Capítulo 3

Desarrollo del Modelo

Este capitulo contempla las fases que componen el desarrollo del modelo de sistema crítico para la caracterización de señales. El término caracterización, como efecto de **caracterizar** (*"Determinar los atributos peculiares de alguien o de algo, de modo que claramente se distinga de los demás"*. Real Academia Española 2001). La especificación de los atributos, en este caso de atributos sobre las señales, así como la configuración e interpretación de tales señales corresponde a la implementación detallada de este modelo.

El modelo propuesto plantea la estructura de elaboración de proyectos de software, pero estableciendo unos principios de desarrollo alrededor de la planificación, ciclo de vida y sistema de calidad, dejando para establecer en el marco del problema puntual el análisis y diseño detallado del sistema solución.

Por proyecto se entiende en este capítulo el potencial sistema solución, que se

desarrollará como producto de software en etapas posteriores al modelo.

3.1. Planeación del proyecto

Esta primera fase del modelo se contempla el reconocimiento de las necesidades y los recursos del proyecto a desarrollar. En esta fase se pretende establecer las características del entorno del sistema, así como la definición de ciertos criterios generales que se usarán durante todo el desarrollo del proyecto.

3.1.1. Análisis de factibilidad

Como proyecto de ingeniería, el proyecto a desarrollar debe establecer un grado de confiabilidad respecto a la viabilidad operacional y económica sobre su elaboración. De acuerdo a la naturaleza del modelo de software crítico, el análisis de factibilidad permite medir si existen las condiciones para que se pueda desarrollar un sistema que cumpla con los requerimientos del modelo. Como criterios para evaluar la factibilidad del proyecto se tienen:

3.1.1.1. Factibilidad técnica

Estudia si el trabajo para el proyecto puede desarrollarse con el software y el personal existente, o en caso de necesitar nueva tecnología, cuales son las posibilidades de desarrollarla o adquirirla. En el marco del modelo se debe establecer los criterios y restricciones de la definición de sistema crítico para el proyecto.

3.1.1.2. Factibilidad económica

Investiga si los costos se justifican con los beneficios que se obtienen, o si se ha invertido demasiado, como para no desarrollar el sistema si se considera necesario. También es posible evaluar, hasta que nivel funcional y con qué nivel de confiabilidad se puede desarrollar el sistema, de acuerdo a los recursos disponibles.

3.1.1.3. Factibilidad operacional

La factibilidad operacional investiga si será utilizado el sistema, si los usuarios usaran el sistema y si éste podrá satisfacer las necesidades reales. Realizar este análisis parece una tarea no muy productiva, ya que se asume existe un problema o falencia. Sin embargo, pueden no existir las condiciones para que el sistema pueda operar en conjunto con los usuarios u otros sistemas como se espera. Si bien esta factibilidad es difícil de detectar en las primeras etapas del proyecto, es aconsejable realizarla.

3.1.2. Recursos y actividades

Se debe realizar una identificación de recursos disponibles para el proyecto, según:

- **Tiempo** : asociado a la elaboración de un cronograma, indica los plazos y metas a alcanzar en periodos de tiempo establecidos.
- **Personal** : tanto en disponibilidad física y motivación del recurso humano, así como la capacidad técnica para elaborar el producto.
- **Información** : disponibilidad de acceso a la información y procesos asociados al sistema. Incluye formatos, protocolos, estándares internos, entre otros.

- **Tecnología** : disponibilidad de equipos, software y comunicaciones, de acuerdo a las posibilidades económicas del proyecto.

- **Economía** : como valor nominal para indicar costos del proyecto. No es un criterio de interés para el modelo.

- **Asesoría** : Como recurso no tangible, es necesario tener acompañamiento de personal calificado que conozca el entorno del sistema y que tenga disponibilidad para asesorar en temas relacionados.

- **Experiencia** : experiencia del grupo de desarrollo, como mecanismo para favorecer la gestión del riesgo y la estimación del proyecto.

Las actividades generales que describe el modelo son:

1. Aplicación del Modelo.

    a) Revisión de Análisis Global

    b) Revisión de Diseño Global

2. Diseño de etapas

    a) Análisis detallado

    b) Identificación de etapas: diseño detallado, implementación, pruebas.

