

UNIVERSIDAD NACIONAL DE LANUS

LICENCIATURA EN SISTEMAS

Ingeniería de Software I

Prof. Tit.: Dr. Ramón García-Martínez

Profs: Adjs: Dr. Darío Rodríguez e Mg. Hernán Amatriain

Ayudantes: Lic. Ezequiel Baldizzoni e Lic. Sebastian Martins

GUIA DE PREGUNTAS

Material "Ciclo de Vida de Software, Proceso Software y Plan de Actividades"

CICLOS DE VIDA

- 1.1. Defina lo que un Ciclo de Vida debe satisfacer.
- 1.2. Defina en base a que se elige un Ciclo de Vida.
- 1.3. Defina y de un esquema gráfico del Ciclo de Vida en Cascada.
- 1.4. Defina Ventajas del Ciclo de Vida en Cascada.
- 1.5. Defina Desventajas del Ciclo de Vida en Cascada.
- 1.6. Enuncie tres propiedades del Ciclo de Vida en Cascada.
- 1.7. Defina y de un esquema del Modelo alternativo del Ciclo de Vida en Cascada que enfatiza la validación de los productos.
- 1.8. Defina condiciones en la que es aconsejable el Ciclo de Vida Prototipado.
- 1.9. Defina Ventajas del Ciclo de Vida Prototipado.
- 1.10. Defina Desventajas del Ciclo de Vida Prototipado.
- 1.11. Describa la como queda el ciclo de vida clásico modificado por la introducción del uso de prototipos.
- 1.12. Defina los tres modelos derivados del uso de Ciclo de Vida Prototipado.
- 1.13. Defina y de un esquema gráfico del Ciclo de Vida en Espiral.
- 1.14. Defina Ventajas del Ciclo de Vida en Espiral.

PROCESO SOFTWARE (Standart IEEE 1074-1989)

- 1.15. Defina los usos de un proceso software
- 1.16. Defina y esquematicice el Proceso Base de Construcción de Software
- 1.17. Defina planificación de proyecto de software
- 1.18. Defina estimación de proyecto de software
- 1.19. Defina medir en el contexto de proyecto de software
- 1.20. Defina Plan de Gestión del Proyecto Software
- 1.21. Defina Plan de Contingencia del Proyecto Software
- 1.22. Defina Plan de Garantía de Calidad de Software y los aspectos que cubre.

- 1.23. Defina Métricas del Software
- 1.24. Defina Verificación y Validación del Software
- 1.25. De un esquema del proceso de diseño de software.
- 1.26. Enuncie criterios de calidad del diseño.
- 1.27. Enuncie objetivos de tipos de cambio que se dan durante el proceso de mantenimiento del software.
- 1.28. Defina las funciones de la gestión de configuración.
- 1.29. Complete el siguiente cuadro en base a los procesos, su definición, las actividades a realizar asociadas, los documentos de salida indicados y las posibles técnicas a utilizar.

PROCESOS	PROCESOS CONTENIDOS	DEFINICIÓN	ACTIVIDADES A REALIZAR	DOCUMENTOS DE SALIDA	TECNICAS A UTILIZAR
Proceso de Selección de un Modelo de Ciclo de Vida del Producto					
Procesos de Gestión del Proyecto					
	Proceso de iniciación del proyecto				
	Proceso de seguimiento y control del proyecto				
	Proceso de gestión de la calidad del software				
Procesos de pre-desarrollo					
	Proceso de exploración de conceptos				
	Proceso de asignación del sistema				
Procesos Orientados al Desarrollo del Software					
	Proceso de requisitos				
	Proceso de diseño				
Procesos de Post-Desarrollo					
	Proceso de instalación				
	Proceso de operación y soporte				
	Proceso de mantenimiento				
	Proceso de Retiro				
Procesos Integrales del Proyecto					
	Proceso de verificación y validación				
	Proceso de gestión de la configuración				
	Proceso de desarrollo de documentación				
	Proceso de formación				

MAPA DE ACTIVIDADES DE UN PROYECTO

- 1.30. Defina Mapa de Actividades
- 1.31. Defina al mapa de actividades como tabla que vincula proceso software (con sus actividades) y el ciclo de vida elegido (descompuesto en sus etapas).
- 1.32. De un ejemplo de mapa de actividades.

Ingeniería del Software I



Guía de Estudio I

Ciclos de Vida, Proceso de Software y Mapa de Actividades

Docentes:

Dr. Ramon García Martínez

Mg. Hernan Amatriain

Lic. Ezequiel Baldizzoni

Dr. Dario Rodriguez

Lic. Sebastian Martins

2016

Índice

1. INTRODUCCIÓN A LOS MODELOS DE CICLOS DE VIDA	3
2. APROXIMACIÓN TRADICIONAL	4
2.1. MODELO DE CICLO DE VIDA EN CASCADA	4
2.2. PROTOTIPADO	7
2.3. MODELO EN ESPIRAL	9
3. VENTAJAS DE DEFINIR UN PROCESO SOFTWARE	11
4. ESTANDAR IEEE SOBRE PROCESO SOFTWARE	12
4.1. INTRODUCCIÓN	12
4.2. DESCRIPCIÓN GLOBAL DEL PROCESO	12
4.3. PROCESO DE SELECCIÓN DE UN MODELO DE CICLO DE VIDA DEL SOFTWARE	14
4.4. PROCESOS DE GESTIÓN DEL PROYECTO	15
4.5. PROCESOS DE PRE-DESARROLLO	19
4.6. PROCESOS DE DESARROLLO	22
4.7. PROCESOS DE POST-DESARROLLO	30
4.8. PROCESOS INTEGRALES DEL PROYECTO	33
5. MAPA DE ACTIVIDADES DE UN PROYECTO	38
6. BIBLIOGRAFÍA	43

1. INTRODUCCIÓN A LOS MODELOS DE CICLOS DE VIDA

No existe un único Modelo de Ciclo de Vida que defina los estados por los que pasa cualquier producto software. Dado que existe una gran variedad de aplicaciones para las que se construyen productos software (software de tiempo real, de gestión, de ingeniería y científico, empotrado, de sistemas, de computadoras personales, etc.) y que dicha variedad supone situaciones totalmente distintas, es natural que existan diferentes Modelos de Ciclo de Vida. Por ejemplo, en aquellos casos en que el problema sea perfectamente conocido, el grupo de desarrollo tenga experiencia en sistemas del mismo tipo, el usuario sea capaz de describir claramente sus requisitos, un ciclo de vida tradicional, en cascada o secuencial sería el adecuado. Por el contrario, si el desarrollo conlleva riesgos (sean técnicos o de otro tipo), un ciclo de vida en espiral será el más apropiado. Sin embargo, si se está ante el caso en que es necesario probarle el producto al usuario para demostrar la utilidad del mismo, se estará ante un ciclo de vida con prototipado, etc.

Un ciclo de vida debe:

- Determinar el orden de las fases del Proceso Software
- Establecer los criterios de transición para pasar de una fase a la siguiente

A continuación, se presentan los diferentes modelos de ciclo de vida más representativos: cascada, espiral y prototipado. No existe un Modelo de Ciclo de Vida que sirva para cualquier proyecto, esto debe quedar claro, cada proyecto debe seleccionar un ciclo de vida que sea el más adecuado para su caso. El ciclo de vida apropiado se elige en base a:

- la cultura de la corporación,
- el deseo de asumir riesgos,
- el área de aplicación,
- la volatilidad de los requisitos,
- y hasta qué punto se entienden bien dichos requisitos.

El ciclo de vida elegido ayuda a relacionar las tareas que forman el proceso software de cada proyecto.

2. APROXIMACIÓN TRADICIONAL

2.1. MODELO DE CICLO DE VIDA EN CASCADA

Este modelo fue presentado por primera vez por Royce en 1970. Se representa, frecuentemente, como un simple modelo con forma de cascada de las etapas del software, como muestra la Figura 2.1. En este modelo la evolución del producto software procede a través de una secuencia ordenada de transiciones de una fase a la siguiente según un orden lineal. Tales modelos semejan una máquina de estados finitos para la descripción de la evolución del producto software. El modelo en cascada ha sido útil para ayudar a estructurar y gestionar grandes proyectos de desarrollo de software dentro de las organizaciones.

Este modelo permite iteraciones durante el desarrollo, ya sea dentro de un mismo estado, ya sea de un estado hacia otro anterior, como muestran las flechas ascendentes de la Figura 2.1. La mayor iteración se produce cuando una vez terminado el desarrollo y cuando se ha visto el software producido, se decide comenzar de nuevo y redefinir los requisitos del usuario.

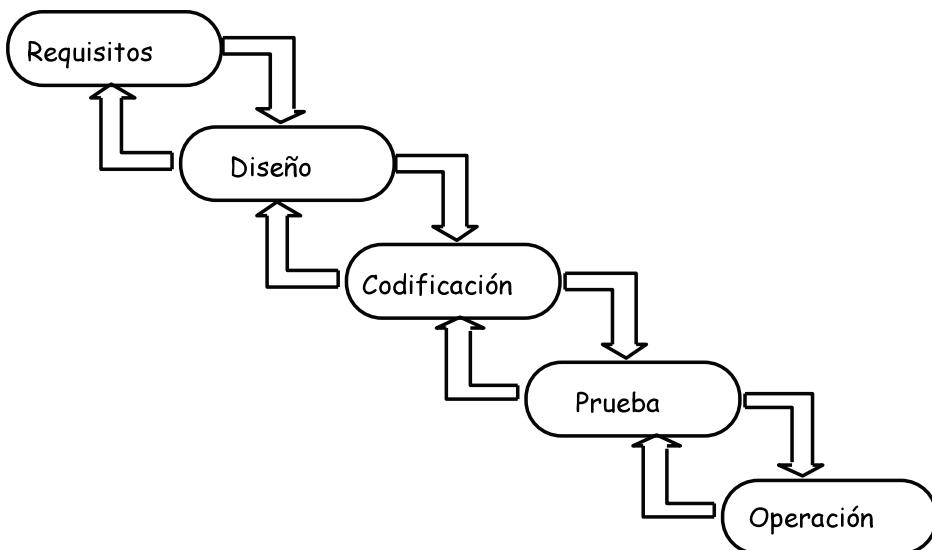


Figura 2.1. Ciclo de vida en cascada

El uso del modelo en cascada:

- Obliga a especificar lo que el sistema debe hacer (o sea, definir los requisitos) antes de construir el sistema (esto es, diseñarlo).
- Obliga a definir cómo van a interactuar los componentes (o sea, diseñar) antes de construir tales componentes (o sea codificar).

- Permite al líder del proyecto seguir y controlar los progresos de un modo más exacto. Esto le permite detectar y resolver las desviaciones sobre la planificación inicial.
- Requiere que el proceso de desarrollo genere una serie de documentos que posteriormente pueden utilizarse para la validación y el mantenimiento del sistema.

A menudo, durante el desarrollo, se pueden tomar decisiones que den lugar a diferentes alternativas, el modelo en cascada no reconoce esta situación. Por ejemplo, dependiendo del análisis de requisitos se puede implementar el sistema desde cero, o adoptar uno ya existente, o comprar un paquete que proporcione las funcionalidades requeridas. Es decir, resulta demasiado estricto para la flexibilidad que necesitan algunos desarrollos.

El modelo en cascada asume que los requisitos de un sistema pueden ser congelados antes de comenzar el diseño.

La figura 2.2. muestra la constante evolución de las necesidades del usuario. Se han representado en un espacio de tiempo/funcionalidad: según pasa el tiempo, aumentan las expectativas de funcionalidades que el usuario espera que tenga el sistema. La evolución está simplificada, pues ni mucho menos es lineal ni continua, pero para hacerse una idea es suficiente.

