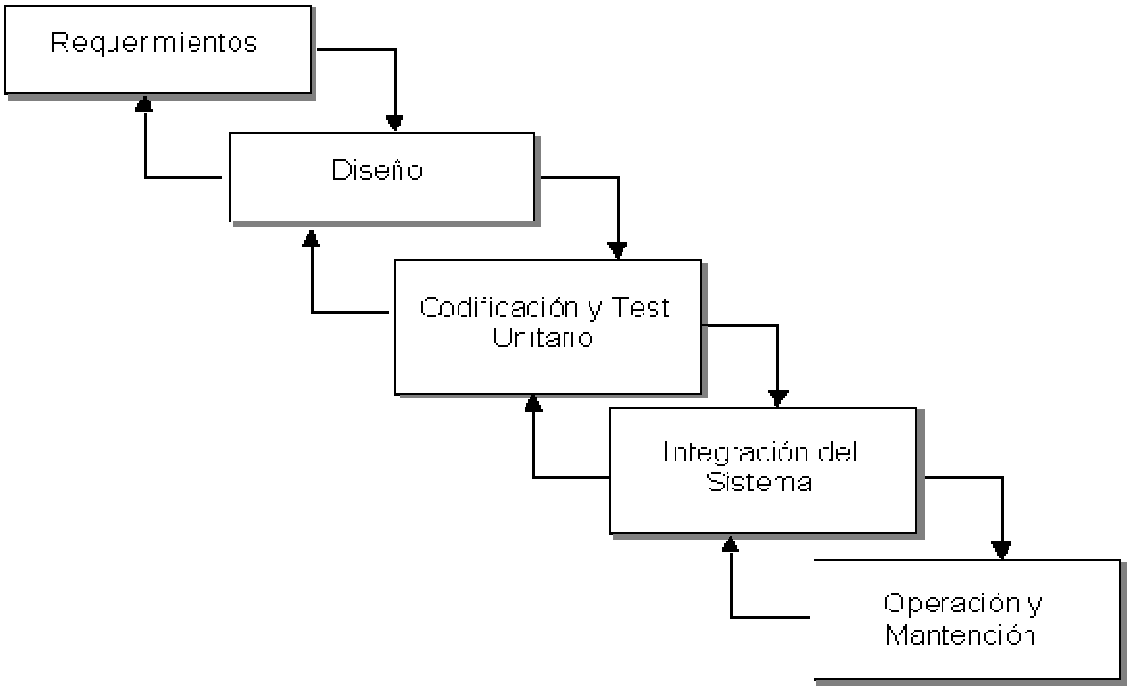

Figura 2. Modelo cascada

3.1.4.2. Modelo de ciclo en V

Propuesto por Alan Davis, tiene las mismas fases que el anterior pero se considera el nivel de abstracción de cada una. Una fase además de utilizarse como entrada para la siguiente, sirve para validar o verificar otras fases posteriores. Este modelo proviene del principio que establece que los procedimientos utilizados para probar si la aplicación cumple las especificaciones ya deben haberse creado en la fase de diseño. En la figura 3 se puede observar un esquema de este modelo.