3. Diagnóstico y mejoras

3.1.3. Sistema de calidad

El sistema de calidad se propone como mecanismo que contribuye a la gestión del riesgo, ya que favorece en:

- Asegurar que el producto entregado tiene un nivel de calidad aceptable. El concepto de calidad será dado como “el producto funciona como dice debe funcionar”.
- Detectar los errores del proyecto cuanto antes, a fin de evitar que estos ocasionen retrasos en tiempo y sobrecostos.

El control de calidad sugerido por este modelo propone unos **criterios generales**, los cuales ayudarán a *guiar* la calidad del software:

- Proponer un modelo único de ciclo de vida de desarrollo que favorezca la gestión de riesgos.
- Establecer procedimientos de documentación de todas las etapas del desarrollo.
    - Asignación de manejo de configuraciones
    - Asignación de procedimientos para manejo de errores.
- Aspectos técnicos de diseño: Utilidad, Portabilidad, Integridad, Extensibilidad, Reutilización, Eficiencia, Verificabilidad, Facilidad de uso, Compatibilidad, Confiabilidad (corrección y robustez), Modularidad, Legibilidad.
- Utilización de patrones de diseño de software.

- Definición y aplicación de un plan de pruebas del sistema.

A nivel de diseño, contemplar :

- Estabilidad de la plataforma o sistema operativo.
- Confiabilidad de las herramientas de programación.
- Eficiencia y calidad en la utilización de módulos externos.

3.2. Comprensión y Análisis

La comprensión del proceso considera el estudio e investigación de las variables, procesos e interacciones que ocurren en el sistema. El análisis a realizar documenta las necesidades del entorno del problema puntual, enfocándose sobre los requerimientos globales identificados para el modelo, ésto significa que algunas características de la composición del sistema se mencionan, pero se debe obtener el detalle según la aplicación puntual del modelo. Los siguientes puntos consideran el esquema y contenido global propuesto.

3.2.1. Descripción general

El sistema a desarrollar procesa un flujo de datos de entrada correspondiente a un conjunto de canales o señales independientes desde uno o varios sistemas de adquisición. Estas señales deben poseer una referencia o variable independiente (regularmente el tiempo) por la cual se realizará la sincronización de las diversas señales si es necesario. La información recibida se presenta disponible para ser utilizada por un proceso que se encargará de operar las señales y otro para su visualización

o control de parámetros. El procesamiento de la señal es un mecanismo específico del problema a tratar, se considera como una tarea general ya que se pueden contener niveles de detalle que correspondan directamente a técnicas rigurosas de diseño y codificación. La parte de interfaz de manejo corresponde a una primera interfaz de interacción con los usuarios del sistema. Finalmente se encuentra el actuador del sistema como el mecanismo encargado de dar respuesta ante los criterios establecidos para indicar la ocurrencia de eventos.

3.2.2. Análisis de requerimientos

3.2.2.1. Panorama general

Se tiene como propósito la creación de un modelo funcional basado en criterios de seguridad crítica. El sistema tiene la siguiente estructura:

1. Señales e información de entrada.
2. Procesamiento de la señal.
3. Actuador del sistema.
4. Interfaz de manejo.

El modelo presenta una estructura semejante a un sistema de control considerando una unidad de entrada, una de procesamiento o control y una respuesta o salida. Se tiene como propósito realizar el flujo y procesamiento de la señal de entrada, en busca de características consideradas de *interés* según el sistema a evaluar, para términos de este modelo se denominará ***eventualidades***.

3.2.2.2. Funciones del sistema

Las funciones del sistema se indican en la Tabla 3.1.