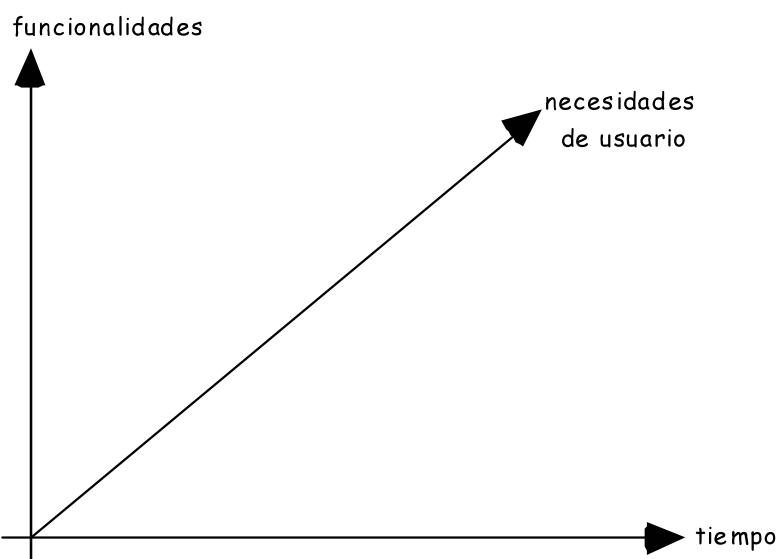


Figura 2.2. Evolución constante de las necesidades del usuario

El ciclo de vida en cascada tiene tres propiedades muy positivas:

- Las etapas están organizadas de un modo lógico. Es decir, si una etapa no puede llevarse a cabo hasta que se hayan tomado ciertas decisiones de más alto nivel, debe esperar hasta que esas decisiones estén tomadas. Así, el diseño espera a los requisitos, el código espera a que el diseño esté terminado, etc.
- Cada etapa incluye cierto proceso de revisión, y se necesita una aceptación del producto antes de que la salida de la etapa pueda usarse. Este ciclo de vida está organizado de modo que se pase el menor número de errores de una etapa a la siguiente.
- El ciclo es iterativo. A pesar de que el flujo básico es de arriba hacia abajo, el ciclo de vida en cascada reconoce, como ya se ha comentado, que los problemas encontrados en etapas inferiores afectan a las decisiones de las etapas superiores.

Existe una visión alternativa del modelo de ciclo de vida en cascada, mostrada en la figura 2.3, que enfatiza en la validación de los productos, y de algún modo en el proceso de composición existente en la construcción de sistemas software.

El proceso de análisis o descomposición subyacente en la línea superior del modelo de la figura 2.3 consiste en: los requisitos del sistema global se dividen en requisitos del hardware y requisitos del software. Estos últimos llevan al diseño preliminar de múltiples funciones, cada una de las cuales se expande en el diseño detallado, que, a su vez, evoluciona aún en un número mayor de programas unitarios. Sin embargo, el ensamblaje del producto final fluye justo en sentido contrario, dentro de un proceso de síntesis o composición. Primero se aceptan los programas unitarios probados. Entonces, éstos se agrupan en módulos que, a su vez, deben ser aceptados una vez probados. Los módulos se agrupan para certificar que el grupo formado por todos ellos incluyen todas las funcionalidades deseadas. Finalmente, el software es integrado con el hardware hasta formar un único sistema informático que satisface los requisitos globales.

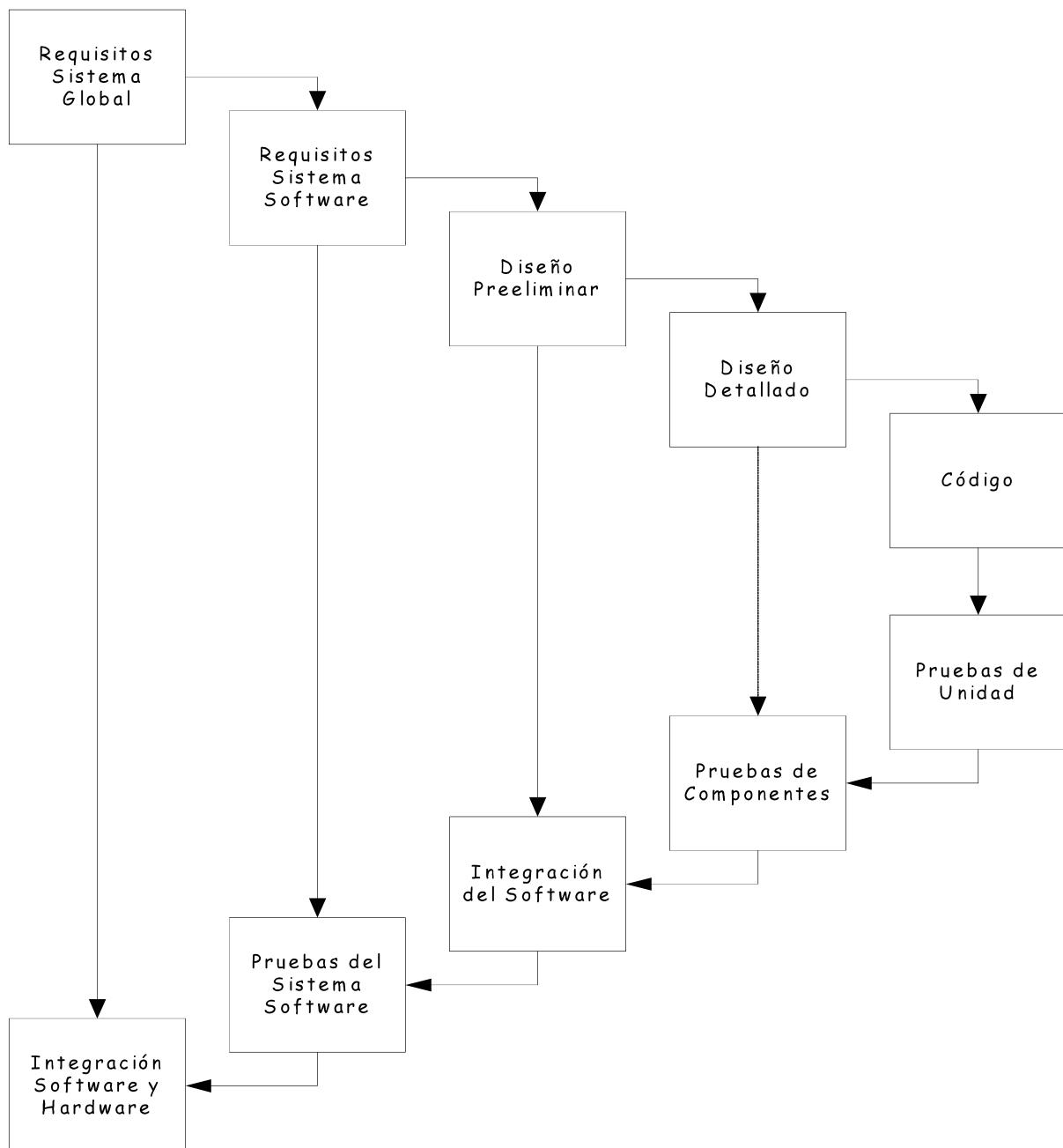


Figura 2.3. Visión alternativa del ciclo de vida en cascada

2.2. PROTOTIPADO

Suele ocurrir que, en los proyectos software, el cliente no tiene una idea muy detallada de lo que necesita, o que el ingeniero de software no está muy seguro de la viabilidad de la solución que tiene en mente. En estas condiciones, la mejor aproximación al problema es la realización de un prototipo.

El modelo de desarrollo basado en prototipos tiene como objetivo contrarestar el problema ya comentado del modelo en cascada: la congelación de requisitos mal comprendidos. La idea básica es que el prototipo ayude a comprender los requisitos del usuario. El prototipo debe incorporar un subconjunto de la función requerida al software, de manera que se puedan apreciar mejor las características y posibles problemas.

La construcción del prototipo sigue el ciclo de vida estándar, sólo que su tiempo de desarrollo será bastante más reducido, y no será muy rigurosa la aplicación de los estándares.

El problema del prototipo es la elección de las funciones que se desean incorporar, y cuáles son las que hay que dejar fuera, pues se corre el riesgo de incorporar características secundarias, y dejar de lado alguna característica importante. Una vez creado el prototipo, se le enseña al cliente, para que "juegue" con él durante un período de tiempo, y a partir de la experiencia aportar nuevas ideas, detectar fallos, etc.

Cuando se acaba la fase de análisis del prototipo, se refinan los requisitos del software, y a continuación se procede al comienzo del desarrollo a escala real. En realidad, el desarrollo principal se puede haber arrancado previamente, y avanzar en paralelo, esperando para un tirón definitivo a la revisión del prototipo.

El ciclo de vida clásico queda modificado de la siguiente manera por la introducción del uso de prototipos:

- 1.- Análisis preliminar y especificación de requisitos
- 2.- Diseño, desarrollo e implementación del prototipo
- 3.- Prueba del prototipo
- 4.- Refinamiento iterativo del prototipo
- 5.- Refinamiento de las especificaciones de requisitos
- 6.- Diseño e implementación del sistema final

Existen tres modelos derivados del uso de prototipos:

- *Maqueta.* Aporta al usuario ejemplo visual de entradas y salidas. La diferencia con el anterior es que en los prototipos desecharables se utilizan datos reales, mientras que las maquetas son formatos encadenados de entrada y salida con datos simples estáticos.
- *Prototipo desecharable.* Se usa para ayudar al cliente a identificar los requisitos de un nuevo sistema. En el prototipo se implantan sólo aquellos

aspectos del sistema que se entienden mal o son desconocidos. El usuario, mediante el uso del prototipo, descubrirá esos aspectos o requisitos no captados. Todos los elementos del prototipo serán posteriormente desechados.

- *Prototipo evolutivo.* Es un modelo de trabajo del sistema propuesto, fácilmente modificable y ampliable, que aporta a los usuarios una representación física de las partes claves del sistema antes de la implantación. Una vez definidos todos los requisitos, el prototipo evolucionará hacia el sistema final. En los prototipos evolutivos, se implantan aquellos requisitos y necesidades que son claramente entendidos, utilizando diseño y análisis en detalle así como datos reales.

2.3. MODELO EN ESPIRAL

El modelo en espiral para el desarrollo de software representa un enfoque dirigido por el riesgo para el análisis y estructuración del proceso software. Fue presentado por primera vez por Böehm en 1986. El enfoque incorpora métodos de proceso dirigidos por las especificaciones y por los prototipos. Esto se lleva a cabo representando ciclos de desarrollo iterativos en forma de espiral, denotando los ciclos internos del ciclo de vida análisis y prototipado precoz, y los externos, el modelo clásico. La dimensión radial indica los costes de desarrollo acumulativos y la angular el progreso hecho en cumplimentar cada desarrollo en espiral. El análisis de riesgos, que busca identificar situaciones que pueden causar el fracaso o sobrepasar el presupuesto o plazo, aparecen durante cada ciclo de la espiral. En cada ciclo, el análisis del riesgo representa groseramente la misma cantidad de desplazamiento angular, mientras que el volumen desplazado barrido denota crecimiento de los niveles de esfuerzo requeridos para el análisis del riesgo como se ve en la figura 2.4.

La primera ventaja del modelo en espiral es que su rango de opciones permiten utilizar los modelos de proceso de construcción de software tradicionales, mientras su orientación al riesgo evita muchas dificultades. De hecho, en situaciones apropiadas, el modelo en espiral proporciona una combinación de los modelos existentes para un proyecto dado. Otras ventajas son:

- Se presta atención a las opciones que permiten la reutilización de software existente.
- Se centra en la eliminación de errores y alternativas poco atractivas.
- No establece una diferenciación entre desarrollo de software y mantenimiento del sistema.
- Proporciona un marco estable para desarrollos integrados hardware-software.

