UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ZARAGOZA

IMPORTANCIA DE LA CALIFICACIÓN DEL DISEÑO EN LA VALIDACIÓN DE SOFTWARE

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:
QUÍMICO FARMACÉUTICO BIÓLOGO

P R E S E N T A :
RICARDO GABRIEL LONA GARCÍA

ASESOR:
Q.F.B. JOSÉ LUIS BALDERAS LÓPEZ

MÉXICO, D.F. 2004

Dedico este trabajo

A Angélica, por su apoyo y amor incondicionales, que han sido pilares en el logro de esta y todas las metas que hemos alcanzado en nuestro camino juntos.

A Riquis y Gabriel, que son fuente inagotable de inspiración y fortaleza para seguir luchando día a día por ser mejor.

A mi Papá, que siempre está conmigo.

A mi Mamá, por ser la mejor amiga y consejera, y por representar el más claro ejemplo de fortaleza, de lealtad, y de que el amor perdura por siempre.

A mis hermanos, Mingo, Tomás, Víctor y Daniel, por ser la guía en mi camino y por darme los mejores legados que son mi carrera y su ejemplo de responsabilidad y calidad humana.

A Lupe y Enrique, por el cariño demostrado con desvelos, que significaron logros en el camino hacia esta meta.

INDICE

1.0 Introducción

A lo largo de la historia de la industria de medicamentos, desde su elaboración en pequeños "talleres" o boticas hasta la actualidad, se han ido gestando junto con los descubrimientos e innovaciones tecnológicas y científicas, las regulaciones sanitarias que rigen actualmente, siempre con la finalidad de obtener y brindar al paciente productos de calidad, seguros y eficientes.

Actualmente y de forma más específica, las Prácticas Adecuadas de Manufactura (GMP´s, PAM´s, BPM´s), la Prácticas Adecuadas de Laboratorio (GLP´s, PAL´s, BPL´s) y las Prácticas Clínicas Adecuadas (GCP´s, PCA´s, BPC´s) se han emitido en respuesta a problemas que de una forma u otra amenazaron la seguridad de los productos farmacéuticos.

La validación de los sistemas de cómputo, surge de la implementación de las GMP´s y GLP´s.

Debido a que gran parte de estos lineamientos se enfocan al manejo de registros, la industria creó sistemas de manejo de datos y reporte de operaciones, por lo que la FDA inició con inspecciones a estos sistemas considerando los puntos de desarrollo, mantenimiento y documentación. Como resultado de estas inspecciones, surgió en 1983 el llamado "Libro azul".

Las entidades regulatorias están preocupadas de que los sistemas utilizados en las distintas etapas del desarrollo de un medicamento, ensayos clínicos, manufactura, análisis y distribución estén adecuadamente desarrollados, utilizados y cumplan con un mantenimiento adecuado para asegurar la integridad de los datos y productos. Las regulaciones incluyen GLP´s y GMP's. La FDA considera las GLP´s en la parte 58 del

CFR Title 21 (Code of Federal Regulations). " El cumplimiento con las GLP´s está destinado a asegurar la calidad y la integridad de los datos seguros" (1). La s GMP´s están consideradas en el CFR parte 211. "Las regulaciones en esta parte contienen lo mínimo para buenas prácticas de manufactura para la preparación de medicamentos para administración en humanos y animales". (2)

Durante la década pasada existió una proliferación de sistemas de cómputo en la industria farmacéutica que manejan una enorme cantidad de datos críticos. Y es tarea de la validación de sistemas de cómputo evaluar la integridad, la exactitud y reproducibilidad para asegurar la confiabilidad de los datos, aún cuando el sistema utilizado haya sido desarrollado de acuerdo a estándares aceptables, sea utilizado correctamente, por personal entrenado y cumpla con su mantenimiento.

De manera recíproca el diseño de un producto se convierte en un proceso de aprendizaje cooperativo. El proceso se presenta como un modelo en cascada en donde se identifican ciclos de aprendizaje a lo largo del desarrollo del software, tal y como se observa en la figura 1. (3)

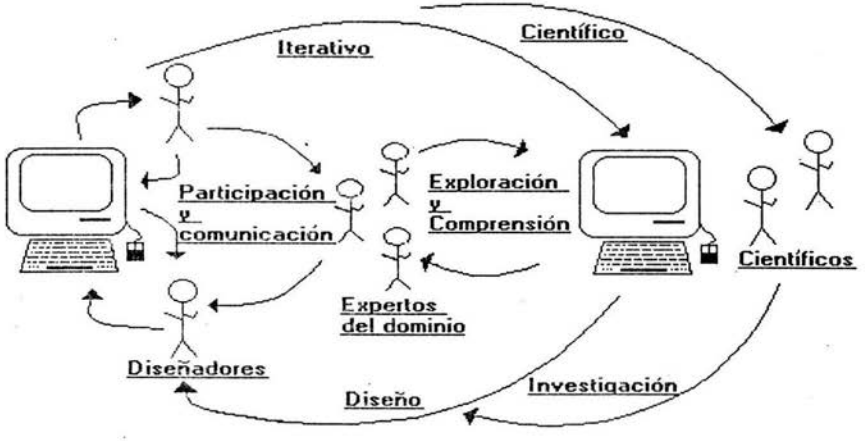


Figura 1: Ciclos de aprendizaje en el desarrollo del software

2.0 Planteamiento del Problema

El desarrollo tecnológico de las industrias, provocado por una necesidad de competitividad comercial en los mercados globalizados, marca la pauta de las estrategias de negocio, enfocadas hacia la obtención de productos de manera eficiente y rápida, sin descuidar los preceptos de calidad establecidos internamente desde el diseño del producto.

Es este mismo empuje hacia la rapidez, lo que ha generado la necesidad de manejo de información, comunicación y procesos a través de sistemas de computo.

En la industria de insumos para la salud, los sistemas de computo, pueden en su mayoría estar relacionados con actividades de impacto para la calidad del producto, y por esto existen instancias gubernamentales que proponen la manera en que estos sistemas deban ser verificados y validados (FDA principalmente).

Tales sistemas relacionados con las actividades de la empresa deben estar diseñados con la perspectiva de cumplir con las reglamentaciones y la satisfacción de los requerimientos del cliente (usuario del sistema), sin perder el enfoque de calidad del producto.

La validación de sistemas de cómputo representa un tema novedoso dentro de la verificación regulatoria de la industria farmacéutica. En nuestro país, aún no contamos con alguna norma oficial que considere este tópico. Sin embargo, existen guías internacionales que marcan la pauta y los puntos y criterios esenciales para probar tales sistemas desde su diseño hasta su mantenimiento en el sitio de uso.

La etapa crítica considerada dentro del ciclo de vida de un software es la de diseño, ya que en ella serán desarrollados los requerimientos del sistema, involucrando

las necesidades de ingeniería, regulatorias y de cumplimiento de los requerimientos del usuario.

Es esta complejidad de puntos de vista, lo que hace necesario el estudio minucioso de la relevancia de esta etapa a través del impacto en las otras etapas del desarrollo del ciclo de vida del software.

En el presente trabajo se establece la importancia de la calificación del diseño de software, destacando la elaboración y evaluación de los requerimientos en la creación, desarrollo e implementación de un sistema y su relación con las fases de la validación prospectiva de software como una herramienta indispensable en la detección y solución de problemas en el ciclo de vida de desarrollo del mismo.

3.0 OBJETIVOS

3.1 Objetivo General

Determinar la importancia de la calificación del diseño en una validación prospectiva de software a través de la identificación de los requerimientos principales y la evaluación de su impacto en la verificación y validación del sistema de cómputo.

3.2 Objetivo Particular

Establecer una guía que permita identificar los requerimientos regulatorios y expectativas del usuario en el diseño y aprobación de software.

4.0 Metodología

La metodología de análisis se dividió en tres etapas principales involucrando el uso de fuentes de información Primarias (libros, artículos, documentos oficiales y referencias de red)

Primera etapa.

Se establecieron los requerimientos principales de validación y diseño de software desde el punto de vista regulatorios y de ingeniería.

Método: Revisión y análisis de documentos de ingeniería de software y guías internacionales de la FDA.

Segunda etapa.

Se revisó la relación de los requerimientos con las etapas consideradas en la validación de software y el ciclo de vida del software.

Método: Revisión y análisis de las etapas de validación y ciclo de vida del software encontradas en las guías y aquellos documentos relacionados, contra los requerimientos detectados en la etapa anterior.

Tercera etapa.

Se elaboró una guía para detección de requerimientos de software de acuerdo a los puntos críticos detectados en la segunda etapa.

Método: Análisis de los resultados y elaboración de una lista de verificación de requerimientos de software.