Ref.#	**Función**	**Categoría**
R.1.1	Recibir señal desde la(s) fuente(s) establecida(s).	Evidente
R.1.2	Proporcionar mecanismos para suministrar las señales a otros procesos que las requieran, de acuerdo a las prioridades de acceso.	Oculta
R.1.3	Restablecer el sistema para la recepción de nuevas señales.	Opcional
R.2.1	Recibir flujo de datos desde la entrada de señales.	Oculta
R.2.2	Detectar eventualidades sobre señales de entrada.	Evidente
R.2.3	Transmitir resultados del procesamiento.	Oculta
R.2.4	Registrar estado / respuesta de la detección.	Oculta
R.2.5	Clasificar parámetros de señales detectadas.	Opcional
R.2.6	Establecer parámetros de operación para el actuador.	Evidente
R.2.7	Enviar solicitud de gestión hacia el actuador.	Oculta
R.2.8	Registrar estado / respuesta de la transacción con el actuador.	Oculta
R.3.1	Recibir solicitud de gestión para el actuador.	Oculta
R.3.2	Activar procedimiento(s) del actuador.	Evidente
R.3.3	Registrar estados / respuesta del actuador.	Oculta
R.4.1	Recibir flujo de datos desde la entrada de señales.	Oculta
R.4.2	Activar Interfaz para interacción con el usuario	Oculta

Cuadro 3.1: Funciones del sistema

3.2.2.3. Atributos del sistema

Los atributos del modelo se consideran en la Tabla 3.2.

3.3. Determinación del Ciclo de Vida

Como Ciclo de Vida para el desarrollo del proyecto, se realizaron evaluaciones comparativas para determinar cual ciclo favorece al proceso de elaboración del software, teniendo en cuenta los siguientes criterios:

- Que facilite mantener procesos documentados sobre el proyecto.
- Que favorezca el análisis y la gestión del riesgo.
- Que permita obtener resultados parciales durante el mismo proceso de elaboración.
- Que facilite la distribución de trabajo en equipo.
- Que permita hacer planificación sobre actividades y tiempo.

Se darán los factores de pros y contras de un modelo de ciclo de vida, denominado entrega por etapas, considerado éste como el ciclo de vida general a seguir por el modelo.

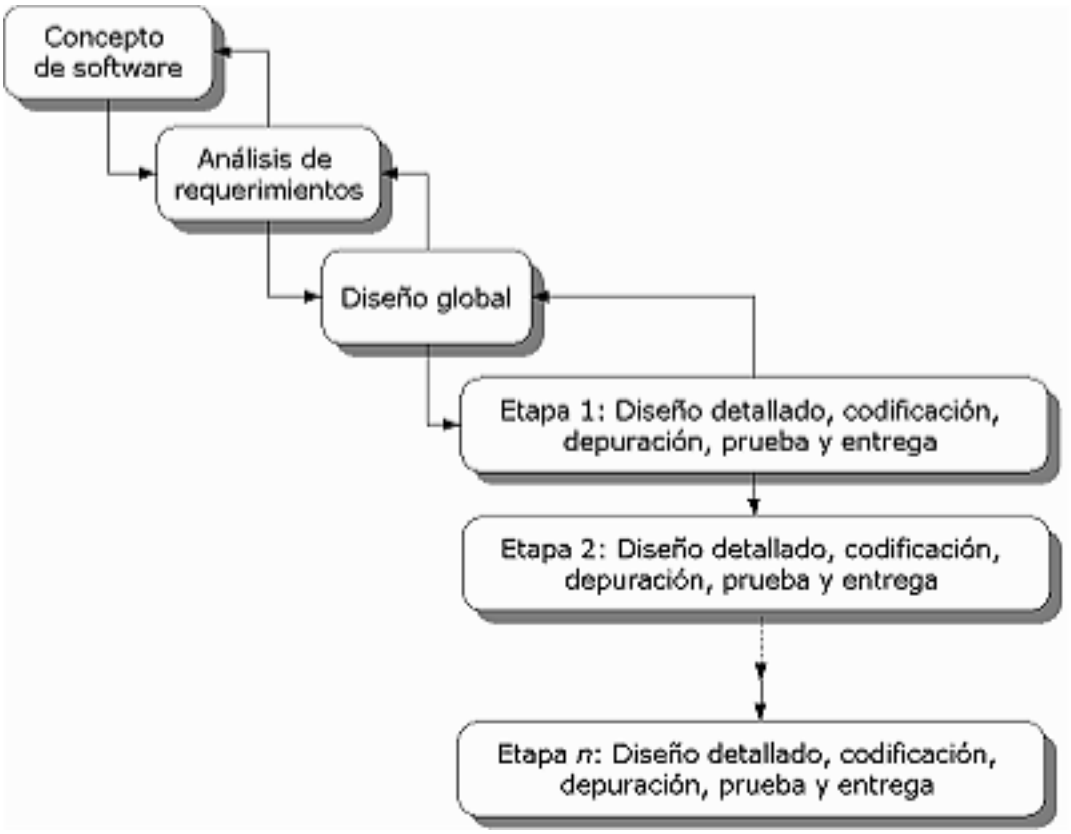
3.3.1. Entrega por etapas

Con el modelo de entrega por etapas, se siguen los pasos del modelo de cascada pasando por la definición del proceso del software, análisis de requerimientos y