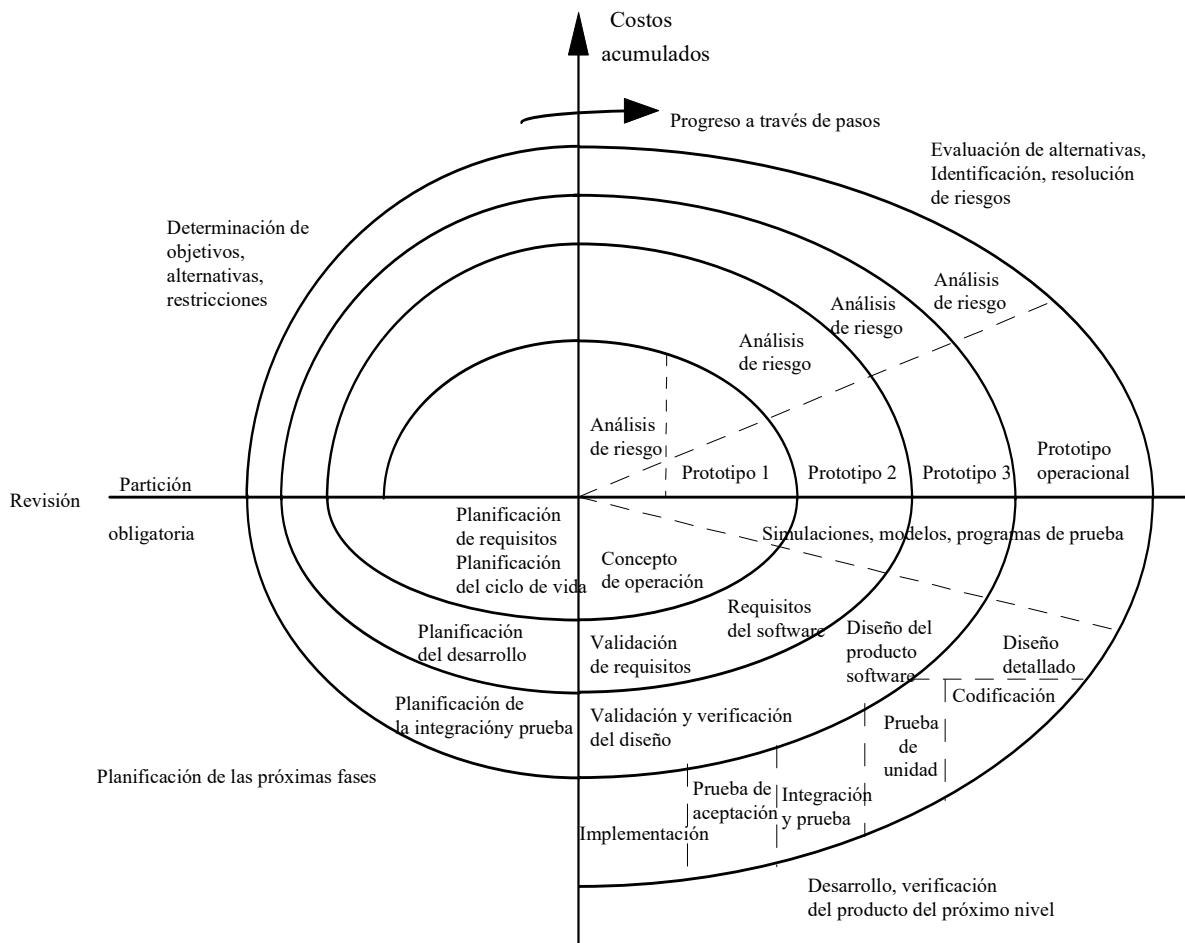


Figura 2.4. *Modelo en Espiral*

En realidad el modelo en espiral no significa una visión radicalmente distinta de los modelos tradicionales, o prototipado. Cualquiera de los modelos pueden verse con una representación espiral. Este modelo de Boehm es más bien una formalización o representación de los modelos de ciclo de vida más acertada que la representación en forma de cascada, pues permite observar mejor todos los elementos del proceso (incluido riesgos, objetivos, etc.).

3. VENTAJAS DE DEFINIR UN PROCESO SOFTWARE

Un proyecto sin estructura es un proyecto inmanejable. No puede ser planificado, ni estimado, ni su progreso ser controlado, y mucho menos alcanzar un compromiso de costes o tiempos. La idea originaria de buscar ciclos de vida que describan los estados por los que pasa el producto, o procesos software que describan las actividades a realizar para transformar el producto, surge de la necesidad de tener un esquema que sirva como base para planificar, organizar, asignar personal, coordinar, presupuestar, y dirigir las actividades de la construcción de software. De hecho, muchos proyectos han terminado mal porque las fases de desarrollo se realizaron en un orden erróneo.

Por tanto, al comienzo de un proyecto software, se debe elegir el Ciclo de Vida que seguirá el producto a construir, en base a las consideraciones del apartado anterior. El Modelo de Ciclo de Vida elegido llevará a *encadenar* las tareas y actividades del Proceso Software de una determinada manera: algunas tareas no será necesario realizarlas, otras deberán realizarse más de una vez, etc. Una vez conseguido un Proceso Software concreto para el proyecto en cuestión, se está preparado para planificar los plazos del proyecto, asignar personas a las distintas tareas, presupuestar los costos del proyecto, etc.

Concretamente, los procesos software se usan como:

- Guía prescriptiva sobre la documentación a producir para enviar al cliente.
- Base para determinar qué herramientas, técnicas y metodologías de Ingeniería de Software serán más apropiadas para soportar las diferentes actividades.
- Marco para analizar o estimar patrones de asignación y consumo de recursos a lo largo de todo el ciclo de vida del producto software.
- Base para llevar a cabo estudios empíricos para determinar qué afecta a la productividad, el coste y la calidad del software.
- Descripción comparativa sobre cómo los sistemas software llegan a ser lo que son.

4. ESTANDAR IEEE SOBRE PROCESO SOFTWARE

4.1. INTRODUCCIÓN

A continuación, se describen en detalle las fases o subprocessos que conforman el Proceso Base de Construcción del Software y que se corresponden con el estándar IEEE 1074 [IEEE 1074, 1989]. Cada subprocesso se detalla a nivel de propósito, actividades involucradas y documentación principal propuesta por el estándar. El estándar IEEE 1074-1989 determina el conjunto de actividades esenciales, no ordenadas en el tiempo, que deben ser incorporadas dentro de un Modelo de Ciclo de Vida del producto software. Este modelo es seleccionado y establecido por el usuario para el proyecto a desarrollar, ya que la norma no define un ciclo de vida en particular.

El estándar, además, incluye la siguiente información que no se trata aquí:

- Descripción de cada actividad.
- Informe de entrada y salida para cada actividad.
- Fuente y destino de la información a nivel de proceso y de actividad, que refleja la relación entre los procesos.
- Productos obtenidos por el desarrollo de cada actividad.

El estándar ha sido escrito por organizaciones responsables de la gestión y desarrollo del software. Está dirigido a los gestores de proyectos, a los desarrolladores de software, a los responsables de la garantía de la calidad, a quienes ejecutan tareas de apoyo, a los usuarios y al personal de mantenimiento.

4.2. DESCRIPCIÓN GLOBAL DEL PROCESO

El Proceso Base de Construcción de Software consiste en analizar las necesidades de la organización en un dominio, desarrollar una solución que las satisfaga y posteriormente reinsertar la solución en el dominio, bajo un marco de gestión, seguimiento, control y gestión de la calidad. Esta relación del Proceso Base con su entorno organizativo se representa en la figura 4.1.

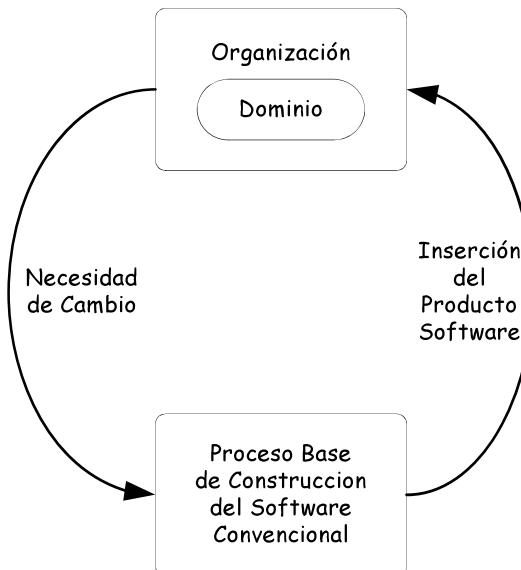


Figura 4.1. Relación del Proceso Base de Construcción de Software con su entorno organizativo

El proceso software está compuesto, a su vez, de cuatro procesos principales tal y como se muestra en la figura 4.2., cada uno de los cuales agrupa una serie de actividades que se encargan de la realización de sus requisitos asociados. Estos procesos son los siguientes:

- **Proceso de Selección de un Modelo de Ciclo de Vida del Producto:** identifica y selecciona un ciclo de vida para el software que se va a construir.
- **Procesos de Gestión del Proyecto:** crean la estructura del proyecto y aseguran el nivel apropiado de la gestión del mismo durante todo el ciclo de vida del software.
- **Procesos Orientados al Desarrollo del Software:** producen, instalan, operan y mantienen el software y lo retiran de su uso. Se clasifican en procesos de pre-desarrollo, desarrollo y post-desarrollo.
- **Procesos Integrales del Proyecto:** son necesarios para completar con éxito las actividades del proyecto software. Aseguran la terminación y calidad de las funciones del mismo. Son simultáneos a los procesos orientados al desarrollo del software e incluyen actividades de no desarrollo.

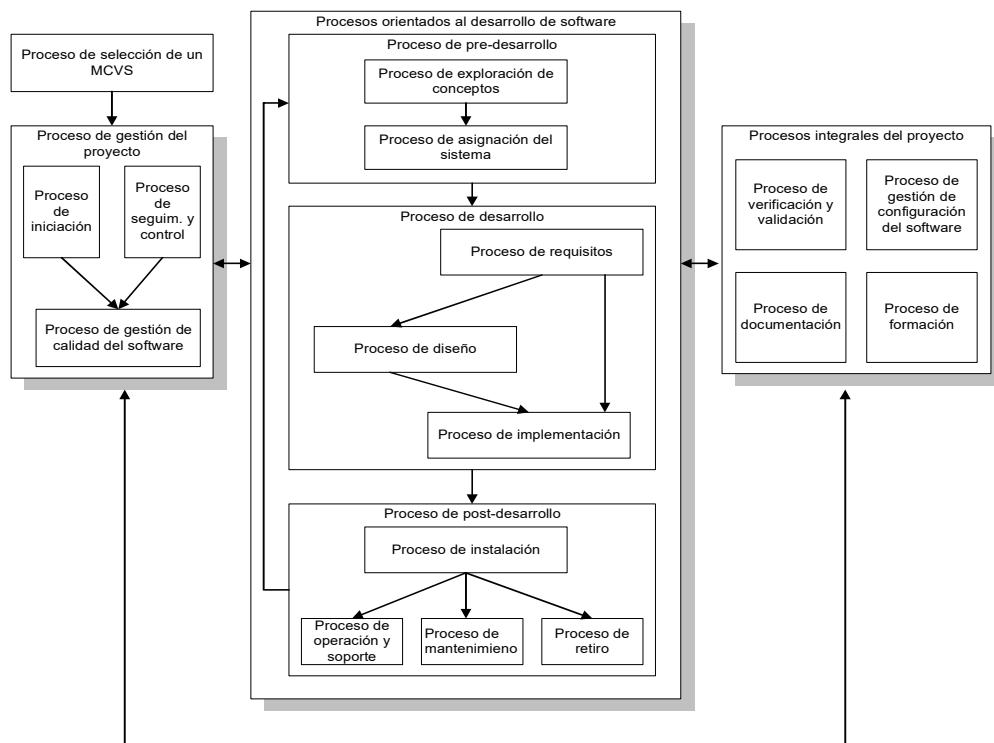


Figura 4.2. *Modelo de proceso software (IEEE 1989)*

4.3. PROCESO DE SELECCIÓN DE UN MODELO DE CICLO DE VIDA DEL SOFTWARE

Durante este proceso se selecciona un ciclo de vida que establece el orden de ejecución de las distintas actividades marcadas por el proceso e involucradas en el proyecto. Como ya se ha visto, en base al tipo de producto software a desarrollar y a los requisitos del proyecto se identifican y analizan posibles modelos de ciclo de vida para dicho proyecto y se selecciona un único modelo que lo soporte adecuadamente. El estándar no dictamina ni define un ciclo de vida del software específico, ni una metodología de desarrollo, sólo requiere elegir y seguir un modelo de ciclo de vida.