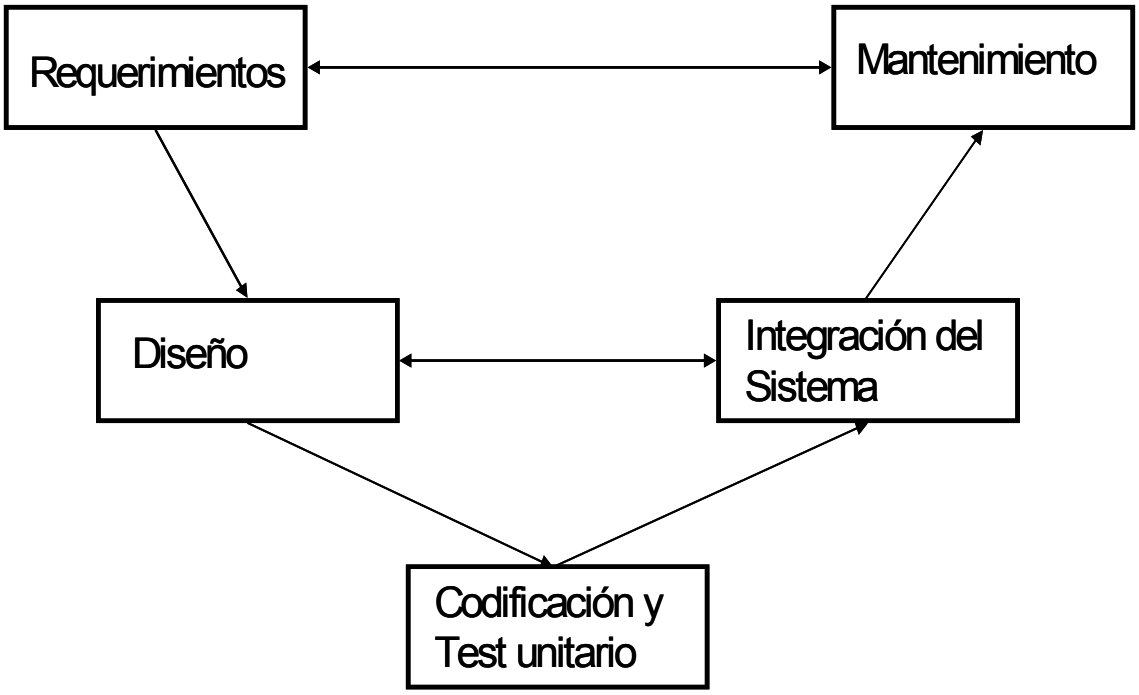

Figura3. Ciclo en V

3.1.4.3. Modelo Sashimi

Según el modelo en cascada puro, una fase solo puede empezar cuando ha terminado la anterior. En este modelo sin embargo, se permite un solapamiento entre fases. Por ejemplo, sin tener terminado del todo el diseño se comienza a implementar. El nombre "sashimi" deriva del estilo de presentación de rodajas de pescado crudo en Japón. Una ventaja de este modelo es que no necesita generar tanta documentación como el ciclo de vida en cascada puro, debido a la continuidad del mismo personal entre fases. Los problemas planteados son:

- Es aún más difícil controlar el progreso del proyecto debido a que los finales de fase ya no son un punto de referencia claro.

- Al hacer cosas en paralelo, si se presentan problemas de comunicación pueden surgir inconsistencias.

En la figura 4 se puede observa un esquema Sashimi.