Atributo	**Condiciones**
Metáfora de Interfaz	*(detalles)* se utilizarán dos esquemas de interfaces: - De visualización y manejo simple para usuarios (interfaz externa). - De registro de transacciones o de mensajes del sistema (interfaz interna).
Tolerancia a fallas	*(Restricciones de frontera)* Se tratará de acuerdo a la especificación del sistema crítico - Análisis detallado.
Plataforma de ejecución Condiciones de uso Operación	*(Restricciones de frontera)* A seleccionar según el detalle de la aplicación.
Condiciones del modelo	*(detalles)* Características y consideraciones de la aplicación del modelo.

Cuadro 3.2: Atributos del sistema

creación del diseño global de una arquitectura para el programa completo, luego se procede a realizar el diseño detallado, la codificación, depuración y pruebas dentro de cada etapa. Figura 3.1.



(Tomado de [SM96])

Figura 3.1: Ciclo de vida de Entrega por etapas

Este modelo de desarrollo de software, contribuye a disminuir el riesgo total sobre el proyecto debido a que se contempla *liberar* versiones del producto que son parcialmente funcionales. Entre algunos aspectos a favor :

- Favorece la visibilidad del desarrollo ante los usuarios.
- Enfrenta en las primeras etapas detalles importantes de diseño.
- Detección temprana de problemas.
- Ayuda a controlar la ejecución del cronograma del proyecto.
- Facilita mantener la modularidad desde el diseño de las etapas.

En contraposición, tiene asociados riesgos potenciales, como el cambio de prestaciones, lo que constituye la incapacidad o dificultad de modificar su estructura de acuerdo a cambios o adición de requerimientos, que afecten significativamente a etapas previas. Para el caso del modelo propuesto, se subsana este ítem realizando un análisis más detallado como parte de la aplicación del modelo. Adicionalmente el modelo propone una arquitectura general que facilita la modularidad de componentes generales, sin comprometer el funcionamiento y especificaciones de cada uno de estos componentes.

Cada etapa tiene una estructura interna similar al resto. Una estructura recomendada para las etapas es:

- **Descripción general**: Explicación de qué hace o realiza la etapa.
- **Diseño detallado**: Diagramas y conceptos del diseño de los componentes o módulos.
- **Formatos y estructuras**: Estructuras de entrada y salida, formatos de archivo, protocolos y configuraciones.

- **Codificación**: Información general sobre la codificación: lenguajes, métodos, algoritmos.
- **Pruebas**: Formatos de aplicación de los planes de prueba.

3.3.2. Discusión

Aunque el ciclo de vida de entrega por etapas favorece la gestión de riesgos, es posible considerar como parte del diseño global una planificación de las etapas que permita la implementación temprana de los módulos base: clases y bibliotecas comunes de desarrollo, siguiendo con los mecanismos para entrada de datos y posteriormente el núcleo, considerado como el sistema crítico en sí. Como se encuentra definido en cada etapa, se describen los procedimientos de diseño y prueba.

Si bien el ciclo de vida de entrega por etapas se ajusta bien a las condiciones y características del modelo, es posible definir aspectos adicionales que beneficiarían al modelo, tales como:

- **Realizar planificación de las etapas**: identificar un orden que permita la realización temprana de revisión y pruebas, como mecanismos de control de calidad.
- **Análisis específico de requerimientos**: a fin de definir con mayor detalle cuales serán las funcionalidades y limitaciones del sistema.
- **Definición del sistema crítico**: el componente de sistema crítico deberá tener un alcance definido. En rigor, el modelo solo contempla una parte del sistema como crítico, debido al alto costo de desarrollo que podría tener su

aplicación. Sin embargo, el componente crítico deberá tener unas condiciones de *supervivencia* que le permitan operar una parte reducida del sistema completo. La identificación, diseño, codificación y prueba de este componente tendrá mayores exigencias de calidad y verificación, para asegurar confiabilidad.