La información de salida de este proceso es el **Modelo de Ciclo de Vida del Software** seleccionado

Actividades a realizar:

- Identificar los posibles modelos de ciclo de vida del software.
- Seleccionar el modelo más adecuado para el proyecto.

Documentos de salida:

- Modelo de ciclo de vida seleccionado.

4.4. PROCESOS DE GESTIÓN DEL PROYECTO

La gestión del proyecto presupone establecer condiciones para el desarrollo del mismo. Involucra actividades de planificación, estimación de recursos, seguimiento y control, y evaluación del proyecto.

La **planificación** de proyectos se define como la predicción de la duración de las actividades y tareas a nivel individual, los recursos requeridos, la concurrencia y solapamiento de tareas para ser desarrollados en paralelo y el camino crítico a través de la red de actividades.

La **estimación** se define como la predicción de personal, esfuerzo y costos que se requerirá para terminar todas las actividades y productos conocidos asociados con el proyecto.

La determinación del tamaño del producto a desarrollar es una de las primeras tareas en la gestión del proyecto, ya que sin unos conocimientos razonables, es imposible planificar o estimar el esfuerzo involucrado. El tamaño se define como la cantidad de código fuente, especificaciones, casos de prueba, documentación del usuario y otros productos tangibles que son salida del proyecto. La determinación del tamaño se basa principalmente en la experiencia de proyectos anteriores.

El **seguimiento** de proyectos es la recolección de datos y su acumulación sobre recursos consumidos, costos generados, e hitos asociados con un proyecto. **Medir** en un proyecto se define como el registro de todos los productos generados en el mismo, de todos los recursos requeridos, planificación y solapamiento de todas las actividades y tareas y de todos los factores que impactan en el proyecto (conocimientos, métodos, herramientas, lenguajes, limitaciones, problemas y el entorno físico).

La medición en los proyectos de desarrollo de software es una actividad fundamental para la mejora de la productividad, el costo y la calidad del producto final. La medición, entrada a estudios empíricos, es el único modo para detectar fallos en el proceso de construcción de software.

PROCESO DE INICIACIÓN DEL PROYECTO

Abarca aquellas actividades de creación de la estructura del proyecto. Durante este proceso se define el ciclo de vida del software (asignación de responsables de cada actividad) para este proyecto y se establecen los planes para su gestión. Se estiman y asignan los recursos; esto consiste en determinar los costos y recursos necesarios a fin de ejecutar las distintas tareas que demanda el proyecto. Se identifican y seleccionan estándares, metodologías y herramientas para la gestión y ejecución del mismo y, por último, se prepara y establece un plan para su implementación adecuada y oportuna, incluyendo hitos y revisiones.

El **Plan de Gestión del Proyecto Software** que conducirá el desarrollo se produce como culminación de este proceso.

Actividades a realizar:

- Establecer el mapa de actividades para el modelo de ciclo de vida del software seleccionado.
- Asignar los recursos del proyecto.
- Definir el entorno del proyecto.
- Planificar la gestión del proyecto.

Documentos de salida:

- Plan de Gestión del Proyecto.
- Plan de Retiro.

Técnicas a utilizar:

- Análisis de Camino Crítico (CPM).
- Análisis PERT.
- Diagrama de GANTT.
- Técnicas Estadísticas.
- Técnicas de Simulación (Método de MONTECARLO).
- Puntos de Función.
- Modelos Empíricos de Estimación (COCOMO, PUTMAN).
- Técnicas de Descomposición para Estimación.

PROCESO DE SEGUIMIENTO Y CONTROL DEL PROYECTO

Es un proceso iterativo de seguimiento, registro y gestión de costos, problemas, y rendimiento de un proyecto durante su ciclo de vida. En este proceso se realiza un análisis de riesgos (técnico, económico, operativo y de soporte, y del programa o calendario) que permite identificar los problemas potenciales, determinar su

probabilidad de ocurrencia y su impacto y establecer los pasos para su gestión. Los riesgos identificados y su gestión se documentan en el **Plan de Contingencia**.

El progreso de un proyecto se revisa y mide con respecto a los hitos establecidos en el Plan de Gestión del Proyecto Software.

Actividades a realizar:

- Analizar riesgos.
- Realizar la planificación de contingencias.
- Gestionar el proyecto.
- Archivar registros.
- Implementar el Sistema de Informes de Problemas.

Documentos de salida:

- Análisis de riesgos.
- Plan de contingencias.
- Registro histórico de proyectos.

Técnicas a utilizar:

- Análisis de Riesgo Técnico.
 - Modelización y Simulación Estática y Dinámica.
 - Prototipado.
 - Revisiones.
 - Auditorías.
- Análisis de Riesgo Económico.
- Análisis de Finanzas.
- Retorno de la Inversión.
- Análisis de Riesgo Operativo y de Soporte.
- Análisis de Riesgo del Programa.
 - Análisis de Camino Crítico (CPM).
 - Técnicas de Nivelación de Recursos.

PROCESO DE GESTIÓN DE LA CALIDAD DEL SOFTWARE

Su objetivo es la planificación y administración de las acciones necesarias para proveer una confianza adecuada en la calidad de los productos software; es decir, que satisfagan los requisitos técnicos establecidos.

El proceso de Gestión de la Calidad del Software se documenta en un **Plan de Garantía de la Calidad del Software**. Sus actividades abarcan el ciclo de vida completo del software. Para abordar este proceso de protección del software, con una visión global, se consideran los siguientes tres aspectos principales:

- Métricas del Software para el control del proyecto.
- Verificación y Validación, incluyendo pruebas y procesos de revisión.
- Gestión de la Configuración del producto software.

Las **Métricas del Software** se definen como la aplicación continua de técnicas basadas en las medidas de los procesos de desarrollo del software y de sus productos para producir una información de gestión significativa y a tiempo, a la vez que se mejoran los procesos y sus productos.

La **Verificación y Validación** del Software involucra actividades imprescindibles para el control de la calidad del software. Se entiende por **Verificación** al conjunto de actividades para la comprobación de que un producto software está técnicamente bien construido; es decir, que el producto funciona (¿está el producto correctamente construido?). En general, comprobar si los productos construidos en una fase del ciclo de vida satisfacen los requisitos establecidos en la fase anterior, decidiendo si el producto, hasta el momento, es consistente y completo. De modo complementario la **Validación** trata la comprobación de que el producto software construido es el que se deseaba construir; es decir, que el producto funciona como el usuario quiere y hace las funciones que se establecieron (¿el producto construido es el correcto?). En general, comprobar si el software construido satisface los requisitos del usuario. Evidentemente, sólo tiene objeto validar el producto que está verificado (no interesa comprobar si un producto que no funciona es lo que se pedía).

Tanto el proceso de Verificación y Validación, como el proceso de Gestión de la Configuración del software se tratan posteriormente, dentro de los procesos integrales.

Actividades a realizar:

- Planificar la garantía de la calidad del software.
- Desarrollar métricas de calidad.
- Gestionar la calidad del software.
- Identificar necesidades de mejora de la calidad.

Documentos de salida:

- Plan de garantía de calidad del software.
- Recomendaciones de mejora de calidad software.

Técnicas a aplicar:

- Técnicas de Planificación y Estimación.
- Métricas de Calidad del Software.

4.5. PROCESOS DE PRE-DESARROLLO

Los procesos orientados al desarrollo de software, sean pre, en o pos, se estudian en el Módulo del curso llamada "Técnicas de Ingeniería de Software"

Son los procesos que se deben realizar antes de que comience el desarrollo propiamente dicho del software. El esfuerzo de desarrollo se inicia con la identificación de una necesidad de automatización. Esta necesidad, para ser satisfecha, puede requerir una nueva aplicación, o un cambio de todo o parte de una aplicación existente. El pre-desarrollo abarca desde el reconocimiento del problema hasta la determinación de los requisitos funcionales a nivel de sistema, pasando por el estudio de la viabilidad de su solución automatizada.

PROCESO DE EXPLORACIÓN DE CONCEPTOS

Este proceso incluye la identificación de una necesidad, la formulación de soluciones potenciales, su evaluación (estudio de viabilidad) y refinamiento a nivel de sistema. Una vez establecidos sus límites, se genera el **Informe de la Necesidad** del sistema a desarrollar. Este informe inicia el Proceso de Asignación del Sistema y, o, el Proceso de Requisitos, y alimenta los Procesos de Gestión del Proyecto.

El **Informe de la Necesidad** es un documento que constituye la base de todo el trabajo de ingeniería posterior.

Actividades a realizar:

- Identificar ideas o necesidades.
- Formular soluciones potenciales.
- Conducir estudios de viabilidad.
- Planificar la transición del sistema (si se aplica).
- Refinar y Finalizar la idea o necesidad.

Documentos de salida:

- Modelo de la situación actual.
- Modelo del dominio del problema.
- Informe preliminar de necesidades.
- Soluciones alternativas posibles.
- Soluciones recomendadas.
- Plan de transición.
 - Informe del impacto de la transición.

Técnicas a usar:

- Técnicas de Adquisición de Conocimientos.
- Análisis Económico (Coste/Beneficio).
- Análisis Técnico.
- Análisis Alternativos.
- Técnicas de Modelización.
- Diagramas de Flujos de Datos (DFD).
- Prototipado.

PROCESO DE ASIGNACIÓN DEL SISTEMA

Este proceso se realiza cuando el sistema requiere tanto del desarrollo de hardware como de software, o cuando no se puede asegurar que sólo se necesita desarrollo de software. El Informe de la Necesidad se analiza para identificar las entradas, el procesamiento que se aplica a la entrada, las salidas requeridas y las funciones del sistema total, que permiten desarrollar la arquitectura del sistema e identificar las funciones del hardware, del software y de los interfaces. Este proceso culmina con la Especificación de Requisitos del Software, la Especificación de Requisitos del Hardware y la Especificación del Interfaz del Sistema.

Debido a que el software es parte de un sistema mayor, se comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software, para su análisis y refinamiento en el Proceso de Requisitos. Este planteamiento del sistema es esencial cuando el software debe interrelacionarse con otros elementos, tales como hardware, personas y bases de datos. Antes de la asignación del sistema, el análisis abarca los requisitos globales a nivel de sistema con una pequeña cantidad de análisis y de diseño a un nivel superior.

El análisis de sistemas requiere una comunicación intensa entre el cliente y el analista. El cliente debe comprender los objetivos del sistema y ser capaz de exponerlos claramente. El analista debe saber qué preguntas hacer, qué consejos dar y qué investigación realizar. Si la comunicación se rompe, el éxito del proyecto entero estará en peligro.

En el análisis del sistema (Proceso de Exploración de Conceptos) se definen los objetivos del mismo, la información que se va a obtener, la información que se va a suministrar, las funciones, el comportamiento y el rendimiento requerido. El analista se asegura de distinguir entre lo que "necesita" el cliente (elementos críticos para la realización) y lo que el cliente "quiere" (elementos deseables pero no esenciales).