5.0 VALIDACION DE SISTEMAS DE COMPUTO

Validación de sistemas de cómputo: "Es un proceso que documenta que un sistema de cómputo lleva a cabo de manera reproducible la función para la que fué diseñado." (4)

"Confirmación por examinación y provisión de evidencia objetiva, de que las especificaciones del software cumplen con las necesidades y usos programados por el usuario y que los requerimientos particulares implementados con el uso del software **pueden ser cumplidos consistentemente" (5)**

Validación Prospectiva: "El proceso de validación se realiza mientras el sistema es desarrollado e instalado. Regularmente el proceso se completa antes de que el sistema haya sido aprobado y se encuentre en uso. La documentación necesaria para la validación es generada durante el desarrollo del sistema manteniendo en mente los requerimientos para la validación."(6)

Validación Retrospectiva: "Este tipo de validación aplica a los sistemas que han sido utilizados por muchos años y que cuentan con gran cantidad de evidencia de que funcionan adecuadamente. Tal evidencia deberá ser de naturaleza Tangible (ej. Reportes impresos, resultados de pruebas, cartas de control, registros de control de calidad, etc.). La validación entonces, consistirá en recopilar y ensamblar toda la información en un protocolo de validación y completar la información faltante."(6)

Para llegar a la conclusión de que un software ha sido validado se requiere de tiempo y una serie de esfuerzos y tareas. La preparación para la validación debe comenzar desde las etapas tempranas, por ejemplo, durante la planeación del diseño y

desarrollo. La conclusión final de que un software está validado, deberá estar sustentado en evidencia colectada de las tareas planeadas a lo largo del ciclo de vida del software.

5.1 El ciclo de vida del software

"Es una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software (IEEE 1074)" (7)

"Es un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso (ISO 12207-1)" (7)

El ciclo de vida del software se compone de una serie de tareas de ingeniería y documentación, necesarias para soportar la validación, además de tareas específicas de verificación y validación acordes al uso propuesto del software. (5)

Procesos principales del ciclo de vida del software (4,5,7):

- Adquisición
- Suministro
- Desarrollo
- Explotación
- Mantenimiento

Procesos de soporte

- Documentación
- Gestión de configuración
- Aseguramiento de calidad
- Verificación

- Validación
- Auditoría
- Revisión conjunta
- Resolución de problemas

Procesos de la organización

- Gestión
- Infraestructura
- Mejora
- Formación

5.1.1 Adquisición y Suministro

Representan el inicio y final de la negociación administrativa de las condiciones en que se proveerá el software y representa sólo la negociación contractual del proceso iterativo del desarrollo del software. Comprende una parte esencial de los procesos de la organización (gestión, infraestructura, mejora y formación).

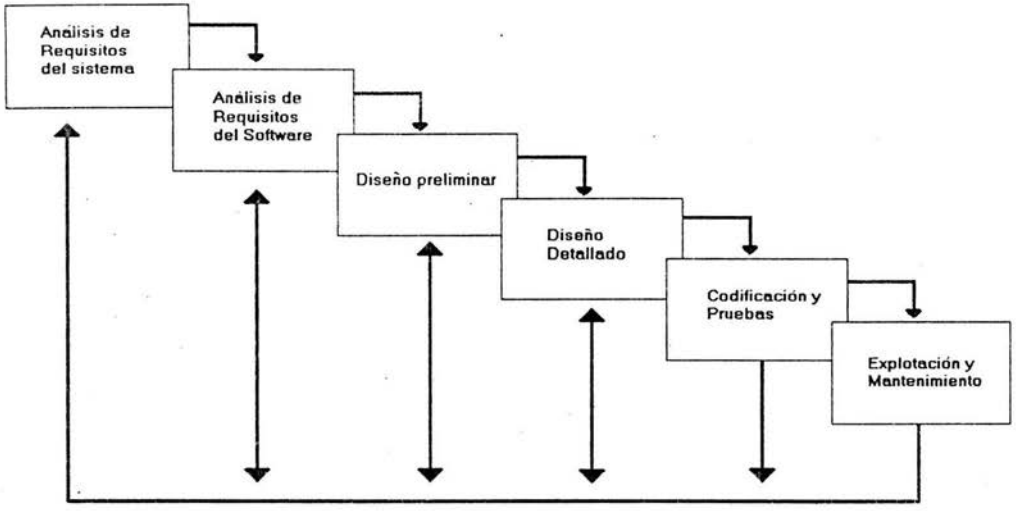
5.1.2 Desarrollo

La etapa de desarrollo, comprende la construcción del sistema desde una idea abstracta, hasta la aceptación final en el punto de uso.

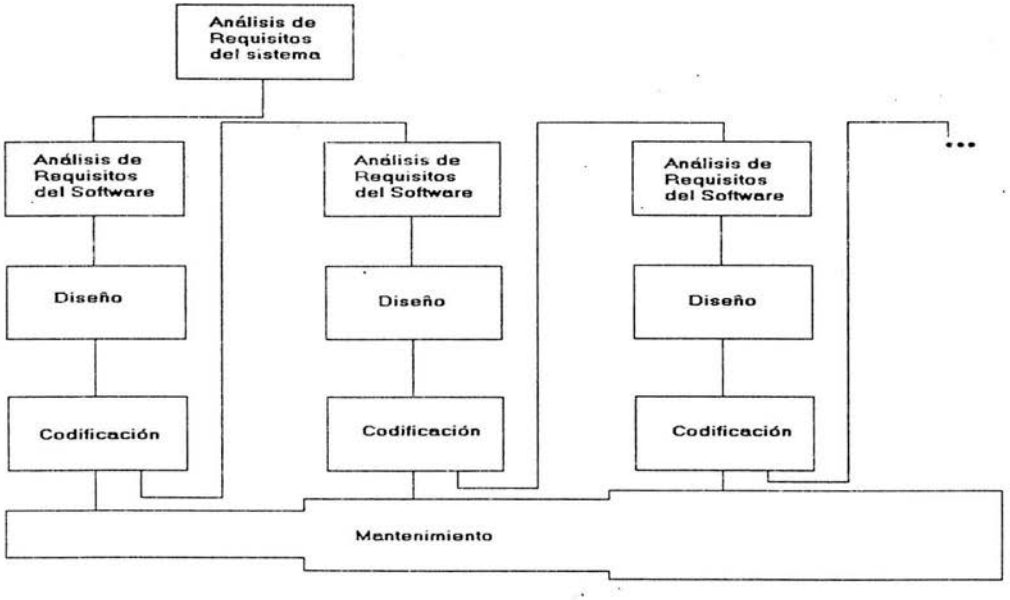
Existen diferentes modelos para el desarrollo de software, que son considerados dependiendo de la complejidad del sistema a desarrollar.

Para los fines del presente trabajo sólo se presentarán los modelos sin ahondar en las características o métodos de implementación particulares de cada uno (7).