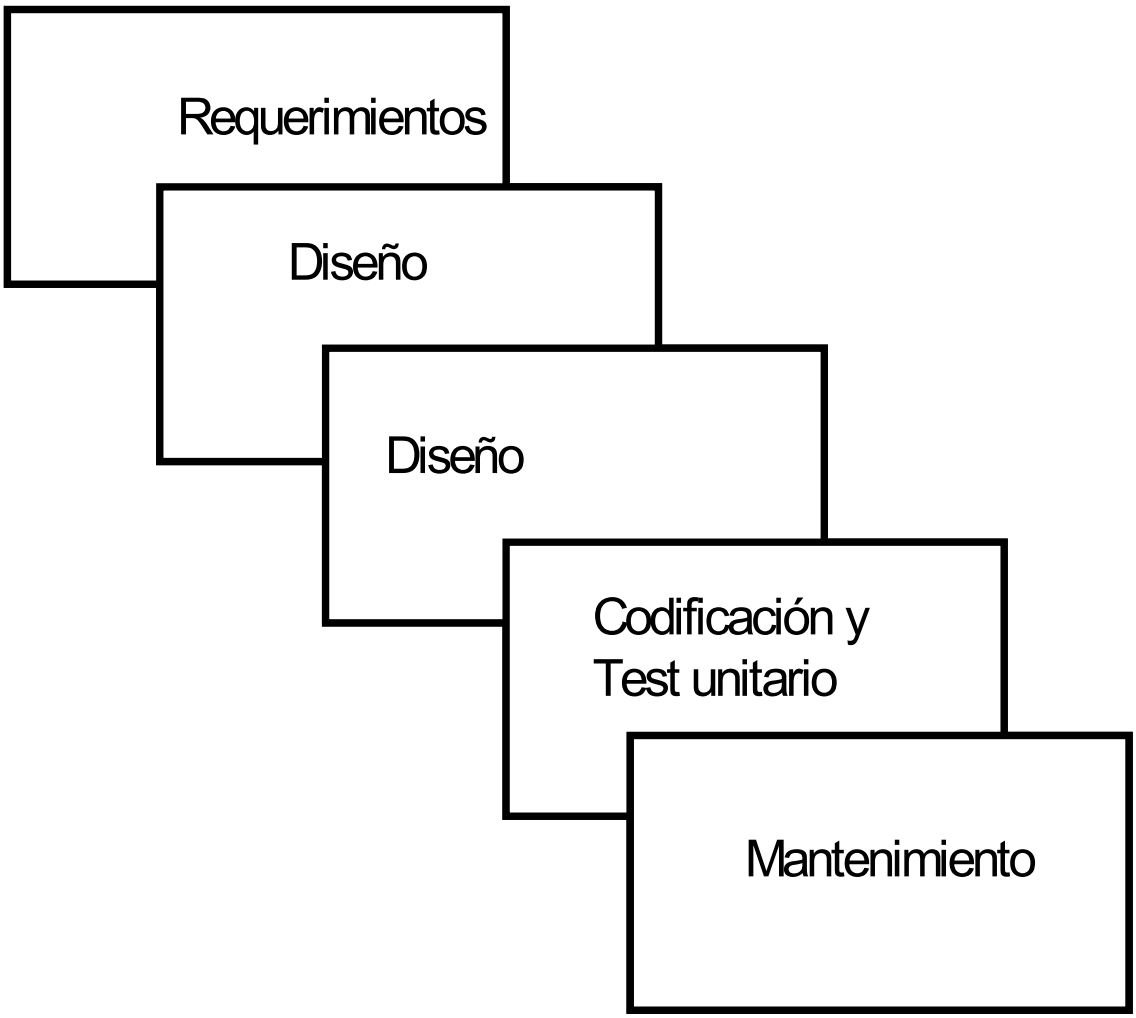

Figura 4. Modelo Sashimi.

3.1.4.4. Modelo de desarrollo incremental

En este caso se va creando el sistema añadiendo pequeñas funcionalidades. Cada uno de los pequeños incrementos es parecido a lo que ocurre dentro de la fase de mantenimiento. La ventaja de este método es que no es necesario tener todos los requisitos en un principio. El inconveniente es que los errores en la detección de requisitos se encuentran tarde. Es ideal cuando se desea ir presentando entregas al cliente antes de la versión final.

La figura 5 y 6 nos muestran como es dicho modelo:

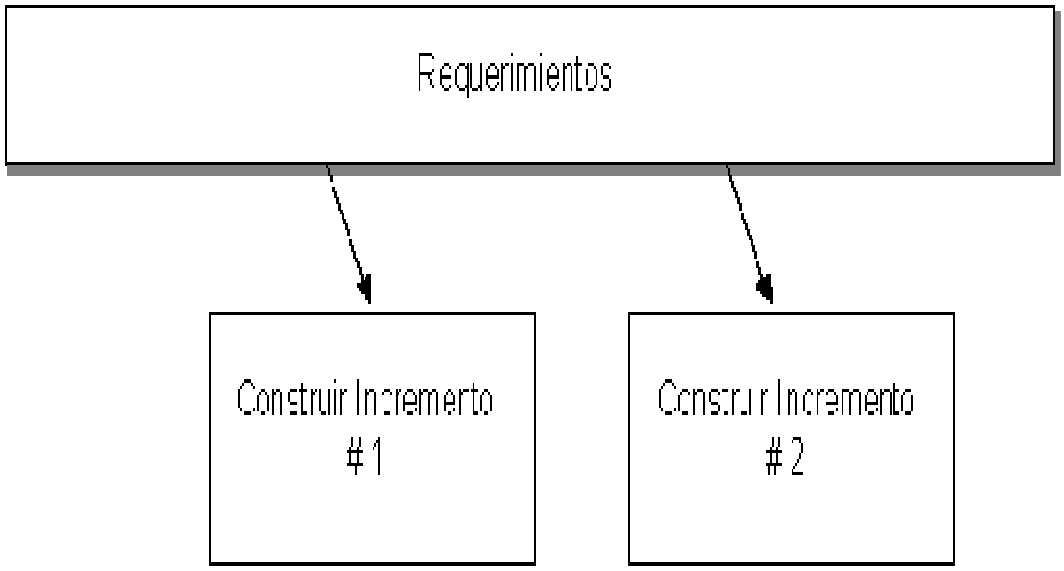

Figura 5. Modelo de desarrollo Incremental

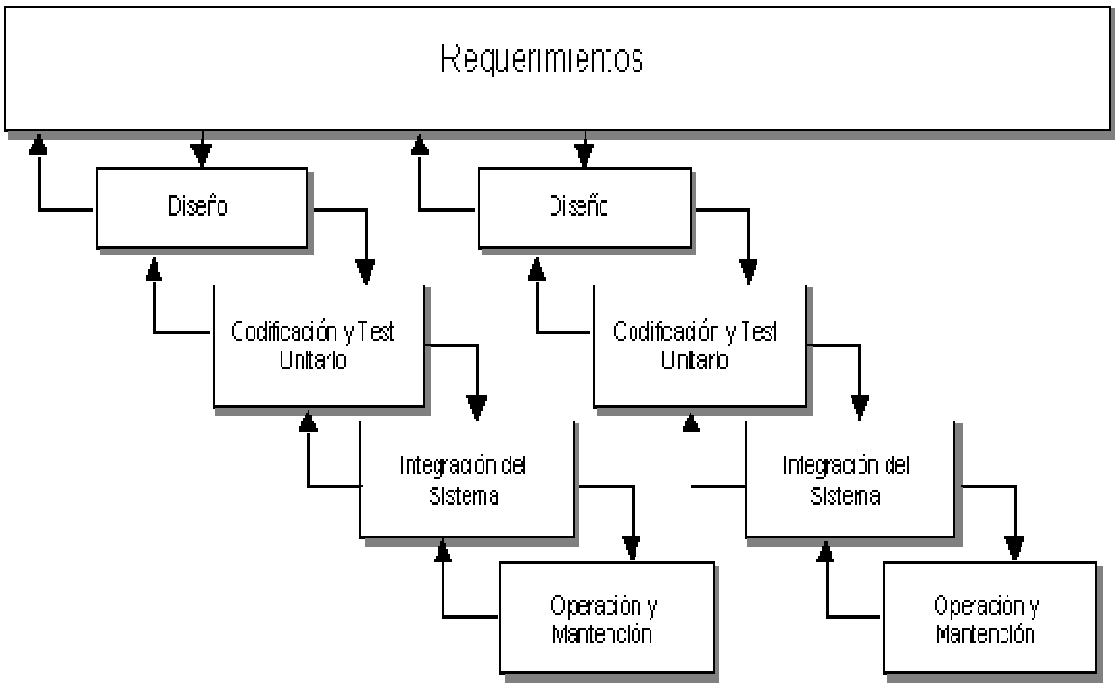

Figura 6. Modelo de desarrollo incremental.

3.1.4.5. Modelo basado en prototipos.

Un prototipo es una versión preliminar de un sistema con fines de demostración o evaluación de ciertos requisitos. El uso de los prototipos implica un método menos formal de desarrollo, donde su fundamento es hacer comprender las especificaciones del producto final.

Un prototipo usado durante un desarrollo software de estas características puede formar parte del producto final o bien puede ser desechado. Las fases que se dan en la construcción de los distintos prototipos de un desarrollo son:

1. Identificación de requisitos que debe de cumplir el prototipo.
2. Diseñar e implementar el prototipo.
3. Utilizar el prototipo con el fin de probar que cumple los requisitos para los que fue diseñado.
4. Revisar y mejorar el prototipo.