Una vez que la función, el rendimiento y los interfaces (determinados por el

comportamiento) están delimitados, se procede a realizar la tarea denominada asignación. Durante la asignación, las funciones son asignadas a uno o más elementos genéricos del sistema; es decir, software, hardware, personal, bases de datos, documentación, procedimientos. A menudo, se proponen y evalúan varias asignaciones. Esencialmente, lo que se hace es asignar a cada elemento del sistema un ámbito de funcionamiento y de rendimiento.

Asignadas las funciones del sistema informático, se puede crear un modelo que represente las interrelaciones, entre los distintos elementos del sistema y establezca una base para los posteriores pasos de análisis de requisitos y de diseño. Se representa el sistema definido mediante modelos de la arquitectura del sistema (salida, entrada, proceso y control, interfaz de usuario, y mantenimiento y autocomprobación). En primer lugar, se realiza un **Diagrama de Contexto** de la arquitectura (DC), que establece los límites de información entre los que se está implementando el sistema y el entorno en el que va a funcionar. En esencia, el DC ubica el sistema en el contexto de su entorno externo.

Se refina el diagrama de contexto de la arquitectura considerando con más detalle la función del sistema. Se identifican los subsistemas principales que permiten el funcionamiento del sistema considerado en el contexto definido por el DC. Se definen los subsistemas principales en un **Diagrama de Flujo** de la arquitectura (DF). El diagrama de flujo de la arquitectura muestra los subsistemas principales y las líneas importantes de flujo de información (control y datos). En esta etapa, cada uno de los subsistemas pueden contener uno o más elementos del sistema (por ejemplo: hardware, software, personal) según hayan sido asignados.

El diagrama inicial de flujo de la arquitectura se constituye en el nodo raíz de la jerarquía de DF. Se puede ampliar cada subsistema del DF inicial en otro diagrama de arquitectura dedicado exclusivamente a él. Este proceso de descomposición de arriba a abajo (top-down) permite disponer de una jerarquía de DF, donde cada uno de los DF del sistema se puede utilizar como punto de partida para los posteriores pasos de ingeniería del subsistema que describe.

Actividades a realizar:

- Analizar las funciones del sistema.
- Desarrollar la arquitectura del sistema.
- Descomponer los requisitos del sistema.

Documentos de salida:

- Especificación de requisitos funcionales del software.
- Especificación de requisitos funcionales del hardware.

- Especificación de la interfaz del sistema.
- Descripción funcional del sistema.
- Arquitectura del sistema.

Técnicas a utilizar:

- Técnicas de Adquisición de Conocimientos.
- Técnicas de Modelización.
- Diagramas de Flujo de Datos (DFD).

4.6. PROCESOS DE DESARROLLO

Son los procesos que se deben realizar para la construcción del producto software. Estos definirán qué información obtener y cómo estructurar los datos, qué algoritmos usar para procesar los datos y cómo implementarlos y qué interfaces desarrollar para operar con el software y cómo hacerlo.

A partir del Informe de la Necesidad, con el soporte de las actividades de los Procesos Integrales y bajo el Plan de Gestión del Proyecto, los Procesos de Desarrollo producen el software (código y documentación).

PROCESO DE REQUISITOS

Incluye las actividades iterativas dirigidas al desarrollo de la Especificación de Requisitos del Software. Para la determinación completa y consistente de los requisitos del software el análisis se enfatiza sobre la salida resultante, la descomposición de los datos, el procesamiento de los datos, las bases de datos (si existen) y los interfaces del usuario, del software y del hardware.

La **Especificación de Requisitos del Software** es el establecimiento conciso y preciso de un conjunto de requisitos que deben ser satisfechos por un producto software, indicando, siempre que sea apropiado, el procedimiento mediante el cual se puede determinar si se satisfacen los requisitos dados. Describe los requisitos funcionales, de rendimiento, y de interfaz del software y define los entornos de operación y de soporte. Este documento es la salida con que culmina este proceso.

Un requisito es una condición o característica que debe tener o cumplir un sistema o componente de un sistema para satisfacer un contrato, norma, especificación, u otro documento formalmente impuesto. El conjunto de todos los requisitos forma la base para el desarrollo consiguiente del sistema o componente del sistema.

Existen tres tipos de requisitos: funcionales, de rendimiento y de interfaz. Un **requisito funcional** especifica la función que un sistema o componente de un sistema debe ser capaz de realizar (por ejemplo: emitir facturas). Un **requisito de rendimiento** especifica una característica numérica tanto estática (por ejemplo: número de terminales del sistema, número de usuarios simultáneos que soporta el sistema, número de archivos y registros del mismo, etc.) como dinámica (por ejemplo: el 95% de las transacciones se deben procesar en menos de un segundo) que debe tener un sistema o componente de un sistema. Los **requisitos de interfaz** determinan las características que el software debe soportar para cada interfaz humano del producto software (interfaz de usuario), las características lógicas de cada interfaz entre el producto software y las componentes hardware del sistema (interfaz de hardware), el uso de otros productos software (por ejemplo: un sistema de gestión de base de datos, un sistema operativo, o un paquete matemático) e interfaces con otros sistemas de aplicación (interfaz de software). Por ejemplo: que la interfaz del usuario cuente con ventanas superpuestas.

Existen además algunos atributos del software (seguridad, consistencia, facilidad de traza, etc.) que pueden dar lugar a requisitos específicos del mismo, por ejemplo: que el acceso a ciertos datos sea restringido.

Actividades a realizar:

- Definir y desarrollar los requisitos del software.
- Definir los requisitos de interfaz.
- Priorizar e integrar los requisitos del software.

Documentos de salida:

- Especificación de requisitos del software.
- Requisitos del interfaz con el usuario.
- Requisitos del interfaz con otro software.
- Requisitos del interfaz con el hardware.
- Requisitos del interfaz con el sistema físico.

Técnicas a utilizar:

- Técnicas Orientadas a los Procesos:
 - Análisis Estructurado.
 - Diagramas de Flujos de Datos (DFD).
 - Diccionario de Datos (DD).
 - Especificación de Procesos Primitivos (EPP).
 - SADT (Structured Analyses and Design Techniques).
 - Diagramas de Transición de Estados.

- Diagramas de Descomposición.
 - WRS (Working Breakdown Structure).
 - RBS (Resources Breakdown Structure).
 - OBS (Object Breakdown Structure).
- ACTIGRAMAS (Diagrama de Actividades).
- Técnicas Orientadas a los Datos:
 - Diagramas de Entidad-Relación.
 - DATAGRAMAS (Diagramas de Datos).
- Técnicas Orientadas a los Objetos:
 - Diagrama de Clases/Objetos.
 - Jerarquía de Clases/Objetos.
- Técnicas Formales de Especificación
 - Técnicas Relacionales:
 - Ecuaciones Implícitas.
 - Relaciones Recurrentes.
 - Axiomas Algebraicos.
 - Expresiones Regulares.
 - Técnicas Orientadas al Estado.
 - Tablas de Decisión.
 - Tablas de Eventos.
 - Tablas de Transición.
 - Mecanismos de Estados Finitos.
 - Redes de Petri.
- Técnicas de Prototipación.

PROCESO DE DISEÑO

Es el proceso central que unifica los procesos de desarrollo y de mantenimiento del software. Su objetivo es desarrollar una representación coherente y organizada del sistema software que satisfaga la Especificación de Requisitos del Software. La calidad de dicha representación se puede evaluar. El Proceso de Diseño traduce el "qué hacer" de las especificaciones de los requisitos en el "cómo hacerlo" de las especificaciones de diseño. Inicialmente, la representación describe una visión sistemática y holística del software. Posteriores, refinamientos de diseño conducen a una representación que se acerca al código fuente.

El diseño del software puede verse desde dos perspectivas: la técnica y la de gestión del proyecto. Desde el punto de vista técnico, el diseño comprende cuatro actividades: diseño de los datos, diseño arquitectónico, diseño procedimental y diseño de las interfaces. Desde el punto de vista de gestión del proyecto, el diseño va del diseño arquitectónico (diseño preliminar o diseño de alto nivel) al diseño

detallado (diseño de bajo nivel).

Desde el punto de vista de la gestión, el nivel de diseño arquitectónico o preliminar se focaliza en las funciones y estructuras de las componentes que conforman el sistema software. El nivel de diseño detallado se ocupa del refinamiento de la representación arquitectónica que lleva a una estructura de datos detallada y a las representaciones algorítmicas que se usan para cada componente modular del software.

El proceso de diseño del software comienza con la actividad de Realizar el Diseño Arquitectónico. Esta actividad genera la **Descripción de Diseño Arquitectónico del Software** en donde se describe el diseño de cada una de las componentes software, se especifican los datos, las relaciones y restricciones y se definen todos los interfaces externos (usuario, software y hardware) y los interfaces internos (entre las componentes).

La última actividad del proceso de diseño es Realizar el Diseño Detallado, donde se genera la **Descripción de Diseño del Software (DDS)** que especifica la estructura de los datos, los algoritmos y la información de control de cada componente software, y los detalles de los interfaces (usuario, hardware y software).

El diseño detallado se deriva del diseño preliminar, en consecuencia sus correspondientes actividades se realizan en secuencia mientras que el resto de las actividades de este proceso (analizar el flujo de información, diseñar la base de datos, diseñar los interfaces, seleccionar o desarrollar algoritmos) se ejecutan en paralelo. Estas últimas producen refinamientos del diseño que son también entradas a la actividad de diseño detallado.

Una actividad relevante es la de diseño de la base de datos. Esta comprende el diseño conceptual, lógico y físico de la base de datos. Los requisitos se modelizan dentro de un esquema externo que describe las entidades de datos, atributos, relaciones y restricciones. Los distintos esquemas externos se integran en un esquema conceptual único. El esquema conceptual se *aplica* entonces en un esquema lógico dependiente de la implementación. Finalmente, se definen las estructuras físicas de datos y los caminos de acceso. El resultado de esta actividad es la **Descripción de la Base de Datos**.

Como ya se ha mencionado, desde el punto de vista técnico y en el contexto de los diseños preliminar y detallado, se llevan a cabo varias actividades de diseño diferentes: diseño de datos, diseño arquitectónico, diseño procedimental y diseño del interfaz.

El diseño de datos transforma el modelo del campo de información, creado durante el análisis, en las estructuras de datos que se van a requerir para implementar el software. El diseño arquitectónico define las relaciones entre los principales elementos estructurales del programa. El objetivo principal de este diseño es desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos. Además, el diseño arquitectónico mezcla la estructura de programas y la estructura de datos y define los interfaces que facilitan el flujo de los datos a lo largo del programa. El diseño procedimental transforma los elementos estructurales en una descripción procedimental del software; se realiza después de que se ha establecido la estructura del programa y de los datos. El diseño del interfaz establece principalmente la disposición y los mecanismos para la interacción hombre-máquina.

La figura 4.3. representa las actividades del proceso de diseño y los documentos que las mismas producen. El documento de Descripción del Diseño del Software contiene los datos consolidados de la Descripción de la Arquitectura del Software, del Flujo de Información, de la Base de Datos, de las Interfaces y de los Algoritmos y forma parte de la configuración del software.

El diseño es el proceso en el que se asienta la calidad del desarrollo del software. El diseño produce las representaciones del software de las que puede evaluarse su calidad. El diseño es la única forma mediante la cual se puede traducir con precisión los requisitos del cliente en un producto o sistema acabado. El diseño de software sirve como base de todas las posteriores etapas del desarrollo y del proceso de mantenimiento. Sin diseño, se puede construir un sistema inestable; un sistema que falle cuando se realicen pequeños cambios; un sistema que pueda ser difícil de probar; un sistema cuya calidad no pueda ser evaluada hasta más adelante en el proceso de ingeniería del software, cuando quede poco tiempo y se haya gastado ya mucho dinero.

El proceso de diseño en la Ingeniería del Software conduce a un buen diseño mediante la aplicación de principios fundamentales de diseño (abstracción, refinamiento sucesivo, modularidad, estructura jerárquica de los módulos, estructura de los datos, jerarquía de control, procedimiento realizado por capas funcionales, ocultamiento de información), de métodos sistemáticos y de una adecuada revisión.