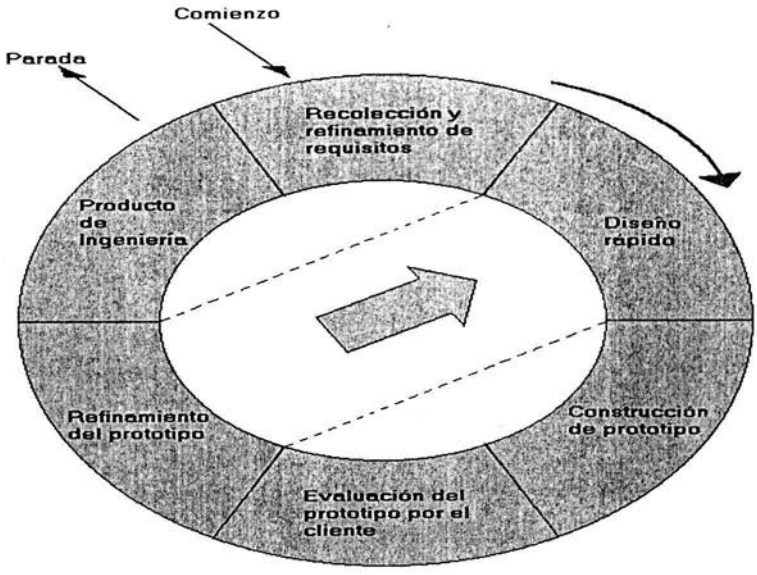
Modelo en cascada



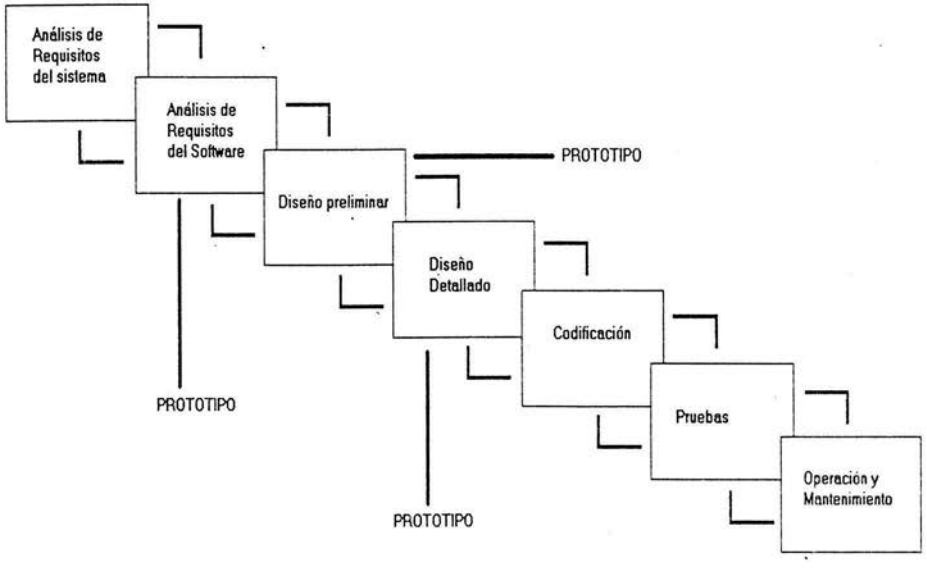
Modelo Incremental



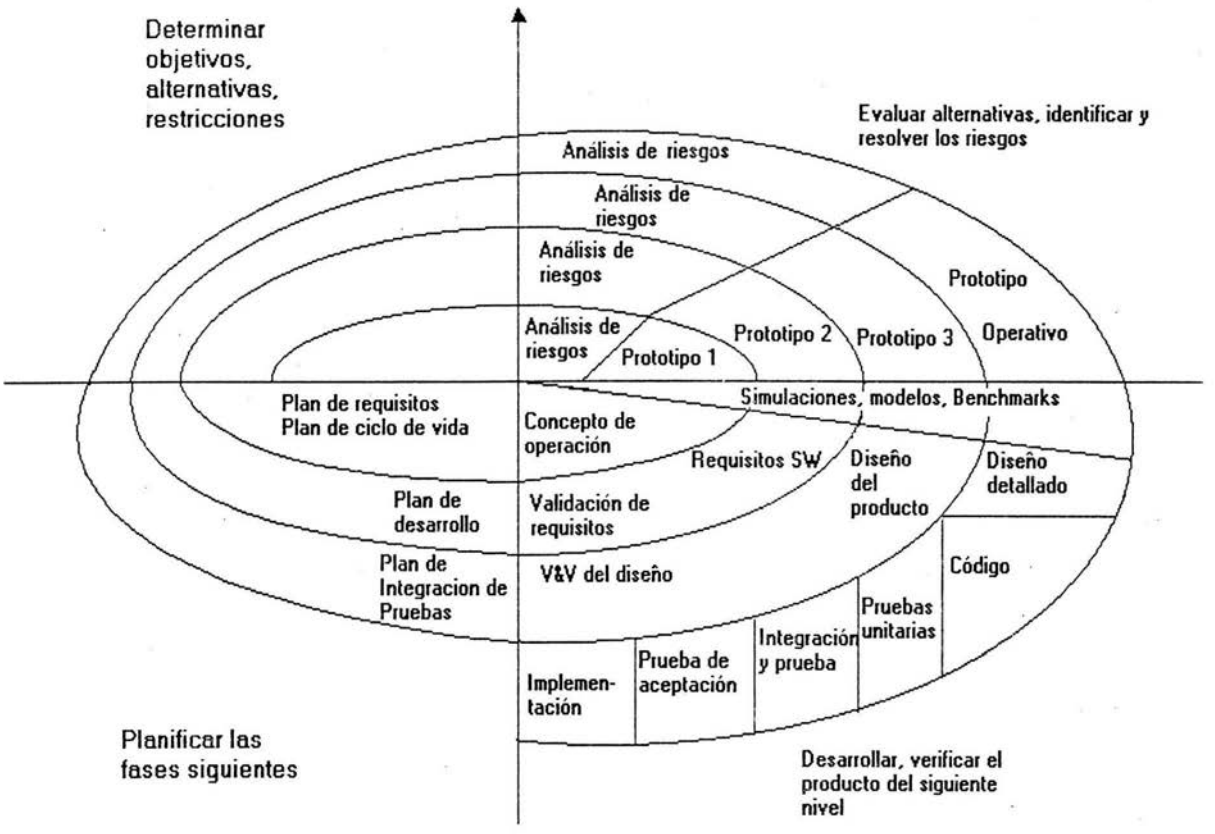
Modelo de Prototipo



Modelo de Prototipado rápido



Modelo en Espiral



La aplicación de los modelos de desarrollo de sistemas estará en función de las necesidades y recursos con los que se cuente. Sin embargo, los diseñadores establecerán el modelo de desarrollo acorde a su producto y organización.

El modelo seleccionado (cualquiera que sea), deberá cubrir el ciclo de vida del software desde su nacimiento hasta su retiro.

Las actividades típicas en cualquier modelo son: (4, 5)

5.1.3 Planificación de la Calidad.

En esta etapa se generará un plan que identifique las tareas necesarias, procedimientos para reporte y resolución de anomalías, los recursos necesarios y la revisión periódica de los requerimientos, incluyendo revisiones formales de la etapa del diseño.

El plan deberá incluir:

- Las tareas específicas para cada actividad del ciclo de vida.
- Enumeración de factores importantes de calidad (ej. confiabilidad, mantenimiento, utilidad).
- Métodos y procedimientos para cada tarea.
- Criterios de aceptación para cada tarea.
- Criterios para la definición y documentación de los resultados en términos que permitan la evaluación de su conformidad con los requerimientos de entrada.
- Entradas para cada tarea
- Resultados de cada tarea
- Roles, recursos, y responsabilidades para cada tarea
- Riesgos y suposiciones y
- Documentación de las necesidades del usuario

5.1.4 Requerimientos

Los requerimientos definen que va a hacer el sistema. Los requerimientos para un software deberán estar detallados y reflejar como se usará el sistema y donde estará localizado (4).

La especificación documentada de los requerimientos del software, provee el punto de partida tanto para la validación, como para la verificación (5).

Un requerimiento puede ser cualquier necesidad o expectación por un sistema o por su software. Los requerimientos reflejan las necesidades establecidas o implícitas del cliente.

Pueden existir diferentes tipos de requerimientos, por ejemplo, de diseño, funcionales, de implementación, de interfase, de desempeño o físicos. Los requerimientos del software se derivan de los aspectos considerados para la funcionalidad del sistema, son definidos, afinados y actualizados a través de un proyecto de progreso de desarrollo.

El desarrollo de los requerimientos incluye la identificación, análisis, y documentación de la información acerca del dispositivo y su uso propuesto. Las áreas de especial importancia que se deben considerar incluyen, distribución de las funciones del sistema, condiciones de operación, características del usuario, riesgos potenciales, y tareas anticipadas. Adicionalmente, deben establecer claramente el uso propuesto del software.

Los requerimientos del software deberán especificar lo siguiente:

- Todas las entradas del software.
- Todas las salidas del software.
- Todas las funciones que el software del sistema desarrollará.
- Todos los requerimientos de las funciones que el software deberá cumplir (ej. el manejo de los datos, la confiabilidad, el tiempo de respuesta).
- La definición de las interfases externas y de los usuarios, así como cualquier interfase interna de software – sistema.

- Cómo los usuarios deberán interactuar con el sistema.
- Cómo se constituyen los errores y cómo se deberán manejar.
- Tiempos de respuesta.
- El entorno de operación proyectado para el software, cuando se trate de un a restricción de diseño (ej. plataforma del hardware, sistema operativo).
- Rangos, límites, valores predeterminados y valores específicos que el software deberá aceptar y,
- Los requerimientos relacionados con la seguridad, especificaciones, características, especificaciones o funciones que serán implementadas en el software y que se derivan de un proceso de administración de riesgos que deberá estar estrechamente ligado al proceso de desarrollo de los requerimientos del sistema.

5.1.5 Diseño

Software: "Programas, procedimientos, reglas, y cualquier documentación asociada a la operación de un sistema." (8)

Hardware: "Equipo físico, lo opuesto a los programas, procedimientos, reglas y documentos asociados. Contraste del software." (8)

Diseño: "Es el proceso de aplicar distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización física." (9)

El desarrollo de software ha tenido cambios sustanciales en la forma de concebir los proyectos de diseño y construcción de programas para microcomputadoras.