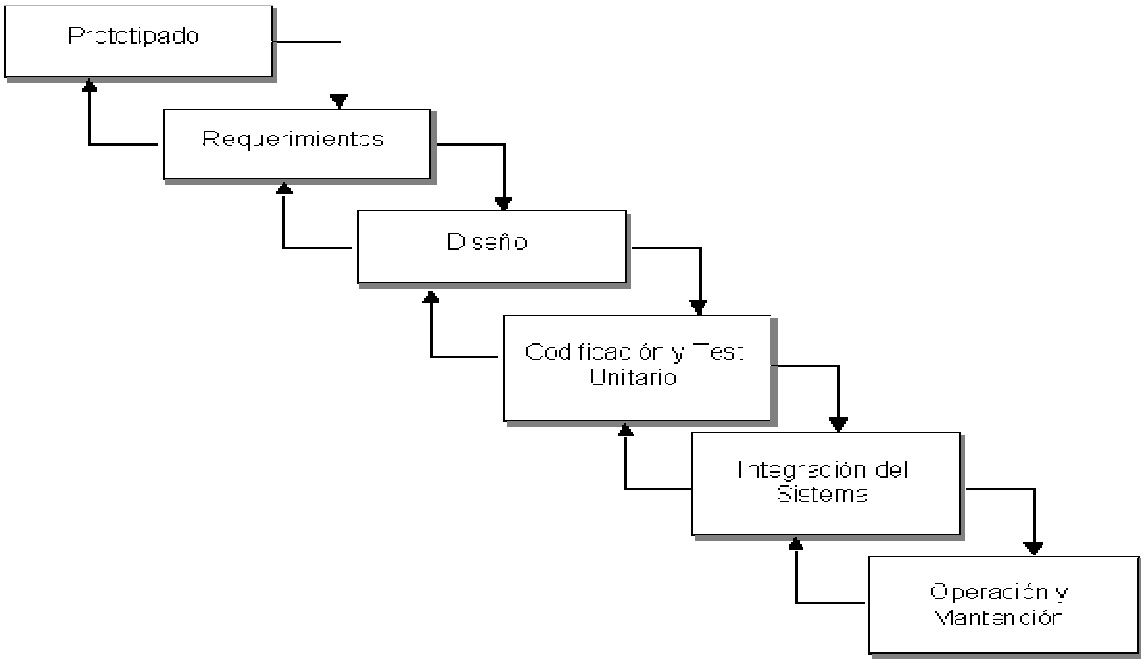

Figura 7. Esquema de prototipado

Con base en lo anterior algunas características de los desarrollos por prototipos son:

- Todos los requerimientos no son conocidos al principio.
- Sólo se desarrollan los que se conocen bien.
- Los usuarios lo prueban y añaden requerimientos.
- La aplicación se hace por fases.
- Se hace una implementación parcial del sistema y se prueba.
- Se utiliza en sistemas complejos.

Las figura 7 y 8 muestran gráficamente como son los esquemas de un modelo de prototipado y un modelo de prototipado incremental.

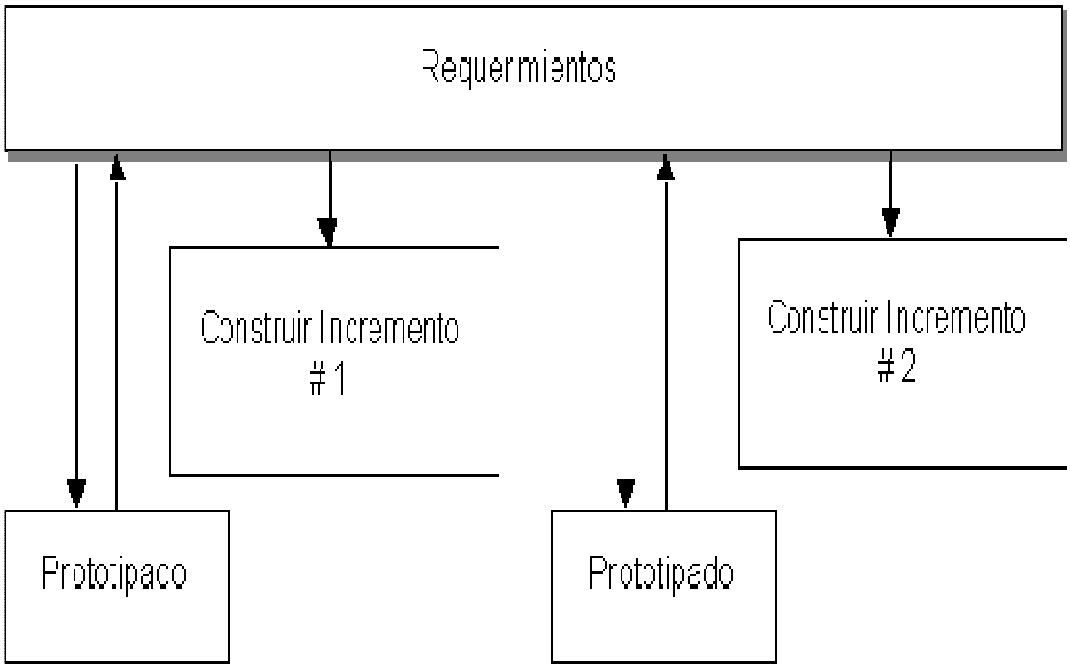

Figura 8. Prototipado Incremental.