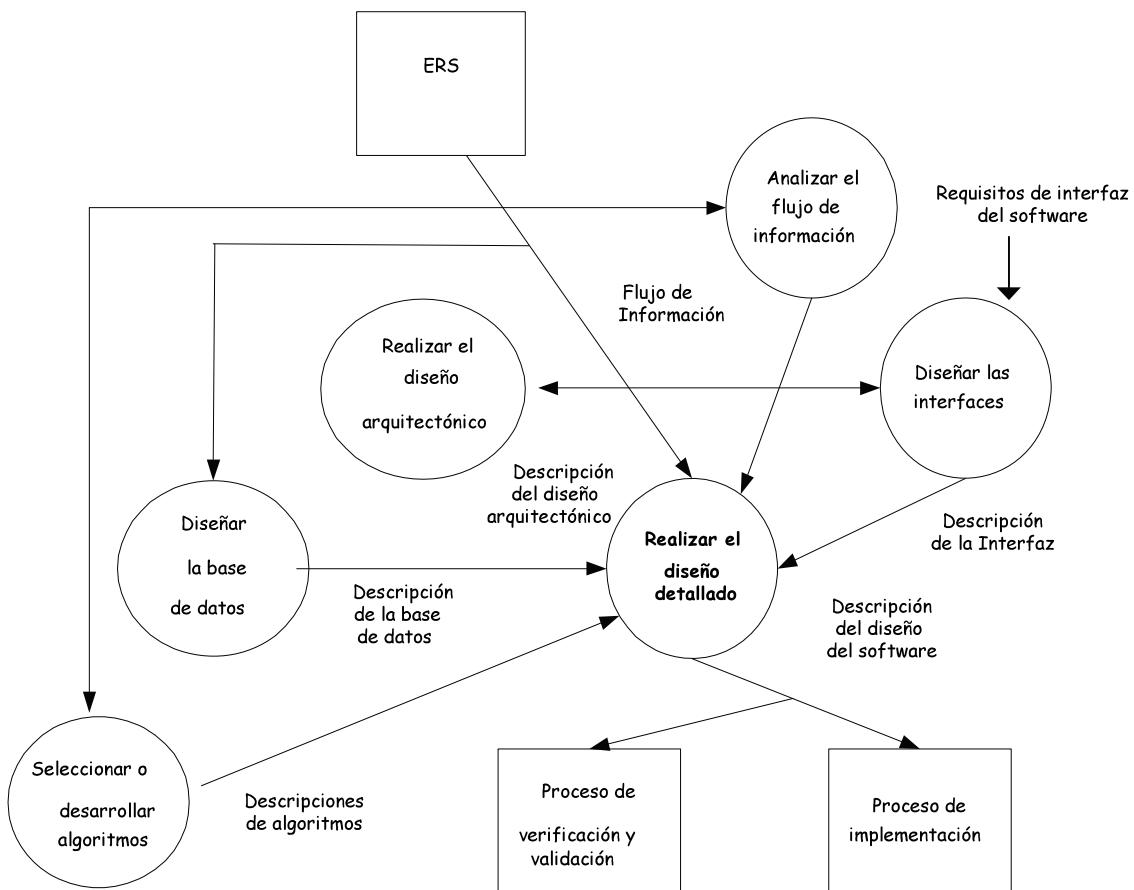


Figura 4.3. Representación del Proceso de Diseño

Para evaluar la calidad de una representación del diseño se deben tener en cuenta, entre otros, los siguientes criterios de calidad del diseño:

1. Un diseño debe exhibir una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
2. Un diseño debe ser modular; esto es, el software debe estar dividido de forma lógica en elementos que realicen funciones y subfunciones específicas.
3. Un diseño debe contener representaciones distintas y separadas de los datos y de los procedimientos.
4. Un diseño debe llevar a módulos (por ejemplo: subrutinas o procedimientos) que exhiban características funcionales independientes.
5. Un diseño debe llevar a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.
6. Un diseño debe obtenerse mediante un método que sea reproducible y que esté conducido por la información obtenida durante el análisis de los requisitos del software

La modularidad se ha convertido en un enfoque ampliamente aceptado, ya que un diseño modular reduce la complejidad, facilita los cambios (un aspecto crítico de la facilidad de mantenimiento del software) y produce como resultado una implementación más sencilla permitiendo el desarrollo paralelo de las diferentes partes de un sistema.

En la arquitectura del software, para la definición de módulos se utiliza los conceptos de abstracción y de ocultamiento de información que derivan en el concepto de independencia funcional.

La independencia funcional se adquiere desarrollando módulos con una función definida y específica (máxima cohesión) y una aversión a una excesiva interacción con otros módulos (mínimo acoplamiento). Es decir, se trata de diseñar software de forma que cada módulo se centre en una subfunción específica de los requisitos y tenga un interfaz sencillo, cuando se ve desde otras partes de la estructura del software. Así, la independencia se mide con dos criterios cualitativos: la cohesión y el acoplamiento. La cohesión es una medida de la fortaleza funcional relativa de un módulo. El acoplamiento es una medida de la interdependencia relativa entre los módulos de una estructura de programa.

La notación de diseño, junto con los conceptos de la programación estructurada, permite al diseñador representar los detalles procedimentales, facilitando su traducción al código que puede realizarse automáticamente a través de una herramienta CASE (Computer Aided Software Engineering).

Actividades a realizar:

- Realizar el diseño arquitectónico.
- Analizar el flujo de información.
- Diseñar la base de datos (si se aplica).
- Diseñar los interfaces.
- Seleccionar o Desarrollar algoritmos (si se aplica).
- Realizar el diseño detallado.

Documentos de salida:

- Descripción de diseño del software.
- Descripción de la arquitectura del software.
- Descripción del flujo de información.
- Descripción de la base de datos.
- Descripción de las interfaces.
- Descripción de los algoritmos.

Técnicas a utilizar:

- Técnicas Orientadas a los Procesos
 - Diseño Estructurado.
 - Análisis de Transformación.
 - Análisis de Transacción.
 - Diseño del Diálogo de los Interfaces
 - Diseño Lógico o Diseño del Perfil.
 - HIPO (Hierarchy Input Process Output).
- Técnicas Orientadas a los Datos
 - Modelo Lógico de Datos.
 - Modelo Físico de Datos.
 - Warnier.
 - Jackson.
- Técnicas Orientadas a los Objetos
 - Modelo de Clases/Objetos.
 - Diagrama de Módulos.
- Técnicas de Diseño de Bajo Nivel
 - Programación Estructurada.
 - Diagramas Arborescentes.
 - Diagramas de Chapin.
 - Programación Orientada a Objetos.
 - Diagrama de Procesos.
 - Warnier.
 - Jackson (JSD - Jackson System Development).
- Técnicas de Prototipación.
- Técnicas de Refinamiento.

PROCESO DE IMPLEMENTACIÓN

Este proceso transforma la representación del diseño detallado de un producto software a una realización en un lenguaje de programación apropiado. El Proceso de Implementación produce el código fuente, el código de la base de datos (si se aplica) y la documentación, que constituyen la manifestación física del diseño de acuerdo a los estándares y metodologías del proyecto. Además, en este proceso se debe integrar el código y la base de datos. En el caso de que el sistema conste de componentes hardware y software, se debe planificar y realizar la integración del sistema. La salida de este proceso está sujeta a las pruebas de verificación y validación adecuadas. El código y la base de datos junto con la documentación producida durante los procesos previos son la primera representación completa del producto software.

Actividades a realizar:

- Crear los datos de prueba.
- Crear el código fuente.
- Generar el código objeto.
- Crear la documentación de operación.
- Planificar la integración.
- Realizar la integración.

Documentos de salida:

- Datos para las pruebas.
- Documentación del sistema.
- Documentación de usuario.
- Plan de integración.
- Sistema software integrado.

Técnicas a utilizar:

- Warnier.
- Jackson.
- Lenguajes de programación.

4.7. PROCESOS DE POST-DESARROLLO

Son los procesos que se deben realizar para instalar, operar, soportar, mantener y retirar un producto software. Se realizan después de la construcción del software. Es decir, se aplican a las últimas fases del ciclo de vida del software.

Una vez terminada la prueba del software, éste está casi preparado para ser entregado a los usuarios finales. Sin embargo, antes de la entrega se llevan a cabo una serie de actividades de garantía de calidad para asegurar que se han generado y catalogado los registros y documentos internos adecuados, que se ha desarrollado una documentación de alta calidad para el usuario y que se han establecido los mecanismos apropiados de control de configuraciones. Entonces, el software ya puede ser distribuido a los usuarios finales.

Tan pronto como se entregue el software a los usuarios finales, el trabajo del ingeniero del software cambia. En ese momento, el enfoque pasa de la construcción al mantenimiento; corrección de errores, adaptación al entorno y mejora de la funcionalidad. En todos los casos, la modificación del software no sólo afecta al código, sino también a la configuración entera; es decir, todos los documentos, datos y programas desarrollados en la fase de planificación y desarrollo.

PROCESO DE INSTALACIÓN

Implica el transporte y la instalación de un sistema software desde el entorno de desarrollo al entorno de destino. Incluye la carga, si es necesaria, de la base de datos, las modificaciones necesarias del software, las comprobaciones en el entorno de destino y la aceptación del cliente. Si durante la instalación surge un problema se identifica e informa acerca de él.

El Proceso de Instalación verifica que se implemente la configuración adecuada del software y culmina con la aceptación formal del mismo por parte del cliente conforme a lo especificado en el Plan de Gestión del Proceso Software y la realización con éxito de la prueba de aceptación del usuario.

Actividades a realizar:

- Planificar la instalación.
- Distribuir el software.
- Instalar el software.
- Cargar la base de datos (si se aplica).
- Aceptar el software en el entorno de operación.
- Realizar las actualizaciones (instalar el software probado).

Documentos de salida:

- Plan de instalación del software.
- Informe de instalación.

PROCESO DE OPERACIÓN Y SOPORTE

Involucra la operación del sistema por parte del usuario y el soporte continuo al usuario que incluye asistencia técnica, consultas con el usuario y registro de las peticiones de soporte en el **Histórico de Peticiones de Soporte**. Así, este proceso puede desencadenar la actividad del Proceso de Mantenimiento que provee información re-entrante al ciclo de vida del software.

Actividades a realizar:

- Operar el sistema.
- Proveer de asistencia técnica y consultas.
- Mantener el histórico de peticiones de soporte.

Documentos de salida:

- Histórico de peticiones de soporte.

PROCESO DE MANTENIMIENTO

Se interesa por los errores, defectos, fallos, mejoras y cambios del software. Un requisito de mantenimiento del software inicia los cambios del ciclo de vida del software; éste se reasigna y ejecuta.

El mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas por la evolución del entorno del software y a las modificaciones debidas a los cambios de los requisitos del cliente dirigidos a reforzar o a ampliar el sistema. El proceso de mantenimiento vuelve a aplicar los pasos del ciclo de vida, pero en el contexto del software ya existente; de este modo, se considera el Proceso de Mantenimiento como iteraciones de desarrollo.

Durante el mantenimiento se encuentran tres tipos de cambios:

- Corrección: incluso llevando a cabo las mejores actividades de garantía de calidad, es muy probable que el cliente descubra defectos en el software. El mantenimiento correctivo cambia el software para corregir los defectos.
- Adaptación: con el paso del tiempo es probable que cambie el entorno original (por ejemplo, la Unidad Central de Proceso, el Sistema Operativo, los periféricos) para el que se desarrolló el software. El mantenimiento adaptativo consiste en modificar el software para acomodarlo a los cambios de su entorno externo.
- Mejora: conforme utilice el software, el cliente/usuario puede descubrir funciones adicionales que podría interesar que estuvieran incorporadas en el software. El mantenimiento perfectivo amplía el software más allá de sus requisitos funcionales originales.