En un principio, el desarrollo se ubicaba en el nivel de una simple tarea de programación, considerándola más como un arte que una ciencia (y no precisamente como una disciplina de ingeniería).

A lo largo de los años, se detectaron tres razones principales, que llevaron al diseño de software al nivel de una ciencia de alta aplicación industrial:

1) El primer punto, corresponde a promover un acercamiento entre los científicos de la computación y los ingenieros encargados, ya que a pesar de la alta capacitación de los diseñadores, aunado a la designación de tareas triviales en la aplicación de software, no existía un enfoque de aplicación neta de los programas.
2) En segundo lugar y con el avance de los sistemas automatizados en las industrias, se busca establecer métodos para el diseño más adecuados y apropiados para aplicaciones reales.
3) En tercer lugar, se trata de asimilar las dificultades de los diseñadores de los microprocesadores por parte de los científicos tradicionales de los sistemas de cómputo. (10)

El diseño es el primer paso de la fase de desarrollo de cualquier sistema o producto de ingeniería. El objetivo del diseñador es producir un modelo o representación de una entidad que construirá más adelante.

El diseño es el proceso en el que se determina la calidad del desarrollo del software.

El diseño del software es un proceso iterativo a través del cual se traducen los requisitos en una representación del software. El diseño se representa a un alto nivel de abstracción, un nivel que se puede seguir hasta requisitos específicos de datos,

funcionales y de comportamiento, reduciendo su nivel de abstracción en las iteraciones posteriores (11).

En esta etapa, el personal encargado del desarrollo del sistema traduce los requerimientos funcionales en un diseño. La fase de diseño define como serán implementadas las funciones listadas en las especificaciones de los requerimientos. Esta etapa puede estar dividida en dos partes (diseño general y diseño a detalle) o puede tener elementos de ambas en un sólo documento. El documento de diseño contiene temas como descripción del sistema, diseño de pantalla, características del reporte, descripción de datos, configuración del sistema, estructura de archivos, y diseño de módulos. (4)

En el proceso de diseño, los requerimientos se traducen en una representación física del software a ser implementado. La especificación de diseño es una descripción de qué va a hacer el software y cómo lo deberá realizar.

Las especificaciones de diseño deberán contener tanto información detallada, como información sumarizada del proceso de diseño.

Tales especificaciones restringen al programador a permanecer dentro de los propósitos acordados en la especificación de los requerimientos, relevándolo en la necesidad de tomar decisiones *ad hoc* al diseño.

Las especificaciones de diseño deberán incluir:

- Especificaciones de los Requerimientos del software, incluyendo criterios predeterminados para la aceptación del software.
- Análisis de riesgos del software.
- Procedimientos de desarrollo y guías de codificado (u otros procedimientos de programación).

- Documentación de los sistemas que describa el contexto en el cual el sistema se supone será utilizado, incluyendo la relación de hardware, software y el ambiente físico.
- Hardware a ser utilizado.
- Parámetros a ser medidos o registrados.
- Estructura lógica (incluido el control lógico) y etapas de procesamiento lógico (ej. algoritmos).
- Estructura de datos y diagrama de flujo de datos.
- Mensajes de error, alarma y advertencias.
- Software de soporte (ej. sistemas operativos, drivers, otros softwares de aplicación).
- Vínculos de comunicación (vínculos entre módulos internos del software, con el software de soporte, con el hardware y con el usuario).
- Medidas de seguridad (tanto físicas como lógicas).
- Cualquier restricción adicional no considerada en las anteriores.

5.1.6 Prueba

Es el procedimiento de reto para el sistema en diferentes niveles del desarrollo, basado en un plan de prueba predefinido y resultados conocidos. La fase de prueba se lleva a cabo a todos los niveles del desarrollo del software. A través de esta fase deben llevarse a cabo revisiones apropiadas de los resultados obtenidos.

Durante esta etapa, se han realizado cuatro niveles que son:

5.1.7 Implementación

Durante esta etapa, el sistema es construido mediante la adquisición y prueba del hardware necesario y el código de escritura y prueba (ej. Software) conforme a las especificaciones del diseño. Mientras cada modulo o subsistema es codificado, el encargado del desarrollo lleva a cabo las pruebas necesarias para demostrar que se cumplen las funciones deseadas y cualquier error de codificación encontrado durante la prueba de módulo es corregido.

5.1.7.1 Integración

En la integración, todos los módulos del software son enlazados en el sistema y probados en conjunto.

5.1.7.2 Instalación

Este es el proceso de instalación del Hardware (si lo requiere) y el software en el sitio de uso y la prueba de su funcionalidad.

5.1.7.3 Aceptación

Este es el proceso de prueba del sistema para determinar si se han cumplido todas las funciones para identificar problemas con el sistema, establecidas en las especificaciones funcionales de los requerimientos. Una aceptación exitosa no solo probará que el sistema trabaja, también encontrará aquellas partes del programa que no funcionan correctamente. En este punto, el usuario puede aceptar el sistema con o sin restricciones o rechazarlo.

Cuando el sistema ha sido aceptado y se encuentra dentro del ambiente operacional, el usuario puede obtener datos de confiabilidad adicionales del sistema,

realizando una prueba paralela. Esta prueba puede ser una comparación de los resultados obtenidos de forma manual o con el sistema anterior y los resultados con el sistema nuevo (4).

El asegurar la calidad del software debe enfocarse en prevenir la introducción de defectos durante el desarrollo del software y no cuando el código ha sido escrito.

Las pruebas del software son una actividad necesaria, sin embargo, en la mayoría de los casos, la prueba de software por sí misma no es suficiente para establecer la confiabilidad de que el software es apropiado para el uso propuesto.

Con la finalidad de establecer tal confiabilidad, los diseñadores deben utilizar una mezcla de métodos y técnicas para prevenir y detectar errores en software. La "mejor mezcla" de métodos depende de muchos factores, incluyendo el entorno del desarrollo, la aplicación, tamaño del proyecto, lenguaje y riesgo (5).

5.1.7.4 Operación

Una vez que el sistema se ha aceptado y opera en línea, comienza la fase operacional y asegura que el sistema se encuentra operando como se diseñó a través de evaluaciones de rutina y acorde a Procedimientos Estándar (Normalizados) de Operación.

5.1.7.4.1 Seguridad

Las medidas de seguridad son vitales para todo sistema debido a que la información o datos son posesiones invaluables y necesarias tanto para las instancias legales y regulatorias. La seguridad puede ser clasificada como física o dependiente del software. La medida física incluye cerraduras y límites de acceso al equipo. La seguridad del software incluye palabras clave (password), dispositivos de reconocimiento (ej. Código de barras), y acceso limitado a funciones del sistema de acuerdo a las

responsabilidades laborales o necesidades de información (niveles). Otro elemento de seguiridad que debe ser considerado es el software de protección antivirus.

5.1.7.4.2 Control de Cambios

Para asegurar que un sistema permanece en un estado validado, se deberá establecer un procedimiento que indique que todos los cambios al sistema hardware y software están documentados, probados y aceptados.

5.1.7.4.3 Mantenimiento

Deberá existir un procedimiento en el que se documente cualquier mantenimiento de rutina o emergencia realizado al hardware del sistema. Deberá incluirse en el documento el establecimiento del problema junto con la acción correctiva y seguimiento.

5.1.7.4.4 Retención de Documentos

Se deberá desarrollar un procedimiento que defina qué documentos serán retenidos, donde serán resguardados y cuánto tiempo. Estos detalles serán determinados, de acuerdo al ambito regulatorio en que se desarrolle la operación.

5.1.7.4.5 Revisión Periódica

Esta etapa consiste de una revisión en dos partes de los resultados y desempeño del sistema. De manera rutinaria, los resultados deben ser evaluados por un usuario capacitado en cuanto a si son coherentes, completos y a su calidad de impresión. Periódicamente, el desempeño global del sistema debe evaluarse, identificar tendencias y realizar planes para mantener al sistema operando de manera optima. El propósito de la revision del sistema es establecer y mantener la confianza en el desempeño del mismo (4).

6.0 ACTIVIDADES REQUERIDAS PARA LA VALIDACIÓN DE SOFTWARE (4, 12,13)

La validación de software es un requerimiento del Quality System Regulation, el cual fue publicado en Octubre de 1996, y se aprobó en Junio de 1997. Los requerimientos de validación aplican para software utilizados como componentes en dispositivos médicos, software que por sí mismo representa un dispositivo médico y software utilizado en la producción de los dispositivos o en la implementación del sistema de calidad del fabricante.

Cualquier software utilizado como dispositivo médico (a menos que no se encuentre considerado en la clasificación regulatoria), que haya sido desarrollado después de Junio de 1997, independientemente del tipo de dispositivo que se trate, se someterá a evaluación de acuerdo a las condiciones de control de diseño aplicables.