- **Aplicación de un plan de calidad sobre las etapas**: Aunque los criterios del control de calidad se encuentran sugeridos en este modelo, la utilización de otros esquemas no contraría tales propuestas. La aplicación de un plan de calidad solo es la confirmación de otros procesos que se encuentran ligados a la gestión del riesgo del software.

- **Realizar un proceso de diagnóstico y mejoramiento**: Como una última etapa no definida completamente en tiempo y alcances, se sugiere incluirla como proceso de acompañamiento y despliegue de la herramienta en el ambiente de operación.

3.3.3. Gestión de riesgos

De acuerdo al desarrollo por etapas la gestión del riesgo será definida antes de cada etapa de manera puntual y evaluada finalizando tal etapa, a fin de no acarrear nuevos indicadores de riesgo a las etapas siguientes. Con la planificación de las etapas, se tiene que las primeras etapas tienen una mayor prioridad para el producto, por tanto la gestión debe realizarse intensivamente. La gestión del riesgo se puede dividir en dos subprocesos a ejecutarse como parte del diseño al inicio de una etapa (**Estimación**) y como parte de la implementación al finalizar la misma (**Control**).

3.3.3.1. Estimación

Se busca establecer durante el proceso de diseño qué factores, restricciones del medio o características del producto, pueden representar problemas que generen sobrecostos en tiempo, recursos o personal. Estos riesgos están generalmente asociados al producto específico que se desee obtener. Sin embargo, a nivel de este modelo se puede estimar un riesgo asociado al ciclo de vida a utilizar:

Sensibilidad a cambios en los requerimientos

> ***Descripción***: con el uso de la entrega por etapas es posible que se presente una sensibilidad muy alta a cambios de prestaciones en las etapas que se realicen primero. El problema es latente si el usuario cambia drásticamente las características del desarrollo afectando detalles o estructuras de diseño sobre etapas previas, obligando a redefinir y modificar los subproductos retrasando cronogramas, incrementando costos, etc.
>
> ***Peor caso***: en el peor de los casos es necesario reconstruir desde la primera etapa, realizando revisión sobre cada una de las siguientes. Es posible que no se reutilice la codificación realizada.
>
> ***Gestión***: como medida tendiente a mitigar el impacto de este riesgo se propone la adición de una etapa 0, en la cual se haga una identificación de requerimientos y prestaciones, que permitan obtener confianza sobre la definición del producto a desarrollar.
>
> ***Excepciones***: no cualquier cambio propuesto por el usuario tendría un impacto negativo sobre el modelo aplicado. En algunos casos, utilizar

la entrega por etapas es favorable, especialmente si los cambios son dados para etapas no desarrolladas.

3.3.3.2. Control

Como mecanismo de monitorización de riesgos se utilizará la información del formato de estimación (punto anterior), evaluando como un criterio cualitativo, cuál fue el impacto del riesgo y cómo se está dando la respuesta. Este control se realizará durante el proceso de codificación, como tareas adicionales a la revisión y prueba.

3.4. Consideraciones de Diseño

3.4.1. Gestión de configuraciones

3.4.1.1. Manejo de configuraciones

Como herramientas para la gestión de configuraciones se propone la utilización de:

- **Estándar de Documentación**: utilización de estándares para la edición de documentación interna y externa al producto.
- **Listas de correo**: Como mecanismo de flujo y almacenamiento, la listas de correo ayudan como instrumento que facilita la comunicación entre el personal involucrado al proyecto.
- **Control de tareas**: Identificación de roles sobre el personal de diseño y desarrollo, asignación y ordenamiento de tareas. Asignación de responsabilidades.

- **Cronogramas**: Manejo de diagramas de Pert y Gant para conocer actividades, tiempos, ruta crítica.
- **RCS** (*Revision Control System*): sistema de control de versiones que automatiza el almacenamiento, recuperación, acceso, identificación y mezcla de versiones. Como herramienta específica CVS (*Code Management System*) proporciona estas prestaciones.

3.4.1.2. Control de errores

Manejo de listado de errores y uso de excepciones que permitan conocer el comportamiento del sistema y solucionar eficientemente los problemas del producto.