La salida de este proceso son **Recomendaciones de Mantenimiento** que entran al ciclo de vida del software en el Proceso de Exploración de Conceptos para mejorar la calidad del sistema software.

Actividad a realizar:

- Reaplicar el ciclo de vida del software.

Documentos de salida:

- Orden de mantenimiento.
- Recomendaciones de mantenimiento.

PROCESO DE RETIRO

Es la jubilación de un sistema existente de su soporte activo o de su uso mediante el cese de su operación o soporte, o mediante su reemplazamiento tanto por un nuevo sistema como por una versión actualizada del sistema existente. Si el sistema en uso, sea manual o automatizado, se reemplaza por un nuevo sistema se requiere un período de operación dual, denominado ensayo en paralelo. En este período se utiliza el sistema en retiro para los resultados oficiales, mientras se completa la preparación del nuevo sistema para la operación formal. Es un período de formación del usuario sobre el nuevo sistema y de validación del mismo.

Actividades a realizar:

- Notificar al usuario.
- Conducir operaciones en paralelo (si se aplica).
- Retirar el sistema.

Documentos de salida:

- Plan de retiro.

4.8. PROCESOS INTEGRALES DEL PROYECTO

Son procesos simultáneos y complementarios a los procesos orientados al desarrollo. Incluyen actividades imprescindibles para que el sistema construido sea fiable (procesos de verificación y validación, gestión de la configuración) y sea utilizado al máximo de sus capacidades (procesos de formación, documentación).

Los Procesos Integrales comprenden dos tipos de actividades:

- aquellas que se realizan discretamente y se aplican dentro de un ciclo de vida del software, y
- aquellas que se realizan para completar otra actividad. Estas son actividades que se invocan (llamadas como a una subrutina) y no se aplican dentro del modelo de ciclo de vida del software para cada instancia.

PROCESO DE VERIFICACIÓN Y VALIDACIÓN

Abarca la planificación y la realización de todas las tareas de verificación, incluyendo pruebas de verificación, revisiones y auditorías, y todas las tareas de validación, incluyendo pruebas de validación, que se ejecutan durante el ciclo de vida del software para asegurar que se satisfacen todos los requisitos del software.

Una actividad útil para la verificación y la validación del software es la prueba del

software. Constituye el proceso de ejecución del software con determinados datos de entrada, denominados juego de ensayo, para observar los resultados que produce y compararlos con los resultados que teóricamente (de acuerdo con las especificaciones) debería producir, para esos datos de entrada, con el objeto de detectar posibles fallos. Las pruebas del software sólo podrán realizarse cuando, en el proceso de desarrollo, ya exista código ejecutable.

La depuración es un proceso frecuentemente asociado a las pruebas (y a algunas otras actividades de verificación y validación) que consiste en tratar de deducir dónde están los defectos en el software que provocan que éste no funcione adecuadamente. Estudia los resultados de pruebas y otras actividades de control para intentar buscar qué está mal en el software.

Este proceso se aplica en cada proceso y producto del ciclo de vida del software.

Actividades a realizar:

- Planificar la verificación y validación.
- Ejecutar las tareas de verificación y validación.
- Recoger y analizar los datos de las métricas
- Planificar las pruebas.
- Desarrollar las especificaciones de las pruebas.
- Ejecutar las pruebas.

Documentos de salida:

- Plan de verificación y validación.
- Informes de evaluación.
- Plan de pruebas.
- Especificación de las pruebas.
- Informe resumen de las pruebas.
- Software probado.

Técnicas a utilizar:

- Técnicas de Prueba de Caja Blanca.
 - Cobertura de Sentencias.
 - Cobertura de Decisión o de Ramificación.
 - Cobertura de Condición.
 - Cobertura de Decisión/Condición.
 - Cobertura de Condición Múltiple.
- Técnicas de Prueba de Caja Negra.

- Partición de Equivalencias.
- Análisis de Valores Límites.
- Gráficos de Causa-Efecto.
- Conjetura de Errores.
- Revisiones Formales.
- Auditorías.

PROCESO DE GESTIÓN DE LA CONFIGURACIÓN

Este proceso involucra un conjunto de actividades desarrolladas para gestionar los cambios durante todo el ciclo de vida del software. Identifica la estructura de un sistema (qué rutinas, módulos, datos, ficheros, etc., lo componen) en un momento dado (incluso cuando se está desarrollando) a lo que se denomina **Configuración del Sistema**. Su objetivo es el control de los cambios en el sistema, mantener su coherencia y su "rastreabilidad" o "trazabilidad", y poder realizar auditorías de control sobre la evolución de las configuraciones.

La gestión de la configuración realiza las siguientes funciones:

- identificación de la configuración de un sistema: descripción documentada de las características reales del sistema en un determinado momento,
- control de la configuración: establece la configuración inicial o básica y controla los cambios en los elementos de la misma,
- informes del estado de la configuración, y
- auditorias de configuración: revisiones independientes de la configuración para comprobar que los elementos de la configuración cumplen los requisitos de configuración establecidos.

En resumen, la gestión de la configuración del software identifica los elementos de un proyecto de desarrollo del software (especificaciones, código, planes, etc.) y provee tanto el control de los elementos identificados como la generación de informes de estado de la configuración. Todo esto con el objeto de conseguir visibilidad en la gestión y responsabilidad de la misma durante el ciclo de vida del software.

Actividades a realizar:

- Planificar la gestión de la configuración.
- Realizar la identificación de la configuración.
- Realizar el control de la configuración.
- Realizar la información del estado de la configuración.

Documentos de salida:

- Plan de gestión de configuración del software.
- Orden de cambio de ingeniería.
- Cambio de estado.
- Informe de estado.

PROCESO DE DESARROLLO DE DOCUMENTACIÓN

El Proceso de Desarrollo de Documentación para el desarrollo y uso del software es el conjunto de actividades que planifican, diseñan, implementan, editan, producen, distribuyen y mantienen los documentos necesarios para los desarrolladores y los usuarios.

Actividades a realizar:

- Planificar la documentación.
- Implementar la documentación.
- Producir y distribuir la documentación.

Documentos de salida:

- Plan de documentación

PROCESO DE FORMACIÓN

Incluye la planificación, desarrollo, validación e implementación de los programas de formación de desarrolladores, personal de soporte técnico y clientes y la elaboración de los materiales de formación adecuados.

Para conseguir una utilización efectiva del sistema software, se debe proporcionar a los usuarios del sistema instrucciones, guía y ayuda para el entendimiento de las capacidades del sistema y de sus limitaciones. Por ello, es imprescindible la formación de los usuarios en el nuevo sistema.

El desarrollo de productos software de calidad depende en gran medida de los conocimientos de las personas y del personal especializado involucrados en el proyecto. Por ello, es esencial la formación de los desarrolladores y personal de soporte técnico.

Actividades a realizar:

- Planificar el programa de formación.
- Desarrollar los materiales de formación.
- Validar el programa de formación.
- Implementar el programa de formación.

Documentos de salida:

- Plan de formación.

5. MAPA DE ACTIVIDADES DE UN PROYECTO

Como ya se ha mencionado, no todos los proyectos de desarrollo de software son iguales. De hecho, ya se ha visto, que al inicio del proceso, un momento crítico es aquél de la decisión de qué ciclo de vida se elegirá para el proyecto en cuestión. Una vez que se ha hecho tal selección, y guiado en cierto modo por ella, se debe adaptar el proceso software genérico al modelo de ciclo de vida elegido. Es decir, establecer el mapa de actividades del proyecto. El proceso software visto en el apartado anterior es un proceso genérico, que obligatoriamente debe adaptarse para cada proceso. La adaptación se lleva a cabo precisamente mediante el establecimiento del mapa de actividades.

El mapa de actividades es una tabla donde se marcan qué actividades del proceso software genérico se van a ejecutar para un determinado proyecto. Existen desarrollados ya ciertos mapas de actividades para los tipos de proyectos más comunes (proyecto grande, proyecto pequeño, etc.), que pretenden facilitar la labor del jefe de proyecto. En cualquier caso, las tablas existentes se deben siempre comprobar y adaptar para un proyecto concreto. Hay que insistir en que no hay dos proyectos de desarrollo de software iguales.

El tipo de proyecto está muy estrechamente relacionado con el ciclo de vida. Por ejemplo, un proyecto pequeño casi siempre se corresponde con un ciclo de vida en cascada; un proyecto medio innovador, se suele corresponder con un ciclo de vida con prototipado; etc.

Por tanto, como puede verse en la Tabla 5.1., el mapa de actividades es una tabla de dos entradas. Una entrada es el proceso software con sus actividades y la otra entrada el ciclo de vida elegido descompuesto en sus etapas.

Proceso software	Requisitos	Diseño
Actividad 1			
Actividad 2			
.			
.			
.			
Actividad n			

Figura 5.1. Mapa de Actividades

En el mapa de actividades simplemente se marcan con una cruz las actividades que se llevarán a cabo. Además, puede incluirse más información del tipo: importancia de la actividad, si está marcada con un + significa mucha y si está marcada con un - significa normal; tipo de actividad, obligatoria (marcada con una O) o condicional (marcada con una C).

A continuación, en la Tabla 5.1, aparece un ejemplo de mapa de actividades para un proyecto que ha elegido un ciclo de vida en cascada con nueve etapas definidas del siguiente modo:

- **Factibilidad (FA).** Definir el producto software, y determinar su factibilidad en el ciclo de vida y superioridad con respecto a productos alternativos.
- **Requisitos (RS).** Una completa especificación validada de las funciones requeridas, interfaces, y rendimiento para el producto software.
- **Diseño del Producto (DP).** Una completa especificación verificada de la arquitectura del hardware-software, estructura de control, y estructura de datos para el producto, junto con otros componentes necesarios como los manuales del usuario preliminar y los planes de prueba.
- **Diseño Detallado (DD).** Una completa especificación verificada de la estructura de control, estructura de datos, relaciones de interfaz, tamaño, algoritmos claves, y suposiciones de cada componente de programa.
- **Codificación (CO).** Un completo conjunto verificado de componentes del programa.
- **Integración (IN).** Un adecuado funcionamiento del producto software compuesto de los componentes software.
- **Implementación (IM).** Un sistema hardware-software funcionando operacionalmente a pleno, incluyendo tales objetivos como conversión del programa y datos, instalación, y entrenamiento.
- **Operación y Mantenimiento (OM).** Un funcionamiento actualizado del sistema hardware-software. Este subobjetivo se repite para cada actualización.
- **Retiro (RE).** Una transición adecuada de las funciones realizadas para el producto y sus sucesores (si existe).