Estos requerimientos incluyen:

- La conclusión de proyectos de desarrollo en proceso.
- Proyectos nuevos.
- Cambios realizados a los paquetes existentes.

El título 21 del Code of Federal Ragulations (CFR) parte 820, contiene los requerimientos específicos para la validación de software, otros controles de diseño como planeación, entrada, verificación, y revisión (21 CFR 820.30). También se consideran en el 21 CFR 820.70 los puntos de validación que aplican para los sistemas automatizados utilizados en el diseño, prueba, aprobación de componentes, manufactura, etiquetado, empaque, distribución, manejo de quejas o cualquier otra etapa automatizada del sistema de calidad.

Adicionalmente (21 CFR 11.10), los sistemas de cómputo utilizados para crear, modificar y mantener los registros electrónicos y administrar las firmas electrónicas. Tales sistemas de cómputo deberán estar validados para asegurar la exactitud, confiabilidad, desempeño proyectado consistente, y la habilidad de diferenciar o invalidar registros alterados.

El diseñador del software define el ciclo de vida y la validación típicamente se soporta mediante:

- Verificaciones de los datos de salida en cada fase del ciclo de vida de desarrollo del software
- Verificación de la operación propia del software final en el ambiente para el uso propuesto (5)

La validación es aplicada a diferentes aspectos de la manufactura farmacéutica y en cada caso el objetivo principal es producir evidencia documentada, la cual debe proveer un alto grado de confiabilidad de que la operación de un equipo, instalación, proceso o sistema trabaja de manera consistente bajo una condición determinada.

Con una progresión lógica, se desarrollan las actividades generales de validación de la siguiente manera:

ETAPA	DOCUMENTO
Planeación	Preparación de un plan de Validación escrito
Especificación	Especificaciones y acuerdos acerca de lo que se requiere
Planeación de Pruebas	Protocolo de Pruebas para validar el sistema (CI, CO, CF)
Pruebas del sistema	Recolección de datos y resultados (CI, CO, CF)
Revisión	Revisión de resultados para demostrar que el sistema se desempeña, tal y como se especificó.

6.1 Calificación de Instalación (CI)

Es la verificación documental de que los aspectos clave de la instalación de software y hardware, los códigos y las propuestas de diseño aprobados, y que se han considerado de manera apropiada las consideraciones del fabricante.

En la práctica, se trata de asegurar que el sistema ha sido instalado de acuerdo a lo especificado, y que existe suficiente evidencia documentada para demostrarlo.

6.2 Calificación de Operación (CO)

Es la verificación documental de que el equipo o sistema opera tal como se propuso, dentro de rangos representativos o previamente establecidos.

En la práctica, consiste en asegurar que el sistema instalado trabaja tal y como se especificó y se puede demostrar con suficiente evidencia documental.

6.3 Calificación de Funcionamiento (CF)

Es la verificación documental de que el proceso y/o sistema relacionado al proceso, funciona dentro de los rangos propuestos anticipadamente.

En la práctica, consiste en demostrar con la evidencia documental suficiente, que el sistema en sus condiciones normales de operación, proporciona productos de calidad aceptable.

Tradicionalmente, las calificaciones descritas contienen tanto información de las especificaciones, como de las instrucciones de prueba. Estas pruebas, conforman el Protocolo de Validación.

Los documentos de calificación necesariamente deberán referenciarse a las especificaciones originales, de esta manera, se establece un esquema de validación ligado al ciclo de desarrollo del software (Figura 2).

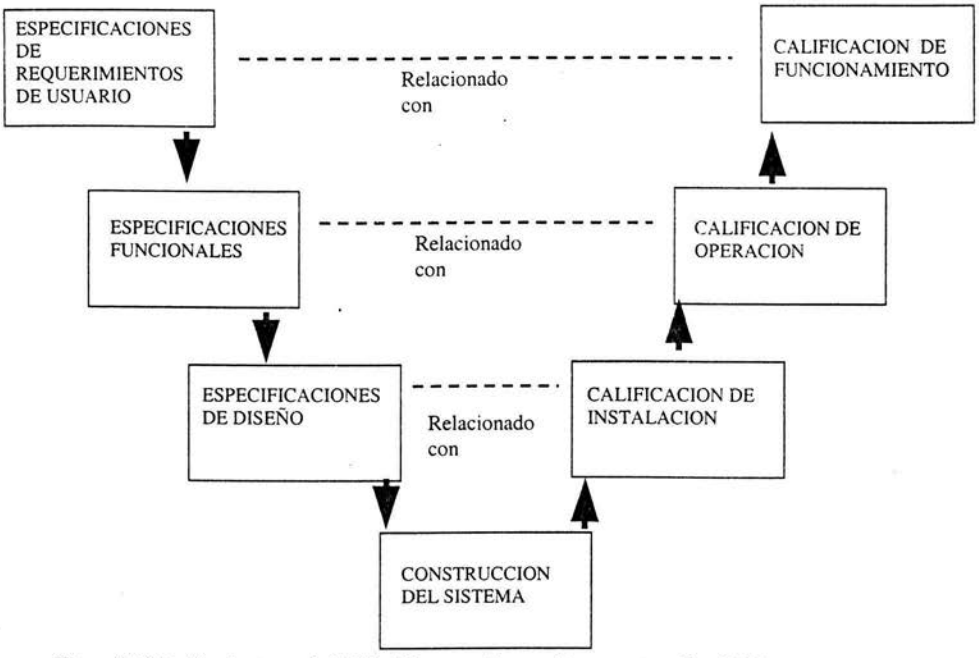


Figura 2. Relación de etapas de Validación con el desarrollo y construcción del sistema

6.4 Especificaciones de los Requerimientos de usuario (ERU)

Normalmente son descritos por el usuario y definen lo que deberá hacer el sistema o equipo. Se deberá realizar una versión inicial, la cual será proporcionada a los proveedores potenciales y que deberá incluir todos los requerimientos esenciales (básicos) y de ser posible, una lista priorizada de requerimientos adicionales (deseables).

Las ERU deberán contar con una revisión final después de seleccionar el proveedor, formará parte de las bases contractuales y deberá estar ligado a la Calificación de Funcionamiento.

6.5 Especificaciones Funcionales (EF)

Normalmente son descritas por el proveedor y describen las funciones detalladas del equipo o sistema. Es un diseño de las especificaciones de los resultados de salida del sistema o funcionamiento del equipo. Están incluidas en la respuesta inicial del proveedor seleccionado y serán actualizadas posteriormente en conjunto con el usuario.

Las EF están vinculadas con la Calificación de funcionamiento para probar las funciones especificadas.

Las EF pueden incluir especificaciones mecánicas, eléctricas, además de las correspondientes al software, cuando se trate de sistemas integrados a equipo de manufactura.

6.6 Especificación de diseño

Es la definición completa del equipo o sistema con el suficiente detalle para que sea construido. Al igual que las EF es un diseño de salida que está ligado a la calificación de Instalación, la cual verifica que el sistema sea provisto e instalado de manera correcta.

Las actividades de Validación que deben realizarse durante el desarrollo de un sistema automatizado, y que se relacionan con el ciclo de vida del software se pueden establecer en un solo diagrama de flujo para la planificación y pruebas del sistema en construcción y que se pueden observar en la figura 3.