3.4.2. Patrones de diseño

Los patrones de diseño de software constituyen un conjunto de principios generales y expresiones que ayudan a desarrollar software. Un conjunto destacado de patrones es GRASP (*General Responsabilty Assignment Software Patterns*) [CL99], que permite establecer algunos parámetros útiles para el diseño del producto.

3.4.2.1. Modularidad

Es la propiedad de un sistema que ha sido descompuesto en un conjunto de módulos coherentes e independientes.

Para favorecer la construcción y uso del sistema se realizará un diseño e implementación modular. Se utilizarán los patrones GRASP experto, Alta cohesión y Bajo acoplamiento [CL99]:

- **Experto**: Asignar una responsabilidad al experto en información: el módulo que cuenta con la información necesaria para cumplir la responsabilidad. Facilidad de entender, mantener y manipular. Reutilización.
- **Alta cohesión**: Asignar una responsabilidad para mantener alta la cohesión. Cuan relacionadas y enfocadas están las responsabilidades de un módulo.
- **Bajo acoplamiento**: Asignar una responsabilidad para mantener bajo el acoplamiento. Tener la mínima conexión entre los módulos.

3.4.2.2. Escalabilidad

El sistema debe contener un núcleo central que realice las operaciones prioritarias. También contendrá módulos de funcionamiento independiente que interactúen con el núcleo principal y con otros módulos. Este esquema permitirá adicionar otros módulos según se presenten nuevos requerimientos o modificar las características de los existentes.

3.4.3. Diseño crítico

De acuerdo al modelo HRT-HOOD [BW94], un objeto HOOD tiene propiedades estáticas (componentes internos funcionales) y dinámicas, las cuales describen los efectos del llamado sobre objetos :

- **Flujo Secuencial**: el control es transferido directamente a la aplicación requerida.
- **Flujo Paralelo**: el control es transferido al objeto llamado pero independientemente de otros flujos.

Aunque el modelo HRT-HOOD está enfocado al diseño de sistemas en tiempo real duro, también es posible su aplicación para sistemas críticos, por lo cual se utilizarán algunos de los conceptos de diseño en este modelo.

3.4.3.1. Diseño lógico de la arquitectura

El diseño lógico puede ser hecho independientemente de las restricciones impuestas por el entorno de ejecución. Hay dos aspectos que facilitan el diseño de la arquitectura lógica de cualquier método de modelado para sistemas de tiempo real duros: Primero, soporte con la abstracción que es requerida por los diseñadores. El segundo aspecto involucra restricciones de arquitectura lógica, así que esto puede ser analizado durante el diseño de la arquitectura física. En particular el diseño es forzado para adaptarse a un modelo computacional que facilite el análisis.

1. **Capa de Interfaz con el usuario**: para el modelo, la interfaz con el usuario, consiste en la visualización y posible interacción del usuario con el sistema. En términos generales tiene una prioridad media-baja en cuanto a uso dentro del sistema crítico, ya que su función corresponde a brindar información gráfica sobre las señales y su comportamiento, mas no es vital para el conjunto del sistema.

2. **Capa de alimentación:** entrada de datos, para el caso del modelo señales e información asociada específica de la aplicación.

3. **Capa de lógica**: corresponde al procesamiento de la señal, específicamente detección, clasificación y respuesta.

4. **Capa de actuador**: sistema de respuesta. Interpreta y ejecuta la respuesta de la capa lógica.

5. **Capa de comunicaciones**: procedimientos y protocolos de comunicaciones entre los módulos.

6. **Capa de registro del sistema**: como mecanismo de supervisión y depuramiento, se incluyen registros (*Logs*), indicando estados de los procesos en ejecución, fallos, advertencias, entre otros.

3.4.3.2. Diseño físico de la arquitectura

Toma los requerimientos no funcionales para crear un diseño funcional y generar un análisis de organización de tareas que garantice su correcto funcionamiento. La actividad de diseño lógico asegura que todo el diseño conforme un modelo computacional que facilite el análisis del tiempo de ejecución. En general el diseño de la arquitectura física concierte cuatro actividades:

1. **Localización de objetos -** Localización de objetos en la arquitectura lógica para procesadores con las restricciones impuestas por los requerimientos funcionales y no funcionales.