ACTIVIDADES DE LOS PROCESOS	FA	RS	DP	DD	CO	IN	IM	OM	RE
Proceso de Selección de un MCVS									
- Identificar los posibles MCVS	x								
- Seleccionar un modelo para el proyecto.	x								
Proceso de Iniciación, Planificación y Estimación del Proyecto	y								
- Establecer la matriz de actividades para el MCVS .	x								
- Asignar los recursos del proyecto.	x	x	x	x	x	x	x	x	x
- Definir el entorno del proyecto.	x	x		x					
- Planificar la gestión del proyecto.	x	x							
Proceso de Seguimiento y Control del Proyecto									
- Analizar riesgos.	x	x	x	x	x	x	x		
- Realizar la planificación de contingencias.		x	x	x	x	x	x		
- Gestionar el proyecto.	x	x	x	x	x	x	x	x	x
- Implementar el sistema de informes de problemas.	x	x	x	x	x	x	x	x	x
- Archivar registros.		x	x	x	x	x	x	x	x
Proceso de Gestión de Calidad del Software									
- Planificar la garantía de calidad del software.		x	x						
- Desarrollar métricas de calidad.		x	x	x					
- Gestionar la calidad del software.	x	x	x	x	x	x	x	x	x
- Identificar necesidades de mejora de la calidad.	x	x	x	x	x	x	x	x	x
Proceso de Exploración de Conceptos									
- Identificar las ideas o necesidades.	x								
- Formular las soluciones potenciales.	x	x							
- Dirigir los estudios de viabilidad.	x	x							
- Planificar la transición del sistema (si se aplica).	x	x							x
- Refinar y Finalizar la idea o necesidad.	x								
Proceso de Asignación del Sistema									
- Analizar las funciones del sistema.		x	x						
- Desarrollar la arquitectura del sistema.		x	x						
- Descomponer los requisitos del sistema.	x								
Proceso de Análisis de Requisitos									
- Definir y Desarrollar los requisitos de software.		x	x						
- Definir los requisitos de interfaz.		x	x						
- Priorizar e Integrar los requisitos del software.	x	x							

ACTIVIDADES DE LOS PROCESOS	FA	RS	DP	DD	CO	IN	IM	OM	RE
Proceso de Diseño									
- Realizar el diseño preliminar. - Analizar el flujo de información. - Diseñar la base de datos (si se aplica). - Diseñar las interfaces. - Seleccionar o Desarrollar algoritmos (si se aplica). - Realizar el diseño detallado.		x	x	x					
Proceso de Implementación e Integración			x	x					
- Crear los datos de prueba. - Crear el código fuente. - Generar el código objeto. - Crear la documentación de operación. - Planificar la integración. - Realizar la integración.		x	x	x	x				
Proceso de Instalación y Aceptación						x			
- Planificar la instalación. - Distribuir el software. - Instalar el software. - Cargar la base de datos (si se aplica). - Aceptar el software en el entorno de operación. - Realizar las actualizaciones.						x	x	x	x
Proceso de Operación y Soporte							x		
- Operar el sistema. - Proveer de asistencia técnica y consultas. - Mantener el histórico de peticiones de soporte.							x		
Proceso de Mantenimiento							x		
- Realizar el mantenimiento correctivo. - Reaplicar el ciclo de vida del software.							x		
Proceso de Retiro							x	x	
- Notificar al usuario. - Conducir operaciones en paralelo (si se aplica). - Retirar el sistema.							x	x	
Proceso de Verificación y Validación							x	x	x
- Planificar la verificación y validación. - Ejecutar las tareas de verificación y validación. - Recoger y Analizar los datos de las métricas. - Planificar las pruebas. - Desarrollar las especificaciones de las pruebas. - Ejecutar las pruebas.	x	x	x	x	x	x	x	x	x

ACTIVIDADES DE LOS PROCESOS	FA	RS	DP	DD	CO	IN	IM	OM	RE
Proceso de Configuración									
- Planificar la gestión de configuración.	x	x							
- Realizar la identificación de la configuración.	x	x	x	x	x	x	x	x	x
- Realizar el control de la configuración.		x	x	x	x	x	x	x	x
- Realizar la información del estado de la configuración.		x	x	x	x	x	x	x	x
Proceso de Documentación									
- Planificar la documentación.	x	x		x	x				
- Implementar la documentación.				x	x	x	x		
- Producir y Distribuir la documentación.						x	x		
Proceso de Formación									
- Planificar el programa de formación.	x	x		x	x	x	x		
- Desarrollar los materiales de formación.		x	x	x	x	x	x		
- Validar el programa de formación.			x	x	x	x	x		
- Implementar el programa de formación.				x	x	x	x		

Tabla 5.2. Matriz de Actividades para un ciclo de vida en cascada con nueve etapas

Como puede verse en la Tabla 5.2, el mapa marca qué actividades del Proceso Software deberán realizarse en cada una de las etapas del Ciclo de Vida. Existen actividades que es necesario realizarlas una única vez (por ejemplo, la selección de un modelo de ciclo de vida). Sin embargo, existen otras actividades que se realizan en cada una de las etapas (por ejemplo, gestionar el proyecto). Obviamente, dependiendo del ciclo de vida elegido el mapa de actividades variará.

Por tanto, el mapa de actividades es el primer paso para conseguir una organización del proyecto que lleve a su gestión. A partir del mapa, puede pasarse a una estimación del tiempo y costo de cada una de las actividades, y por tanto, del proyecto global; a una asignación de recursos para cada actividad, etc.

6. BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

- [Böehm, 1976] Böehm, B. **Software Engineering**. IEEE Trans. Computers C-25, 12. Dec. 1976.
- [Böehm, 1986] Böehm, B. W. **A Spiral Model of Software Development and Enhancement**. ACM Software Engineering Notes 11, 4. 1986.
- [Böehm, 1988] Böehm, B. W. **A Spiral Model of Software Development and Enhancement**. Computer, May 1988, pp. 61-72.
- [Budde, 1984] Budde, R., K. Kuhlenkamp, L. Mathiassen, and H. Zullighoven. **Approaches to Prototyping**. Springer-Verlag, New York 1984.
- [Humphrey, 1995] Humphrey, W. S. **A discipline for Software Engineering**. Addison-Wesley. Readings, Massachusetts, EE.UU. 1995.
- [IEEE, 1989] IEEE Std. 1074-1989. IEEE Standard Software Life Cycle Processes. 1989.
- [McCracken, 1982] McCracken, D.D., and Jackson, M.A. **Life-Cycle Concept Considered Harmful**. ACM SW Eng. Notes, Apr. 1982, pp. 29-32.
- [Pressman, 1993] Pressman, R. S. **Ingeniería del Software: Un enfoque práctico**. 3 ed. McGraw-Hill, Madrid. 1993.
- [Royce, 1970] Royce, W. W. **Managing the Development of Large Software Systems**. Proc. 9th. Intern. Conf. Software Engineering. IEEE Computer Society, 1987, 328-338. Originally published in Proc. WES-CON, 1970.

BIBLIOGRAFÍA OPTATIVA

- [Balzer, 1981] Balzer, R. **Transformational Implementation: An Example**. IEEE Trans. Software Eng. SE-7, 1. 1981.
Para el ciclo de vida con prototipado.
- [Balzer, 1982] Balzer, R., N. Goldman, and D. Wile. **Operational Specifications as the Basis for Rapid Prototyping**.
ACM Software Engineering Notes 7,5. 1982.
Para especificaciones operativas.
- [Balzer, 1983 (a)] Balzer, R., D. Cohen, M. Feather, N. Goldman, W. Swartout and D. Wile. **Operational Specifications as the Basis for Specification Validation**. In Theory and Practice of Software Technology, Ferrari, Bolognani, and Goguen, eds. North-Holland, 1983.
Para especificaciones operativas.

- [Basili, 1975] Basili, V.R., and A.J. Turner. **Iterative Enhancement: A Practical Technique for Software Development.** IEEE Trans. Software Eng. SE-1, 4. Dec. 1975.
Para el modelo de ciclo de vida de refinamiento sucesivo.
- [Bauer, 1976] Bauer, F. L. **Programming as an Evolutionary Process.** Proc. 2nd. Intern. Conf. Software Engineering. IEEE Computer Society, Jan. 1976.
Para especificaciones operativas.
Para ciclo de vida de transformación continua.
- [Biggerstaff, 1984] **Special Issues on Software Reusability.** T. Biggerstaff and A. Perlis, eds. IEEE Trans. Software Eng. SE-10, 5. Sept. 1984.
Para el modelo de ciclo de vida por ensamblaje de componentes reutilizables.
- [Böehm, 1984] Böehm, B. W., T. Gray, and T. Seewaldt. **Prototyping vs. Specifying: A Multi-project Experiment.** Proc. 7th. Intern. Conf. Soft. Engr. 1984.
Variación del modelo de ciclo de vida en cascada.
Para el ciclo de vida con prototipado.
- [Curtis, 1987] Curtis, B., H. Krasner, V. Shen, and N. Iscoe. **On Building Software Process Models Under the Lamppost.** Proc. 9th. Intern. Conf. Software Engineering. IEEE Computer Society, April 1987.
Potencialidades y limitaciones de la automatización del proceso software.
- [Deming, 1982] Dening, W.E. **Out of the crisis.** Center for Advanced Engineering. Massachussets Institute of Technology. Cambridge, Massachusetts, EE.UU. 1982.
Sobre la definición del proceso software en una organización.
- [Distaso, 1980] Distaso, J. **Software Management - A Survey of Practice in 1980.** Proceedings IEEE 68, 9. 1980.
Variación del modelo de ciclo de vida en cascada.
- [Goguen, 1986] Goguen, J. **Reusing and Interconnecting Software Components.** Computer 19, 2. Feb. 1986.
Para el modelo de ciclo de vida por ensamblaje de componentes reutilizables.
- [Hekmatpour, 1987] Hekmatpour, S. **Experience with Evolutionary prototyping in a Large Software Project.** ACM Software Engineering Notes 12, 1. 1987.
Para el ciclo de vida con prototipado.
- [Hoffnagel, 1975] Hoffnagel, G. F., and W. Beregi. **Automating the Software Development Process.** IBM Systems Journal. 24, 2. 1985.
Potencialidades y limitaciones de la automatización del proceso software.

- [Huseth, 1986] Huseth, S., and D. Vines. **Describing the Software Process.** Proc. 3rd. Intern. Software Process Workshop. IEEE Computer Society, 1986.
Potencialidades y limitaciones de la automatización del proceso software.
- [Lehman, 1984 (a)] Lehman, M. M., V. Stenning, and W. Turski. **Another Look at Software Development Methodology.** ACM Software Engineering Notes 9, 2. April 1984.
Para modelo de ciclo de vida visto como una transformación continua.
- [Lehman, 1984 (b)] Lehman, M. M. **A Further Model of Coherent Programming Processes.** Proc. Software Process Workshop. IEEE Computer Society, 1984.
Para modelo de ciclo de vida visto como una transformación continua.
- [MIL-STD-2167, 1987] Dept. of Defense. **DRAFT Military Standard: Defense System Software Development.** DOD-STD-2167A.
Estándar para el desarrollo de software para el Departamento de Defensa estadounidense.
- [Neighbors, 1984] Neighbors, J. **The Draco Approach to Constructing Software from Reusable Components.** IEEE Trans. Software Eng. SE-10, 5. Sept. 1984.
Para el modelo de ciclo de vida por ensamblaje de componentes reutilizables.
- [Osterweil, 1987] Osterweil, L. **Software Processes are Software Too.** Proc. 9th. Intern. Conf. Software Engineering. IEEE Computer Society. April 1987.
Potencialidades y limitaciones de la automatización del proceso software.
- [Paultk, 1993] Paultk, M. C., Curtis, B. Chrisis. M. B. **Capability Maturity Model for Software. Versión 1.1.** Software Engineering Institute Technical Report CMU/SEI-93-TR. Pittsburgh, Pennsylvania, EE.UU. 1993.
Descripción del Modelo de Madurez del SEI.
- [Scacchi, 1984] Scacchi, W. **Managing Software Engineering Projects: A Social Analysis.** IEEE Trans. Software Eng. SE-10, 1. Jan, 1984.
Variación del modelo de ciclo de vida en cascada.
- [[Sen, 1982]] Special Issue on Rapid Prototyping. ACM Software Engineering Notes 7, 5. Dec. 1982.
Para variaciones sobre estrategias de prototipado, demostración, reutilización, etc.

- [Tully, 1984] Tully, C. **Software Development Models.** Proc. Software Process Workshop. IEEE Computer Society, 1984.
Para el modelo de ciclo de vida con emisión gradual.
- [Wirth, 1971] Wirth, N. **Program Development by Stepwise Refinement.** Comm. ACM 14, 4. April 1971.
Para el modelo de ciclo de vida de refinamiento sucesivo.
- [Zave, 1984] Zave, P. **The Operational Versus the Conventional Approach to Software Development.** Comm. ACM 27. Feb. 1984.
Para especificaciones operativas.