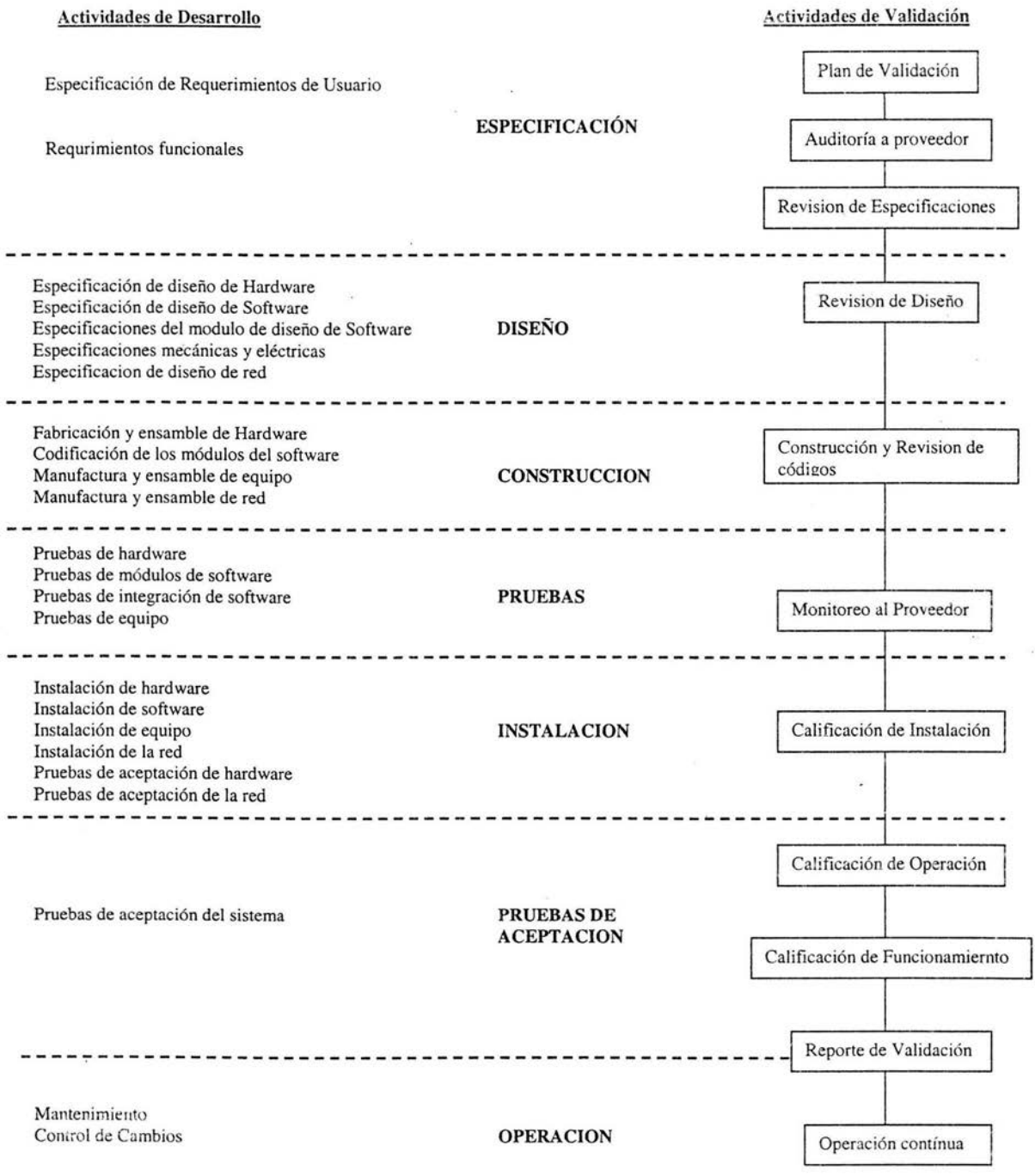


Figura 3. Aplicación del concepto de validación al ciclo de vida del desarrollo de un sistema automatizado

Las actividades correspondientes a la construcción, pruebas e instalación de equipos corresponde a sistemas integrados a equipos de manufactura que cuenten con instrumentos de monitoreo y control.

El monitoreo al proveedor corresponde a las pruebas realizadas por él mismo y que deberán estar consideradas en la CI/CO y deberán evidenciar que fueron correctamente controladas y probadas. Estas pruebas ayudarán a reducir la cantidad de evaluaciones necesarias en la etapa de CO del software.

Como proceso iterativo, la etapa de desarrollo debe establecer responsabilidades compartidas entre proveedor y usuario, que ayudarán a llevar a buen término la creación e implementación de un sistema dentro de una planta:

Para el usuario/proveedor quedan establecidas las tareas de:

- Identificar cuales sistemas son críticos de manera que tengan impacto sobre las BPM's
- Establecer un plan de validación que identifique las actividades a realizar.
- Establecer las ERU de manera que se definan de manera clara y precisa lo que el usuario desea que el sistema haga, además del cumplimiento regulatorio.
- Auditar y seleccionar a los proveedores potenciales, dándoles a conocer los requerimientos del plan de validación y las ERU.
- Revisar y aprobar especificaciones producidas por el proveedor acerca del hardware y funcionalidad del sistema.
- Monitorear el desarrollo del sistema manteniendo control total sobre el progreso a través de la revisión contra lo planeado.
- Revisar y aprobar las especificaciones de las pruebas

- Revisar y aprobar los reportes de pruebas, así como los reportes adicionales.
- Generar el reporte de validación donde se encuentran recopilados todos los documentos generados durante la revisión, así como los entregables definidos en el plan de validación y todo aquello que de evidencia de que el sistema ha sido validado.
- Mantener el sistema, asegurando que se cumpla con un sistema de control de cambios, procedimientos de seguridad y un programa de mantenimiento formal.

En la figura siguiente se esquematiza la relación de las responsabilidades únicas o compartidas entre usuario y proveedor en el desarrollo de un sistema.

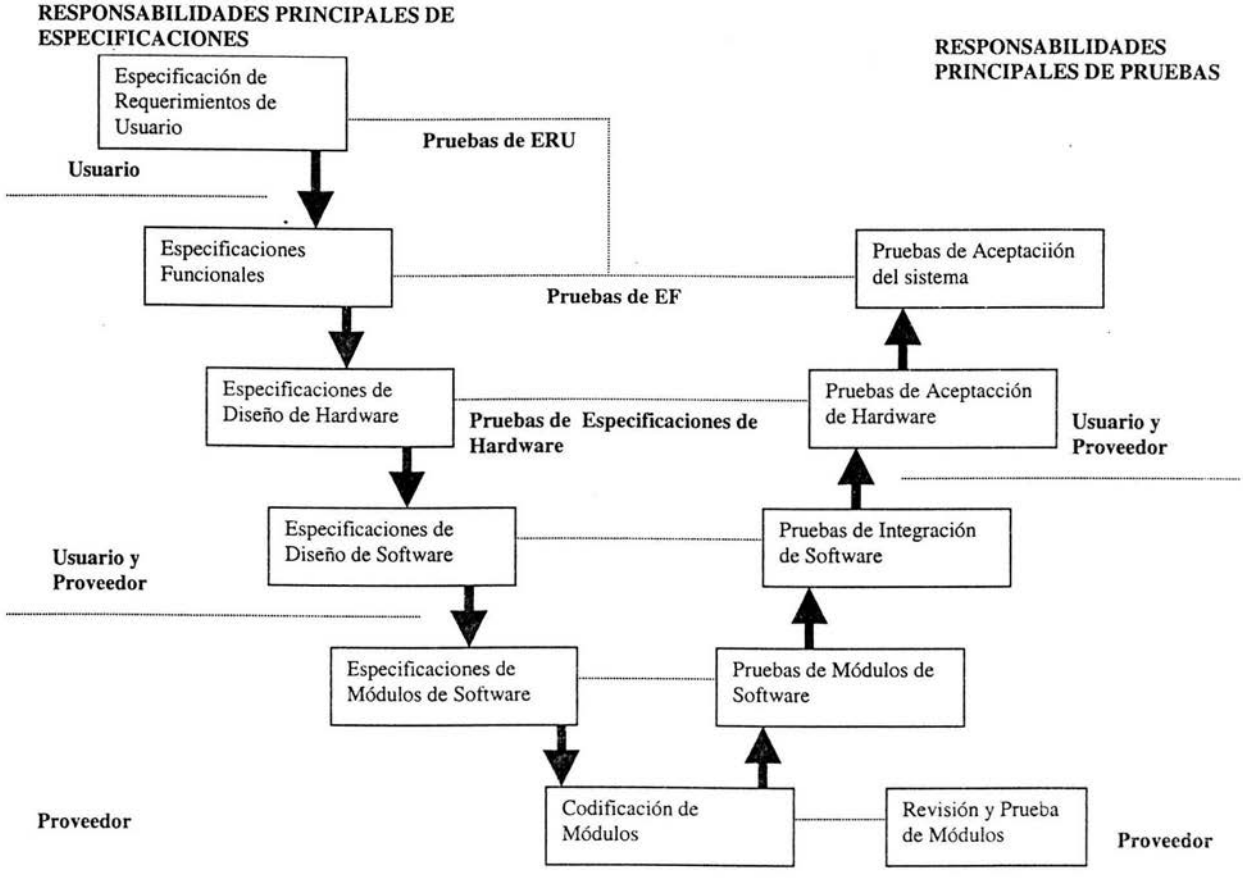


Figura 4. Responsabilidades en el desarrollo de un sistema

7.0 ANALISIS DE INFORMACION Y CONCLUSIONES

La relevancia de las tareas asignadas a los sistemas de cómputo ha ido creciendo a lo largo de los años. La industria ha cambiado considerablemente a través de las fuerzas impulsoras encaminadas a la reducción de costos, mejora de la calidad, confiabilidad y seguridad, además de la reducción de pago por tiempos de desarrollo y diseño.

La satisfacción de tales necesidades requiere del diseño y construcción de sistemas más complejos.