2. **Organización de red -** programar la red de comunicaciones para la espera de mensajes.

3. **Organización de procesadores -** determinar la programación (estática o dinámica) con la cual asegurar que todas las tareas dentro de los objetos residan en los procesadores en el momento de ejecución.

4. **Dependencia** - determinar la capacidad de detectar fallos imprevistos de software o hardware y su tolerancia a ellos.

3.4.4. Excepciones de aplicación

Este aparte puede verse como la personalización de los conceptos y métodos desarrollados en el modelo. Para el problema planteado, en términos generales, se podrán utilizar las descripciones y métodos ofrecidos por el modelo. Sin embargo, para algunos casos de aplicación, los conceptos podrían tener otras interpretaciones, adiciones o no aplicación, de acuerdo a la descripción detallada del problema que se tenga.

La inclusión de un mecanismo de redefinición de conceptos, permite por una parte hacer más flexible la aplicación del modelo a otro tipo de problemas; por otra parte, diversos tipos de aplicación permitirían la retroalimentación al modelo para futuras revisiones.

3.5. Definición Plan de Pruebas

3.5.1. Sobre el diseño del producto

3.5.1.1. Definición de las etapas de desarrollo

Cada etapa es en sí misma es un desarrollo parcial del proyecto total, pero es un sub-producto funcional. Al final de cada etapa se desarrollarán pruebas de verificación del software elaborado. La última etapa de desarrollo corresponderá a la

integración de los módulos elaborados anteriormente.

3.5.1.2. Pruebas de integración

Como parte del proceso de diseño global del modelo se debe realizar la especificación de los protocolos de comunicación o paso de mensajes y los formatos internos para los datos. Cualquier modificación estará sujeta a aprobación por el diseño detallado de cada etapa de desarrollo.

3.5.1.3. Pruebas de sistema crítico

El componente de sistema crítico corresponde a uno o varios módulos del sistema. Debido al nivel de estabilidad que requiere este componente es recomendable que se haga la definición de un plan detallado adaptado a las necesidades del sistema.

3.5.1.4. Pruebas de rendimiento

Como parte de la comprobación del producto, se realizarán pruebas de consumo de recursos computacionales. Para facilitar este proceso se recomienda implementar mecanismos que permitan conocer los estados internos de los módulos mediante sistemas de registros del sistema y codificación de mensajes de error.

3.5.2. Sobre la funcionalidad del producto

La definición de los casos de prueba se realizará por cada etapa. Estas pruebas de funcionalidad consisten en verificar externamente la ejecución del producto, que para el caso del ciclo de vida a utilizar constituye probar los sub-productos y luego el producto final.

El esquema general de las plantillas puede ser dado como:

- **Caso de prueba**: Nombre del caso de prueba
- **Objetivo de la prueba**: Descripción del cometido de esta prueba
- **Configuración hardware y software**: requerimientos y configuración del sistema utilizado.
- **Formatos de entrada**: Formatos o protocolos de la información de entrada.
- **Formatos de salida**: Formatos o protocolos de la información resultante.
- **Procedimiento de prueba**: Descripción del procedimiento utilizado por el encargado de la prueba.
- **Resultados**: Comentarios de ejecución de la prueba y respuesta del sistema.

3.6. Resumen del modelo

En general el modelo presenta y contempla los siguientes conceptos:

Planeación: establecer los recursos y medios para la realización del proyecto.

- Análisis de factibilidad

- Actividades generales
- Sistema de Calidad

Análisis Global: comprensión de los esquemas de funcionamiento del tipo de sistemas objetivo.

- Descripción del sistema
- Análisis de requerimientos

Selección del Ciclo de Vida: determinación del modelo de desarrollo del producto de software a seguir.

- Entrega por etapas
- Estimación y Control del riesgo

Diseño Global: Consideraciones para el diseño de las aplicaciones y concepto de sistema crítico.

- Gestión de configuraciones
- Utilización de Patrones
- Diseño Crítico

Plan de pruebas: Como mecanismo de evaluación continua, se establecen procedimientos para la comprobación de la herramienta.

- Sobre el diseño o pruebas de caja blanca
- Sobre la funcionalidad o pruebas de caja negra