3.1.4.6. Modelo en Espiral.

Propuesto inicialmente por Boehm en 1988. Consiste en una serie de ciclos que se repiten. Cada uno tiene las mismas fases y cuando termina da un producto ampliado con respecto al ciclo anterior. En este sentido es parecido al modelo incremental, la diferencia importante es que tiene en cuenta el concepto de riesgo. Un riesgo puede ser muchas cosas: requisitos no comprendidos, un mal diseño, errores en la implementación, etc.

El ciclo de vida en espiral puede considerarse como una generalización del modelo anterior para los casos en que no basta con una sola evaluación de un prototipo para asegurar la desaparición de incertidumbres y/o ignorancias. El propio producto a lo largo de su desarrollo puede así considerarse como una sucesión de prototipos que progresan hasta llegar a alcanzar el estado deseado. En cada ciclo (espirales) las especificaciones del producto se van resolviendo. A menudo la fuente de incertidumbres es el propio cliente, pues aunque sepa en términos generales lo que quiere, no es capaz de definirlo en todos sus aspectos sin observar previamente cómo unos influyen en otros. En estos casos la evaluación de los resultados por el cliente no puede esperar a la entrega final. Además, la evaluación es necesaria repetidas veces.

Las figura 9 muestra gráficamente como es el esquema de un modelo en espiral.

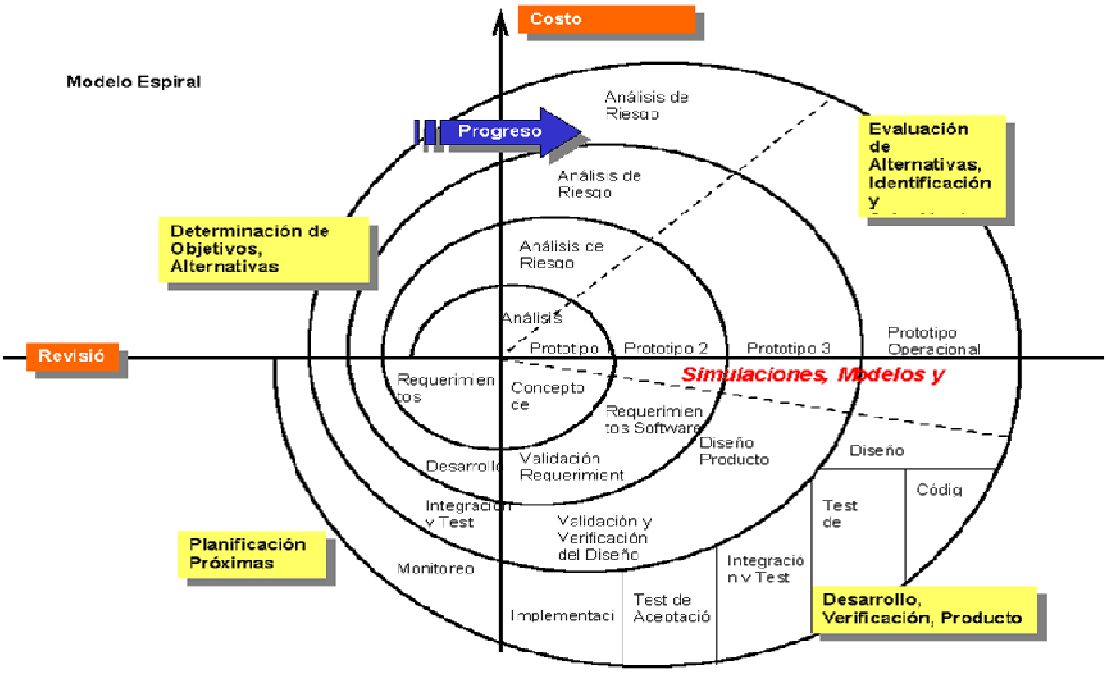

Figura 9. Modelo en espiral.