El diseño de tales sistemas requiere de una serie de tareas que permitan cumplir con los requisitos para el usuario final.

El diseño del software sirve como base de la codificación, prueba y mantenimiento. Sin diseño se puede dar origen a un sistema inestable, que falle al realizar pequeños cambios y cuya calidad sea imposible de evaluar.

La función de supervisión del usuario resulta crucial dentro del esquema del ciclo de vida del sistema, ya que la naturaleza tan compleja de las actividades y niveles de responsabilidad técnica dentro del proceso así lo requiere.

A través del ciclo de vida del software se plantea una estructura de verificación y validación que permite detectar posibles fuentes de error que pueda presentar el sistema al momento de su funcionamiento en el sitio de uso. Tales verificaciones y validaciones del sistema, deben contemplarse desde un inicio, de lo contrario será prácticamente imposible corregir debido a que los errores serán detectados cuando el sistema se encuentre en operación.

La parte fundamental para un correcto diseño y construcción se establece en los requerimientos del usuario, que tal y como se visualiza en la serie de actividades

propuestas por el GAMP, son la base del planteamiento de evaluación y construcción del sistema, la cual por su parte estará bajo la tutela y experiencia del proveedor, y por esto se establecen una serie de responsabilidades compartidas entre proveedor y usuario que deberán proporcionar durante cada etapa del desarrollo del sistema, la evidencia documentada de que el sistema cumple con lo solicitado por el usuario y lo prometido por el proveedor. De esta forma, la etapa de diseño de extiende desde establecer los requerimientos del sistema, hasta la aceptación final del sistema en el lugar de uso.

Con base en lo anterior, se concluye que la etapa de diseño resulta crucial para la creación, codificación y validación, ya que en ella se establecen los requerimientos que al final del ciclo de vida, serán las funciones que el sistema llevará a cabo y darán los criterios de evaluación durante su vida útil.

De igual manera, se llega a la conclusión de que la creación de las ERU debe estandarizarse, con la finalidad de que de manera rutinaria se puedan controlar las actividades involucradas, considerándose también como parte integral del sistema de calidad de cada empresa, ya que la función final de los sistemas creados o acoplados en el punto de uso, comprometerá de manera directa la calidad de los insumos generados.

Considerando los puntos anteriores, en las secciones siguientes se muestra una guía de documentos que se deberán generar en el proceso de elaboración de las ERU.

8.0 ANEXO 1

Clasificación de Sistemas

CATEGORIA		DESCRIPCION Y ESQUEMA DE VALIDACION
1	Sistemas Operativos	Ej. Sistemas operativos y Software de red. No se realiza una validación completa, excepto en aplicaciones particulares ya que son sistemas comerciales, sólo se debe llevar un registro de la versión. Las versiones nuevas deben revisarse previamente, considerando el impacto del cambio en la operación del sistema.
2	Instrumentos y Controladores	Ej. Escalas de pesado, lectores de código de barras, o cualquier instrumento que no utilice programas que pueda modificar el usuario. Estos sistemas son configurables y esto deberá estar registrado en los documentos de instalación
3	Paquetes Estándar	Ej. Paquetes estándar como Lotus 1-2-3, Excel, u otros paquetes. No requieren ser validados, sin embargo es necesario verificar los cambios de versión. Las actividades de validación deberán enfocarse en la aplicación que se dé al paquete, concentrándose en los resultados o las aplicaciones que se utilicen para que el paquete realice la función deseada.
4	Paquetes configurables	Ej. LIMS, MRP,SAP o cualquier paquete comercial que deba configurarse de acuerdo a las funciones deseadas. Se recomienda auditar al proveedor y validar las aplicaciones y cualquier codificación nueva.
5	Sistemas de creación específica	Ej. PC's con una aplicación específica, basada en un sistema construido para tal propósito, pero que no pertenece a compañía. En este caso se recomienda una auditoría a proveedor y Validación completa del sistema

9.0 ANEXO 2

GUIA DE ELABORACION DE ESPECIFICACIONES DE REQUERIMIENTOS DE USUARIO

9.1 Generalidades

Las ERU deben definir de manera clara y precisa, que es lo que el sistema debe hacer, las funciones que debe realizar, los datos sobre los cuales el sistema operará, y el entorno de operación.

Las ERU definen también cualquier requerimiento o restricción no funcional, tal como tiempo, costos y que entregables deberán proveerse. Deberá hacerse énfasis en las funciones requeridas y no en el método con el que se implementarán.

Tips para seguir durante la elaboración de las ERU:

1. Cada requerimiento enunciado, deberá ser únicamente referenciado y no más largo de 250 palabras.
2. Los requerimientos no deberán repetirse o contradecirse.
3. Las ERU deberán expresar requerimientos y no soluciones de diseño.
4. Cada requerimiento deberá ser susceptible de probarse.
5. Las ERU deberán ser de fácil comprensión tanto para el usuario, como para el proveedor, evitando ambigüedades o lenguaje coloquial.
6. Cuando sea posible, las ERU deberán distinguir entre requerimientos mandatorios/regulatorios y características deseadas.
7. Debe considerarse una revisión de las ERU entre el proveedor y el usuario, para verificar la correcta interpretación y que los requerimientos han sido considerados o no en las especificaciones funcionales

9.2 Contenido de los documentos

9.2.1 Introducción

Esta sección deberá contener la siguiente información:

1. Quién produce el documento, bajo que autoridad, y con qué propósito.
2. El estatus contractual del documento.
3. Relación con otros documentos.

9.2.2 Resumen global

Esta sección deberá dar una visión general del sistema, explicando porqué es requerido, y qué es requerido del él, y deberá contener:

1. Información de soporte (ej. estrategias corporativas, estudios previos, etc.)
2. Objetivos clave y beneficios.
3. Funciones principales e interfaces

9.2.3 Requerimientos Operacionales

Esta sección deberá establecer los requerimientos operacionales, funciones del sistema, datos e interfaces. Deberá también definir el entorno en el cual el sistema debe operar. Se deben identificar requerimientos críticos de manera específica. Pueden incluirse cuando así se requiera, descripciones de los procesos o diagramas de flujo.

Esta sección se compone de las siguientes subsecciones:

9.2.3.1 Funciones

En esta parte se deberán definir las funciones requeridas del sistema, modos de operación y comportamiento.

Deberá contener lo siguiente:

1. Funciones requeridas, información sobre el proceso o existencia del manual

2. Cálculos, incluyendo todos los algoritmos críticos.
3. Modos de operación (ej. Arranque, apagado, pruebas, caída).
4. Requerimientos de desempeño y tiempo de respuesta (deberán ser cuantitativos y sin ambigüedades).
5. Falla. Acciones requeridas en caso de fallas en el software o hardware seleccionado.
6. Seguridad y garantía.

9.2.3.2 Datos

En esta fase se deberán establecer los requerimientos del manejo de datos, conteniendo lo siguiente:

1. Definición de los datos, incluyendo identificación de los parámetros críticos, los rangos válidos de los datos y los límites.
2. Requerimientos de capacidad.
3. Requerimientos de velocidad de acceso.
4. Requerimientos de archivo o resguardo.

9.2.3.3 Interfaces

Deberán definirse todas las interfaces:

1. Interfaces con los usuarios. Estas deberán definirse en términos de roles (ej. operador de planta, administrador de almacén, gerente de sistemas, etc) y/o funciones, como sea apropiado.
2. Interfaces con otros sistemas.
3. Interfaces con equipo (ej. sensores/arrancadores).Esta incluye listas de Instalación/Operación para sistemas de control de procesos.

9.2.3.4 Entorno

Se deberá definir el entorno en el cual el sistema va a trabajar. Deberá contener las siguientes subsecciones:

1. Layout. La distribución física de la planta o lugar de trabajo puede tener un gran impacto en el sistema (ej. enlaces a larga distancia, limitaciones de espacio, etc.)
2. Condiciones Físicas (ej. suciedad, polvo o ambientes estériles, etc.)

9.2.3.5 Restricciones

Deberán definirse las restricciones sobre las especificaciones del sistema. Deberán incluirse en las siguientes secciones:

1. Escalas de tiempo y sucesos cuando sea apropiado.
2. Compatibilidad. Deberá considerar cualquier sistema o hardware existente, o estrategia de planta o compañía.
3. Disponibilidad. Deberán establecerse requerimientos de confiabilidad, y periodos máximos permisibles de mantenimiento o algún otro tiempo fuera de servicio.
4. Restricciones de procedimiento (ej. obligaciones regulatorias, cuestiones legales, métodos de trabajo y nivel de habilidades de los usuarios.
5. Mantenimiento (ej. facilidad de mantenimiento, capacidad de expansión, probabilidad de mejora, tiempo de vida esperado, soporte a largo plazo).

9.2.3.6 Ciclo de Vida

Esta sección debe definir cualquier requerimiento concerniente al desarrollo del ciclo de vida y deberá contener las siguientes secciones:

1. Desarrollo (ej. procedimiento para la administración del proyectoy aseguramiento de la calidad, métodos de diseño mandatorios).
2. Pruebas (ej. requerimientos especiales de prueba, datos de pruebas, carga de pruebas, simulaciones).
3. Entrega. Esta sección deberá definir que entregables son requeridos. Deberá contener lo siguiente:

   3.1 Cómo se identificarán los elementos entregables,

   3.2 De qué forma serán presentados los entregables (ej. formatos y medios).

   3.3 Documentos. Qué espera entregar el proveedor (ej. especificaciones funcionales, especificaciones de prueba, especificaciones de diseño, etc.).

   3.4 Datos que serán preparados o convertidos.

   3.5 Herramientas.

   3.6 Cursos de entrenamiento

   3.7 Instalaciones de Archivo.
4. Soporte. Esta sección deberá definir el soporte requerido después de la aprobación.

9.2.4 Glosario

Esta sección deberá contener las definiciones de términos que puedan no ser familiares al lector de las ERU.

10.0 BIBLIOGRAFIA

1. Code of Federal Regulations:
   Federal Register 41, 21 C.F.R. Sections 58.61, 58.63 y 58.130 (April, 1, 2002).
2. Code of Federal Regulations:
   Federal Register 41, 21 C.F.R. Section 211.68, (April, 1, 2002).
3. C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik.. Software Development and Reality Construction. Springer Verlang, Germany 1988. Pags. 184-188, 291-301.
4. Mary Ellen Double/Maryann McKendry. Memorias del curso: Computer Validation Compliance, A quality Assurance Perspective, Interpharm Press, Inc., Buffalo Grove, USA, 1994.
5. Food and Drug Administration, General Principles of Software Validation; Final Guideline for Industry and FDA Staff, January, 11, 2002.
6. Chamberlain Richard. Computer Systems Validation for the Pharmaceutical and Medical Device Industries, 2nd Ed. Alaren Press, Libertyville IL 1994. Pags. 1-22, 25-51.
7. Tema: Ciclo de Vida del Software, fecha de acceso, 01 sep 2003, disponible en: http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema03.pdf
8. Food and Drug Administration, Glossary of computerized system and software development terminology, August 1995.
9. El diseño de Software por E.T.S.I Infomática de la Universidad Malacitana, Fecha de acceso: 01 Sep 2003, disponible en: http:/www.Iccnma.es/docencia/ETSIInf/Isd/Isd0.pdf
10. Cooling J.E. Design for Real-Time Systems. University Professional Division Chapman an Hall Ed. G.B. 1st Ed. 1991. Pags. 1-3, 51-69, 89-95.
11. Tema: El diseño de software, fecha de acceso: 17 jun 2003, disponible en: http:/grulla.hispalinux.es/enunciados/design_stmas.pdf.
12. ISPE, The Society for Pharmaceutical and Medical Devices Professionals. GAMP Guide for Validation of Automated Systems in Pharmaceutical Manufacture Volume 1, Part 1, Part 2, March 1998. pags. 10-24, Apéndice 5
13. ISPE, The Society for Pharmaceutical and Medical Devices Professionals. GAMP Guide for Validation of Automated Systems in Pharmaceutical Manufacture Volume 2, March 1998. Pags. 7-10.
14. H.S. Jagdev, J. Browne, P. Jordan. Verification and Validation issues in manufacturing models. Computers in Industry, 25 (1995) 331-353.
15. C. Burgess. A broader view of Validation; the balance of compliance and science. Laboratory Automation and Information Management, 31 (1995) 35-42.
16. M.V. Zelkowitz, D. Wallace, Experimental Validation in software Engineering. Information and Software Technology. 39 (1997)735-743.
17. M. Webb. Computer systems implementation, batch standards and validation. ISA Transactions, 34 (1995),379-385.
18. D.V. Boriani. Formal specification methods in engineering design. ISA Transactions, Vol. 36 No. 2, (1997),123-129.
19. J.P. Gray, A. Liu, L. Scott. Issues in engineering tool construction. Information and Software Technology., 42, (2000), 73-77.

20. C.R. Turner, A. Fuggetta, L. Lavazza, A.L. Wolf. A conceptual basis for feature engineering, The Journal of Systems and Software. 49 (1999), 3-15.
21. G. Butcher, C. Schroeder. A Model for Addressing software volatility in new system development. Information Sciences. 118 (1999) 121-143.
22. F. Huq. Testing in the software development life cycle: now or later. International Journal of Project Management. 18 (2000) 243-250.
23. J. Tian. Measurement and continous improvement of software reliability throughout software life-cycle. The Journal of systems and Software.47 (1999) 189-195.
24. N. Ashrafi, J.P. Kuilboer, J.M. Wagner. Expert systems reliability: A life cycle approach. Information and Management. 28 (1995) 405-414.
25. F. Stallinger, P. Grünbacher. System dynamics modeling and simulation of collaborative requirements engineering. The Journal of Systems and Softwrae. 59 (2001) 311-321.
26. J.S. Osmundson, J.B. Michael, M.J. Machniak, M.A. Grossman. Quality management metrics for software development. Information and Management. 2033 (2002) 1-14.
27. Z. Xu, T.M. Khoshgoftaar, E.B. Allen. Application of fuzzy expert systems in assessing operational risk of software. Information and Software Technology. Xx (2003) 1-16.
28. F. Alonso, N. Juristo, J.L. Maté, J. Pazos. Software engineering and knowledge engineering: Towards a common life-cycle. Journal Systems Software. 33 (1996) 65-79.
29. H.R. Hartson. Human – Computer interaction: Interdisciplinary roots and trends. The Journal of Systems and Software. 43 (1998) 103-118.
30. R.D. McDowall. Practical computer validation for pharmaceutical laboratories. Journal of Pharmaceutical and Biomedical Analysis. 14 (1995) 13-22.
31. T. Stokes. Laboratory software validation – from corporate policy to bench top systems. Laboratorty Automation and information Management. 31 (1995) 1-10.
32. B. Boehm, A. Egyed. Optimizing software product integrity through life-cycle process integration. Computer Standards and Interfaces. 21 (1999) 63-75.
33. D. Welzel, HL. Hausen. A Method for software evaluation Contribution of the European project SCOPE to international Standards. Computer Standards and Interfaces. 17 91885) 121–129.
34. S. Easterbrook, J. Callahan. Formal methods for verification and validation of partial specifications: A case of study. Journal Systems Software, 40 (1998) 199 210.
35. N.G.P.C. Mahalik, S.K. Lee. Design and development of system level software tool for DCS simulation.
36. D.L. Green, A. M. Dowell. How to design, verify and validate emergency shutdown systems. ISA Transaction. 34 (1995) 261-272.
37. N. Navet, Y. Song. Validation of in-vehicle real-time applications. Computers in Industry. 46 (2001) 107-122.
38. L. Arnold, P. Frauch, A. Klöti, M. Staub. Software assessment under consideration of validationaspects: PPS and PMS systems. Pharmaceutica Acta Helvetiae. 72 (1998) 327 – 332.

39. A.K. Zaidi, A.H. Levis. Validation and verification of decision making rules. Automatica. 33 (1997) 155-169.
40. G. Iazeolla. Performance Validation of Software systems. Performance Evaluation. 45 (2001) 77-79.
41. U. Herzog, J. Rolia. Performance validation tools for software/hardware systems. Performance Evaluation. 45 (2001) 125-146.
42. F. Alquilani, S. Balsamo, P. Inverardi. Performance analysis at the software architectural design level. Performance Evaluation. 45 (2001) 148-178.
43. S. Biffl. Evaluating defect estimation models with major defects. Tha Journal of Systems and Software. 65 (2003) 13-29.
44. W. Li, V.C. Arena, N.B. Sussman, S. Mazumdar. Model validation software for classification models using repeated partitioning: MVREP. Computer Methods and Programs in Biomedicine. 00 (2002) 1-7.
45. N. Maiden, et al. A co-operative scenario based approach to acquisition and validation of system requirements: How exceptions can help !. Interacting with computers. 11 (1999) 645- 664.
46. D. Friedli, W. Kappeler, S. Zimmermann. Validation of computer systems: Practical testing of a estándar LIMS. Pharmaceutica Acta Helvetiae. 72 (1998) 343 – 348.
47. Tema: Análisis y Diseño de Sistemas, fecha de acceso 22 Oct 2003, Disponible en: http//www.monografias.com/trabajos/anaydisesis/anayisesis.html